

OLIT Reference Manual

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



SunSoft
A Sun Microsystems, Inc. Business

© 1994 Sun Microsystems, Inc.
2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX[®] and Berkeley 4.3 BSD systems, licensed from UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and the University of California, respectively. Third-party font software in this product is protected by copyright and licensed from Sun's font suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, the Sun logo, Sun Microsystems, Sun Microsystems Computer Corporation, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and certain other countries. UNIX is a registered trademark of Novell, Inc., in the United States and other countries; X/Open Company, Ltd., is the exclusive licensor of such trademark. OPEN LOOK[®] is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc. All other product names mentioned herein are the trademarks of their respective owners.

All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCstorage, SPARCware, SPARCcenter, SPARCclassic, SPARCcluster, SPARCdesign, SPARC811, SPARCprinter, UltraSPARC, microSPARC, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun[™] Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

X Window System is a product of the Massachusetts Institute of Technology.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.



Contents

1. Introduction	1
Differences From OLIT Release 3.3	1
Motif and OPEN LOOK Interoperability Issues	2
2. Common Resources	5
Resources Summary Tables	6
Resource Files	7
Resource File Bindings	7
OLIT Toolkit Resources	7
Core Resources	17
Composite Resources	23
Primitive Resources	24
Manager Resources	31
Shell Resources	31
WMShell Resources	35
VendorShell Resources	42

TransientShell Resources	50
TopLevelShell Resources	50
ApplicationShell Resources	51
Flat Resources.....	52
3. Activation Types	61
Activation Type Description.....	61
Common Activation Types.....	68
4. Internationalization Features	71
Introduction	71
System Requirements	72
Issues Involved in Internationalizing Applications.....	72
Locale Setting.....	73
Character Encoding and Text Formats	74
Localized Text Handling.....	79
Input Method.....	80
Standards	89
5. Toolkit Functions	91
Initialization and Activation Functions	92
Buffer Functions.....	95
Cursor and Pixmap Functions	99
Display Functions	108
Drag and Drop Functions.....	109
Dynamic Resource Functions.....	140
Error Functions.....	142

Help Function.	146
Input Focus Functions.	150
Multiple Visual Functions.	154
Packed Widget Function.	156
Pixel Conversion Functions	158
Protocol Function.	160
Regular Expression Functions	161
Text Buffer Functions.	163
Text Buffer Functions for Internationalization.	176
Text Selection Operations	204
Toolkit Resource Functions.	206
Virtual Event Functions.	207
6. Widget Reference (A – C).	217
AbbrevMenuButton Widget	217
BulletinBoard Widget	225
Caption Widget	229
CheckBox Widget	237
ControlArea Widget	249
7. Widget Reference (D – F).	259
DrawArea Widget	259
DropTarget Widget	266
Exclusives Widget	277
FileChooser Widget	284
FileChooserShell Widget	311

Flat Widgets	321
FlatCheckBox Widget	329
FlatExclusives Widget	337
FlatNonexclusives Widget	347
Flat Widget Functions	354
FontChooser Widget	357
FontChooserShell Widget	375
FooterPanel Widget	381
Form Widget	385
8. Widget Reference (G – P).	395
Gauge Widget	395
Gauge Function	402
MenuButton Widget	403
MenuShell Widget	414
Nonexclusives Widget	428
NoticeShell Widget	433
NumericField Widget	443
OblongButton Widget	464
PopupWindowShell Widget	475
9. Widget Reference (R – S).	489
RectButton Widget	489
RubberTile Widget	502
Scrollbar Widget	508
ScrolledWindow Widget	529

ScrollingList Widget	547
ScrollingList Functions	578
Slider Widget	586
StaticText Widget	600
Stub Widget	609
10. Widget Reference (T – Z)	623
TextEdit Widget	623
TextEdit Functions	660
TextField Widget	665
TextField Functions	686
TextLine Widget	688
TextLine Functions	708

Preface

Who Should Use This Book

This manual is for applications programmers who want to create applications using the OPEN LOOK[®] Intrinsic Toolkit (OLIT[™]). OLIT is a user interface toolkit based on the X Window System[™] Xt Intrinsic from MIT. OLIT implements the OPEN LOOK Graphical User Interface (GUI); it consists of a set of widgets and the Xt Intrinsic version R5.

This manual provides reference information for the widgets and associated functions of OLIT. It is not intended to provide instruction for first-time OLIT programmers. Programmers who want to learn how to use OLIT should consult:

- *X Window System Programming and Applications with Xt – OPEN LOOK Edition*, by John Pew, published by Prentice Hall, 1992.
- *OLIT Quick Start Programmer's Guide*, Part Number 801-5317-10, Sun Microsystems, Inc., 1993.

Programmers may also want to consult the following manuals for further reference information:

- *X Window System Toolkit, The Complete Programmer's Guide and Specification*, Digital Press, 1992.
- *X Window System, The Complete Guide to Xlib, X Protocol, ICCM, XLFD*, Digital Press, 1992.

-
- *OPEN LOOK Graphical User Interface Functional Specification*, Addison-Wesley Publishing Company, Inc., 1989.
 - *User Interface Specification for Mouseless Operation of the OPEN LOOK Graphical User Interface*, Part Number 800-6816-01, Sun Microsystems, Inc., Version 1.0 – August 1, 1991.

The publications that are not Sun reference manuals are available from SunExpress™ (call 1-800-873-7869) or your local computer bookstore.

How This Book Is Organized

This manual consists of ten chapters. The following is a brief description of each chapter.

Chapter 1, “Introduction,” contains a description of the differences from OLIT 3.3 to 3.4 and Motif interoperability. It also shows a hierarchy diagram of the OLIT widget classes.

Chapter 2, “Common Resources,” describes resources that are common to several widgets.

Chapter 3, “Activation Types,” describes virtual events and activation types that are common to several widgets.

Chapter 4, “Internationalization Features,” describes how OLIT applications can be internationalized.

Chapter 5, “Toolkit Functions,” describes functions that are used to manipulate many of the widgets.

Chapter 6, “Widget Reference (A – C),” is the first of five chapters that describe the widgets in the OLIT widget set and sets of utility, or convenience, functions that can be used to interact with certain specific widgets. The widgets are listed in alphabetical order, with any dedicated functions following the applicable widgets. Each of the five chapters in this group represents a range of widgets, divided into alphabetic groups: A–C, D–F, G–P, R–S, and T–Z.

Introduction



This chapter describes the differences between OLIT™ releases 3.3 and 3.4 and OLIT interoperability with Motif. It also presents a diagram of the OLIT widget class hierarchy.

Differences From OLIT Release 3.3

Solaris x86 Support

OLIT 3.4 now supports Solaris x86, sometimes referred to as “Solaris on Intel”. Since the keyboards on most non-Sparc systems don’t have a “meta” key, support for these systems requires some mechanism which lets users generate a *meta-key event* without pressing an actual meta key (◊). On non-SPARC keyboards this is done by pressing the Control and Alt keys at the same time. This action is functionally identical to pressing the meta key on SPARC keyboards.

XtNctrlAltMetaKey Resource

A new Toolkit resource has been added which controls the translation of the Control-Alt key combination into the *meta-key event*. This resource may be desirable in conjunction with programs that use menu accelerators or mnemonics which conflict with the meta key substitute (Control-Alt). See page 12 for details.

Motif and OPEN LOOK Interoperability Issues

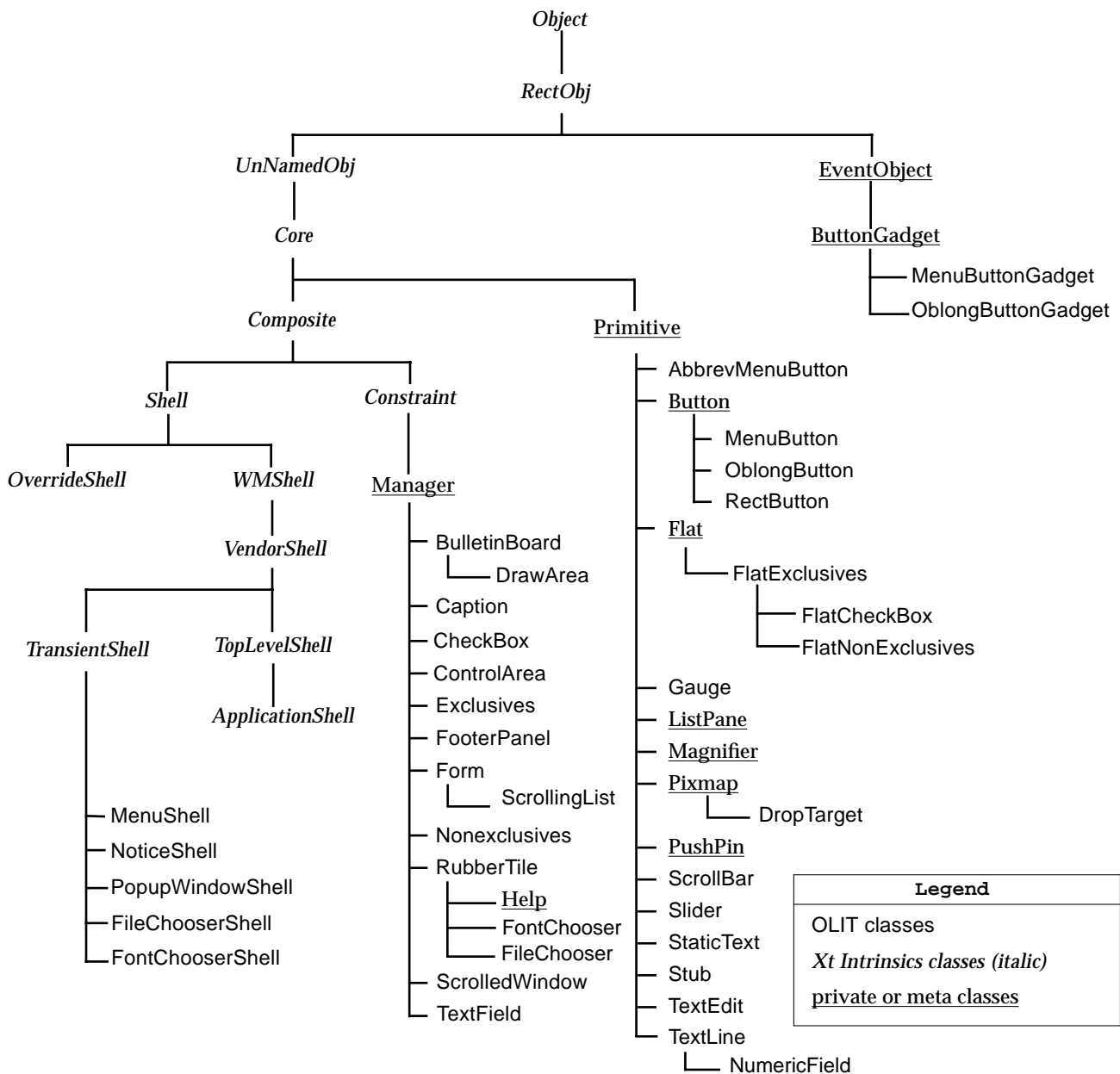
Interoperability Between Motif Applications and OPEN LOOK Applications

- XView and OLIT applications do not support Primary copy, move, and link.
- Secondary selections between Motif applications and XView or OLIT applications do not work.
- Copying or cutting from a Motif application into the Clipboard and then pasting to an XView text subwindow will not work. Pasting to OLIT applications or XView panel text fields will work.
- Copying or cutting and pasting Asian text between a Motif application and an XView or OLIT application does not work.
- Drag and drop between Motif applications and XView or OLIT applications does not work.

Interoperability Between the Motif Window Manager (mwm) and OPEN LOOK Applications

- Window manager decorations for some XView and OLIT windows are not correct. Example: some applications may exhibit extra titles, or extra resize handles. This does not affect application functionality.
- Executing the window manager function `f.kill` on base frame windows of an XView application (typically done by pulling down the default window menu and selecting Quit or Close) may exit the application without user confirmation, or any further application-specific processing.
- If the resource `keyboardFocusPolicy` is set to pointer, then XView or OLIT applications will get the focus when the mouse is moved into their windows, but the caret may not always darken. To darken, click inside the window.
- The default colors for XView and OLIT applications are not those of Motif.
- Dragging and dropping data onto an icon of a closed XView or OLIT application will not work.

OLIT Class Hierarchy



Common Resources



This chapter describes resources that are common to several widgets.

<i>OLIT Toolkit Resources</i>	<i>page 7</i>
<i>Core Resources</i>	<i>page 17</i>
<i>Composite Resources</i>	<i>page 23</i>
<i>Primitive Resources</i>	<i>page 24</i>
<i>Manager Resources</i>	<i>page 31</i>
<i>Shell Resources</i>	<i>page 31</i>
<i>WMShell Resources</i>	<i>page 35</i>
<i>VendorShell Resources</i>	<i>page 42</i>
<i>TransientShell Resources</i>	<i>page 50</i>
<i>TopLevelShell Resources</i>	<i>page 50</i>
<i>ApplicationShell Resources</i>	<i>page 51</i>
<i>Flat Resources</i>	<i>page 52</i>

The resources are listed according to the classes from which they derive. A widget inherits resources from an ancestor class in the widget class tree; see “OLIT Class Hierarchy” on page 3. For example, every widget has the `XtNbackground` resource, because all widgets descend from `coreWidgetClass` and this resource is defined as a Core resource. Likewise, widgets that descend from `primitiveWidgetClass` have Primitive resources, such as `XtNaccelerator`.

Resources Summary Tables

In addition to class resources, there are application wide resources that apply to an entire OLIT application. These resources are called Toolkit Resources and are also listed in this chapter.

Each widget described in Chapters 6–10 has summary tables listing all the application-accessible resources for that widget. If a resource is shared, a detailed description of that resource is in Chapter 2. If the resource is unique to that widget, or if it has some different use or meaning to that widget, then it is described in the individual widget section.

Resources Summary Tables

The summary tables in this chapter and the widget descriptions contain the most commonly sought information about a resource:

- (C-language) type
- Default value (if any)
- Accessibility of the resource

Accompanying the individual description of a resource is a one-line table of the preceding information, plus the class name of the resource. The class name of the resource is not repeated in the summary table, and is only found with the individual description. The values in the “Access” column of the summary tables are abbreviated as follows:

Abbreviation	Meaning
S	(Set.) The application can set the resource using <code>XtSetValues()</code> .
G	(Get.) The application can get the resource using <code>XtGetValues()</code> .
I	(Init.) The application can set the resource when it creates the widget.
O	(Other.) The resource is set or retrieved in some other way, typically by a function just for this purpose; an explanation of this is given with the resource description that follows the table.
D	(Dynamic.) The resource is updated dynamically when the window server resource database changes (for example, when <code>Properties</code> or <code>xrdb</code> is used).
n/a	(Not applicable.) The widget does not use this resource and is unaffected by its value.

Resource Files

The value taken by a resource can often be set in a resource file, typically `$HOME/.Xdefaults`. For further details about resource specification in a resource file, see Chapter 9 of the *Xt Intrinsic Programming Manual*.

Resource File Bindings

In many cases the C-language binding for a resource differs from that used in the resource file. For example, to display keyboard accelerators on controls, a program would set `XtNshowAccelerators` to `OL_DISPLAY`. However, in a resource file, this would be done by a line such as:

```
*showAccelerators: display
```

To show both the C-language value and the resource file value, this manual uses the following convention to specify valid values for resources:

```
OL_DISPLAY/"display"
```

OLIT Toolkit Resources

OLIT uses several resources on an application-wide basis to determine the state of an OPEN LOOK application, as shown in Table 2-1. Unlike the other resources in this chapter, the OLIT Toolkit resources affect an application as a whole, and are unrelated to the widget class hierarchy. Because of this, these resources are not repeated in each widget description. These resources are not retrieved and set in the same manner as the other resources in this manual. Instead of using `XtGetValues()` and `XtSetValues()`, the corresponding functions `OlGetApplicationValues()` and `OlSetApplicationValues()` are used to retrieve or set the values of these resources. (See “Toolkit Resource Functions” on page 206.)

Table 2-1 OLIT Toolkit Resources Summary

Name	Type	Default	Access
<code>XtNbeep</code>	<code>OlDefine</code>	<code>OL_BEEP_ALWAYS</code>	SGIO
<code>XtNbeepVolume</code>	<code>int</code>	0	SGIO
<code>XtNcolorTupleList</code>	<code>OlColorTuple</code>	NULL	GIO
<code>XtNcontrolName</code>	<code>String</code>	“Ctrl”	IOD
<code>XtNdragRightDistance</code>	<code>Dimension</code>	100 (pixels)	SGIO

OLIT Toolkit Resources

Table 2-1 OLIT Toolkit Resources Summary (Continued)

Name	Type	Default	Access
XtNgrabPointer	Boolean	TRUE	SGIO
XtNhelpModel	OlDefine	OL_POINTER	SGIO
XtNinputFocusFeedback	OlDefine	OL_SUPERCARET	IO
XtNlockName	String	“Lock”	IOD
XtNmenuMarkRegion	Dimension	10 (pixels)	SGIO
XtNctrlAltMetaKey	Boolean	(see description)	GI
XtNmnemonicPrefix	Modifiers	Mod1Mask	SGIO
XtNmod1Name	String	“Meta”	IOD
XtNmod2Name	String	“ModeSwitch”	IOD
XtNmod3Name	String	“NumLock”	IOD
XtNmod4Name	String	“Alt”	IOD
XtNmod5Name	String	“Mod5”	IOD
XtNmouseDampingFactor	Cardinal	8 (pixels)	SGIO
XtNmouseless	Boolean	FALSE	GIO
XtNmultiClickTimeout	Cardinal	200 (msec)	SGIO
XtNmultiObjectCount	Cardinal	3	SGIO
XtNolDefaultFont	String	Lucida	SGIO
XtNscale	int	12	SGIO
XtNselectDoesPreview	Boolean	TRUE	SGIO
XtNshiftName	String	“Shift”	IOD
XtNshowAccelerators	OlDefine	OL_DISPLAY	SGIO
XtNshowMnemonics	OlDefine	OL_UNDERLINE	SGIO
XtNthreeD	Boolean	TRUE	SGIO

XtNbeep

Class	Type	Default	Access
XtCBeep	OlDefine	OL_BEEP_ALWAYS	SGI

Synopsis: The type of objects that can generate audible warnings to the user.

OLIT Toolkit Resources

Values: OL_BEEP_NEVER/"never" - Never generate audible warnings.
 OL_BEEP_ALWAYS/"always" - Any object can generate audible warnings.
 OL_BEEP_NOTICES/"notices" - Only Notices should generate audible warnings.

XtNbeepVolume

Class	Type	Default	Access
XtCBeepVolume	int	0	SGI

Synopsis: The percentage of the keyboard's normal beep to use for audible warnings.

Values: $-100 \leq \text{XtNbeepVolume} \leq 100$

See `XBell()` in the *XLib Reference Manual*.

XtNcolorTupleList

Class	Type	Default	Access
XtCColorTupleList	OIColorTuple	NULL	GIO

Synopsis: The alternative background colors for 3D rendering.

Values: Any valid color value for the display.

The emphasis colors are defined as:

- BG0 - highlight color, for emphasis
- BG1 - normal background
- BG2 - invoked background, for indented and menu choices
- BG3 - shadow color, used with highlight for 3D effect

These colors can be specified in a defaults file as tuples (BG0, BG1, BG2, BG3) of either numeric or character color values, or a mixture of the two.

```
*colorTupleList:      (#FFFFFF,#000000,#000000,#AAAAAA),\
                      (pink,gray,green,blue)
```

The effect of the resource is to make the BG0, BG2, and BG3 colors the highlight, indent, and shadow colors for those widgets whose background color matches BG1.

This color scheme is slightly different from OPEN LOOK in that BG2 and BG3 are not necessarily derived from BG1 by the addition of gray. This scheme is intended to provide an alternative to the OPEN LOOK arrangement.

OLIT Toolkit Resources

Normally, the toolkit will compute BG0–BG3 according to the *OPEN LOOK GUI Functional Specification*. Where the toolkit fails because it is faced with a color-poor display, the application can use `XtNcolorTupleList` to override the tuple list.

Note – Color tuples cannot be manipulated from a program by ordinary `XtGetValues()` and `XtSetValues()` functions. Instead, the program must use the `XrmQGetResource()` and `XrmQPutResource()` functions defined in the *XLib Reference Manual*.

XtNcontrolName

Class	Type	Default	Access
XtCControlName	String	“Ctrl”	IOD

Synopsis: The text printed for the Control Key in an accelerator label.

XtNdragRightDistance

Class	Type	Default	Access
XtCDragRightDistance	Dimension	100 (pixels)	SGI

Synopsis: The number of pixels the pointer must be dragged to post the `MenuButton`’s submenu.

This resource specifies how far, in pixels, the pointer must be dragged over a `MenuButton` with the `MENU` button depressed to post the `MenuButton`’s submenu. The direction of the drag is to the right. This resource only applies to `MenuButtons` on *press-drag-release* menus. See the *OPEN LOOK GUI Functional Specification* for a description of *press-drag-release* menus.

XtNgrabPointer

Class	Type	Default	Access
XtCGrabPointer	Boolean	TRUE	SGI

Synopsis: The enabling of pointer grabs.

Values: `TRUE/“true”` – Allow pointer grabs.
`FALSE/“false”` – Do not allow pointer grabs.

OLIT Toolkit Resources

Setting `XtNgrabPointer` to `FALSE` prohibits OLIT from making any active pointer grabs. This is sometimes useful when debugging OLIT applications; for example, a developer debugging an application that pops up a menu might want to set this resource to `FALSE` in order to continue using the mouse while the menu is popped up.

(For example, use the

```
-xrm "*grabPointer: false"
```

argument when running the program.) However, this should be used only for debugging, since the grab ensures all pointer events get delivered to the menu.

XtNhelpModel

Class	Type	Default	Access
XtCHelpModel	OlDefine	OL_POINTER	SGI

Synopsis: The model of how help functions follow the pointer when the HELP key is pressed.

Values: `OL_POINTER/"pointer"` - When the HELP key is pressed, the item under the pointer is the subject of the help message.
`OL_INPUTFOCUS/"inputfocus"` - The subject of the help message is the item with the keyboard input focus.

XtNinputFocusFeedback

Class	Type	Default	Access
XtCInputFocusFeedback	OlDefine	OL_SUPERCARET	IO

Synopsis: The keyboard input focus feedback style.

Values: `OL_SUPERCARET/"supercaret"` - Display a SuperCaret on the object with the input focus.
`OL_INPUT_FOCUS_COLOR/"inputFocusColor"` - Highlight the object with the input focus in the `XtNinputFocusColor` (see page 27).

XtNlockName

Class	Type	Default	Access
XtCLockName	String	"Lock"	IOD

Synopsis: The text printed for the caps lock key in an accelerator label.

XtNmenuMarkRegion

Class	Type	Default	Access
XtCMenuMarkRegion	Dimension	10 (pixels)	SGI

Synopsis: The width (in pixels) of the MenuButton's *menu mark region*. If the pointer is moved into this region with the MENU mouse button depressed, the MenuButton's submenu is posted.

XtNctrlAltMetaKey

Class	Type	Default	Access
XtCCtrlAltMenuKey	Boolean	(see Synopsis)	GI

Synopsis: Controls whether or not the Control-Alt key combination generates the *meta-key event*. On systems with keyboards that do not have a meta key, this resource defaults to TRUE. On systems with keyboards that do have a meta key, this resource defaults to FALSE.

XtNmnemonicPrefix

Class	Type	Default	Access
XtCMnemonicPrefix	Modifiers	Mod1Mask	SGI

Synopsis: The modifier key that must accompany the mnemonic character when activating an object from the keyboard, if that object is not on a menu.

Values: Any valid X11 `KEYSYM` value (see the *XLib Reference Manual*) or any valid Xt translation syntax for a Key event (see the *Xt Intrinsic Reference Manual*).

*XtNmod1Name/
XtNmod2Name/
XtNmod3Name/
XtNmod4Name/
XtNmod5Name*

Class	Type	Default	Access
XtCMod1Name	String	"Meta"	IOD
XtCMod2Name	String	"ModeSwitch"	IOD
XtCMod3Name	String	"NumLock"	IOD
XtCMod4Name	String	"Alt"	IOD
XtCMod5Name	String	"Mod5"	IOD

Synopsis: The text displayed for the Modifier1 to Modifier5 keys in an accelerator label.

XtNmouseDampingFactor

Class	Type	Default	Access
XtCMouseDampingFactor	Cardinal	8 (pixels)	SGI

Synopsis: The number of pixels the pointer can be moved before a drag operation is initiated.

XtNmouseless

Class	Type	Default	Access
XtCMouseless	Boolean	FALSE	GI

Synopsis: The enabling of mouseless operations.

Values: TRUE/"true" - Mouseless mode is enabled.

FALSE/"false" - Mouseless mode is disabled.

When *XtNmouseless* is set to FALSE, only text input objects can acquire keyboard input focus. The traversal mechanism moves the keyboard input focus only through the text input objects.

When *XtNmouseless* is set to TRUE, some widgets, in addition to the text input objects, also can acquire keyboard input focus. Keyboard input focus feedback in non-text input objects can be specified through the *XtNinputFocusFeedback* resource: by default, a SuperCaret. (See

OLIT Toolkit Resources

XtNinputFocusFeedback on page 11 and XtNinputFocusColor on page 27.) Refer to “Input Focus Functions” on page 150 for more information on the behavior of input focus functions when XtNmouseless is set to FALSE.

XtNmultiClickTimeout

Class	Type	Default	Access
XtCMultiClickTimeout	Cardinal	200 (msec)	SGI

Synopsis: The number of milliseconds that determines a multi-click.

This resource specifies the time interval in milliseconds within which two successive button clicks are considered a multi-click, as long as the pointer does not move beyond the XtNmouseDampingFactor value between the clicks.

XtNmultiObjectCount

Class	Type	Default	Access
XtCMultiObjectCount	Cardinal	3	SGI

Synopsis: The repeat count for OL_MOVE *direction* keys.

This resource determines the number of times the OL_MULTIRIGHT, OL_MULTILEFT, OL_MULTIUP, and OL_MULTIDOWN keys repeat the OL_MOVERIGHT, OL_MOVELEFT, OL_MOVEUP, and OL_MOVEDOWN keys, respectively. (These actions are normally on the Ctrl+arrow keys.)

XtNolDefaultFont

Class	Type	Default	Access
XtCOIDefaultFont	String	Lucida	SGI

Synopsis: The value for the XtNfont resource on most widgets that do not set their own.

Values: Any Font specified in the XLFd format (see the *XLib Reference Manual*). If the RESOLUTION_X, RESOLUTION_Y, and PIXEL_SIZE fields in an XLFd font name are set to ‘*’, the toolkit attempts to load a font with RESOLUTION_X and RESOLUTION_Y derived from the screen resolution of the widget’s screen. If the POINT_SIZE and PIXEL_SIZE fields are set to ‘*’, the toolkit attempts to load the given font with a POINT_SIZE derived from the XtNscale of the widget (POINT_SIZE = 10 × XtNscale).

OLIT Toolkit Resources

In the C locale, the default value of `XtNo1DefaultFont` is the 75×75 resolution Lucida sans serif font. In other locales, the default value is a font, or a comma-separated list of fonts, suitable for that locale. The `POINT_SIZE` is derived from the `XtNscale` resource of the widget in all locales.

Note – If a value is specified for the Xt Intrinsic `XtNxtDefaultFont` (or `XtNxtDefaultFontSet`) resource, it may override the `XtNo1DefaultFont` resource. See “XtNfont” on page 26 for a detailed description of the exact algorithm used.

XtNscale

Class	Type	Default	Access
<code>XtCScale</code>	int	12	SGI

Synopsis: The size of graphical elements (widgets), proportioned to the size of text, measured in points (1/72 inch).

The toolkit supports sizes of 10, 12, 14, 16, 19, 20, and 24 points; other values may not display correctly.

XtNselectDoesPreview

Class	Type	Default	Access
<code>XtCSelectDoesPreview</code>	Boolean	TRUE	SGI

Synopsis: The behavior of the SELECT mouse button when it is pressed over a `MenuButton` or an `Abbreviated MenuButton` widget.

Values: `TRUE`/`"true"` – Pressing SELECT will cause the `MenuButton` to preview the submenu’s default item and releasing the SELECT button will activate the default item.
`FALSE`/`"false"` – Pressing SELECT is the same as pressing MENU.

XtNshiftName

Class	Type	Default	Access
<code>XtCShiftName</code>	String	“Shift”	IOD

Synopsis: The text printed for the shift key in an accelerator label.

XtNshowAccelerators

Class	Type	Default	Access
XtCShowAccelerators	OlDefine	OL_DISPLAY	SGI

Synopsis: The display of keyboard accelerators on controls.

Values: OL_DISPLAY/"display" - Keyboard accelerators are displayed.
 OL_INACTIVE/"inactive" - Keyboard accelerators are not displayed and controls ignore them.
 OL_NONE/"none" - Keyboard accelerators are not displayed, but still work.

XtNshowMnemonics

Class	Type	Default	Access
XtCShowMnemonics	OlDefine	OL_UNDERLINE	SGI

Synopsis: The display of keyboard mnemonics on controls.

Values: OL_UNDERLINE/"underline" - Display mnemonics in the controls by drawing a line under the character in the font color.
 OL_DISPLAY/"display" - Same as OL_UNDERLINE.
 OL_HIGHLIGHT/"highlight" - Display the mnemonic character with background and foreground colors reversed. When highlighting a character that is displayed on a pixmap background, the mnemonic character will be drawn in a solid color.
 OL_INACTIVE/"inactive" - Turn off the mnemonic display and make the mnemonic key inactive.
 OL_NONE/"none" - Do not display mnemonic characters.

XtNthreeD

Class	Type	Default	Access
XtCThreeD	Boolean	TRUE	SGI

Synopsis: The rendering method for visuals.

Values: TRUE/"true" - The visuals have a three-dimensional look.
 FALSE/"false" - The visuals have a two-dimensional look.

Core Resources

These are the resources of the Core class, of which all widget classes are subclasses. See the diagram in “OLIT Class Hierarchy” on page 3.

Table 2-2 Core Resources Summary

Name	Type	Default	Access
XtNaccelerators	AcceleratorTable	NULL	SGI
XtNancestorSensitive	Boolean	TRUE	GO
XtNbackground	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth	Dimension	1	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	int	(parent's)	GI
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	GI
XtNsensitive	Boolean	TRUE	GIO
XtNtranslations	XtTranslations	NULL	SGI
XtNwidth	Dimension	0	SGI
XtNx	Position	0	SGI
XtNy	Position	0	SGI

XtNaccelerators

Class	Type	Default	Access
XtCAccelerators	AcceleratorTable	NULL	SGI

Synopsis: The table of accelerator translations for the widget.

Core Resources

XtNancestorSensitive

Class	Type	Default	Access
XtCSensitive	Boolean	TRUE	GO

Synopsis: TRUE if the immediate parent of the widget will receive input events.

Note - `XtIsSensitive()` will return TRUE if both this and `XtNsensitive` are TRUE. To preserve data integrity, neither this nor `XtNsensitive` should be set directly; use `XtSetSensitive()`.

XtNbackground

Class	Type	Default	Access
XtCBackground	Pixel	XtDefaultBackground	SGID

Synopsis: The background color for the widget.

Values: Any `Pixel` value valid for the current display, or any name from the `$OPENWINHOME/lib/rgb.txt` file. (Pixel values are used in C programs, `rgb.txt` values in X resource files. See “Resource Files” on page 7.)

Note - Widgets do not inherit the background color from their parent. Also, any color set by the application when a widget is created, or in a later call to `XtSetValues()`, will override the colors set by the user. Applications should consider this and try to allow maximum flexibility for the user.

XtNbackgroundPixmap

Class	Type	Default	Access
XtCPixmap	Pixmap	XtUnspecifiedPixmap	SGI

Synopsis: The pixmap to be used for tiling the background.

The first tile is placed at the upper left-hand corner of the widget’s window.

Note - This resource takes precedence over the `XtNbackground` resource.

XtNborderColor

Class	Type	Default	Access
XtCBorderColor	Pixel	XtDefaultForeground	SGID

Synopsis: The color of the border.

Values: Any `Pixel` value valid for the current display, or any name from the `$OPENWINHOME/lib/rgb.txt` file. (Pixel values are used in C programs, `rgb.txt` values in X resource files. See “Resource Files” on page 7.)

XtNborderPixmap

Class	Type	Default	Access
XtCPixmap	Pixmap	XtUnspecifiedPixmap	SGI

Synopsis: The pixmap to be used for tiling the border.

The first tile is placed at the upper left hand corner of the border.

Note – This resource takes precedence over the `XtNborderColor` resource.

XtNborderWidth

Class	Type	Default	Access
XtCBorderWidth	Dimension	1	SGI

Synopsis: The width in pixels of the border for a widget.

Values: $0 \leq \text{XtNborderWidth} \leq \min(\text{XtNwidth}, \text{XtNheight}) / 2$

A width of zero means no border will show.

XtNcolormap

Class	Type	Default	Access
XtCColormap	Colormap	(parent’s)	SGI

Synopsis: The colormap used to interpret pixels drawn in the widget’s window.

Values: Any colormap supported by the current display and compatible with the widget’s visual resource.

Core Resources

If not initialized, Shell and DrawArea widgets use their visual resource to find (share or create) the widget's colormap.

Gadgets do not have a colormap resource. To get the colormap associated with any object use the function `OlColormapOfObject()` (see page 155).

XtNdepth

Class	Type	Default	Access
XtCDepth	int	(parent's)	GI

Synopsis: The number of bits used for each pixel in the widget's window.
 Values: Any depth supported by the current display.

The value of this resource is used to set the depth of the widget's window when the widget is created.

Gadgets do not have a depth resource. To get the depth associated with any object use the function `OlDepthOfObject()` (see page 155).

XtNdestroyCallback

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGIO

Synopsis: The callback list invoked when a widget is destroyed.

XtNheight

Class	Type	Default	Access
XtCHeight	Dimension	0	SGI

Synopsis: The height of the widget's window, in pixels, not including the border.

Values: $0 \leq XtNheight$

Programs may request a value at creation or through later calls to `XtSetValues()`, but the request may not succeed because of layout requirements of the parent widget.

When `XtNheight = 0`, the widget will select an appropriate default height; composite widgets will size themselves to fit all of their children.

XtNmappedWhenManaged

Class	Type	Default	Access
XtCMappedWhenManaged	Boolean	TRUE	SGI

Synopsis: The responsibility for mapping and managing the widget.

Values: TRUE/"true" - The widget will be mapped (made visible) as soon as it is both realized and managed.

FALSE/"false" - The program is responsible for mapping and unmapping the widget.

If the value is changed from TRUE to FALSE after the widget has been realized and managed, the widget is unmapped. The Xt `XtSetMappedWhenManaged()` function can be used to change the value of this resource.

XtNscreen

Class	Type	Default	Access
XtCScreen	Screen *	(parent's)	GI

Synopsis: The screen on which the widget appears.

Values: A pointer to an Xlib `Screen` data structure.

This resource can only be specified for Shell widgets; all other widgets appear on the same screen as their parents.

XtNsensitive

Class	Type	Default	Access
XtCSensitive	Boolean	TRUE	GIO

Synopsis: The reception of input events by a widget.

Values: TRUE/"true" - The widget will receive input events.

FALSE/"false" - The widget will not receive input events.

If both `XtNsensitive` and `XtNancestorSensitive` are TRUE, the widget will receive keyboard, mouse button, motion, window enter/leave, and focus events.

Insensitive widgets do not receive these events. Insensitive widgets that appear on the screen are stippled with a 50% gray pattern to show that they are inactive. The 50% gray pattern makes every other pixel of the widget the background color, in a checkerboard pattern.

Core Resources

An application should use the `XtSetSensitive()` function to change this resource, thereby maintaining the integrity of the `XtNancestorSensitive` resource.

Note that for `Caption` and `StaticText` widgets, if `XtNsensitive` is set to `FALSE`, the label will appear grayed out to indicate this.

XtNtranslations

Class	Type	Default	Access
XtCTranslations	XtTranslations	NULL	SGI

Synopsis: The mapping of events from the X server to widget and application functions.

Every widget that descends from the Core class has a default value for this resource. Setting this resource on a widget may completely override the default mapping of events to widget functions for the widget. Refer to Chapter 7 of the *Xt Intrinsic Programming Manual* for details on events and translations.

XtNwidth

Class	Type	Default	Access
XtCWidth	Dimension	0	SGI

Synopsis: The width of the widget’s window in pixels, not including the border.

Programs may request a value at creation or through later calls to `XtSetValues()`, but the request may not succeed because of layout requirements of the parent widget.

When `XtNwidth = 0`, the widget will select an appropriate default width; composite widgets will size themselves to fit all of their children.

*XtNx/
XtNy*

Class	Type	Default	Access
XtCPosition	Position	0	SGI

Synopsis: The position of the widget’s upper-left corner.

Composite Resources

These resources contains the x- and y-coordinates, respectively, of the widget's upper-left corner (including the border) relative to its parent widget. Programs may request a value at creation or through later calls to `XtSetValues()`, but the request may not succeed because of layout requirements of the parent widget.

For Shell widgets, `XtNx` and `XtNy` are measured relative to the root window.

Composite Resources

These are the resources of the Composite class, of which all Constraint and Manager classes are subclasses. See the diagram in "OLIT Class Hierarchy" on page 3.

Table 2-3 Composite Resources Summary

Name	Type	Default	Access
XtNchildren	WidgetList	NULL	G
XtNinsertPosition	XtOrderProc	NULL	SGI
XtNnumChildren	Cardinal	0	G

XtNchildren

Class	Type	Default	Access
XtCReadOnly	WidgetList	NULL	G

Synopsis: The list of children of the widget.

Note – This resource is intended to be used inside an insert-position procedure. It should never be set by the application.

XtNinsertPosition

Class	Type	Default	Access
XtCInsertPosition	XtOrderProc	NULL	SGI

Synopsis: The procedure that determines where a new child is to be inserted into a list of existing children.

The default procedure inserts the new child at the end of the list.

Primitive Resources

XtNnumChildren

Class	Type	Default	Access
XtCReadOnly	Cardinal	0	G

Synopsis: The number of entries in the list of children of the widget.

Note – This resource is intended to be used inside an insert-position procedure. It should never be set by the application.

Primitive Resources

The following resources are available to the widgets that are a subclass of the Primitive class. See the diagram in “OLIT Class Hierarchy” on page 3.

Table 2-4 Primitive Resources Summary

Name	Type	Default	Access
XtNaccelerator	String	NULL	SGI
XtNacceleratorText	String	NULL	SGI
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNfont	OlFont	XtDefaultFont	SGID
XtNfontColor	Pixel	XtDefaultForeground	SGID
XtNforeground	Pixel	XtDefaultForeground	SGID
XtNinputFocusColor	Pixel	Red	SGID
XtNmnemonic	unsigned char	'\0'	SGI
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNscale	int	12	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI
XtNtraversalOn	Boolean	TRUE	SGI
XtNuserData	XtPointer	NULL	SGI

XtNaccelerator

Class	Type	Default	Access
XtCAccelerator	String	NULL	SGI

Synopsis: The single `KeyPress` event that activates the widget.

Values: A subset of the Xt translation manager syntax described in the *XLib Reference Manual* can be used as the string value.

For example, given a button named “File,” the following is a valid accelerator specification:

```
*File.accelerator: Ctrl Shift<Key>f
```

The button will be activated when the user presses the `f` key while holding down both the Meta and Control keys.

Refer to “Activation Type Description” on page 61 for a more complete description of acceptable syntax.

For compatibility reasons, the previous OLIT accelerator syntax is supported. There is also a new general OPEN LOOK syntax that can be used to specify accelerators for OLIT, `xview(1)`, and `olwm(1)`.

XtNacceleratorText

Class	Type	Default	Access
XtCAcceleratorText	String	NULL	SGI

Synopsis: The string describing the associated primitive widget’s accelerator.

For example, a Help button may set the resource to the string `F1` to remind the users that function key 1 is the HELP button. This text will be displayed to the right of the Primitive’s label or image if the `XtNshowAccelerators` toolkit resource is `OL_DISPLAY`. See page 16.

As a default value, the toolkit attempts to construct a user-readable representation of the value of `XtNaccelerator`. However, some language environments might not provide suitable fonts or character encoding for an appropriate user-readable form; applications can overcome this problem by providing a value for `XtNacceleratorText`. In many language environments, this resource defaults to the `XtNaccelerator` string with “-” characters inserted between multiple key sequences.

Primitive Resources

XtNconsumeEvent

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGIO

Synopsis: The callback list invoked to consume an XEvent.

Whenever an event is processed by the standard OLIT translation table, the *XtNconsumeEvent* list is called for the widget in question, allowing the application to consume the XEvent.

The *call_data* for this resource uses data structures of type *OlVirtualEventRec*. (See “OlLookupInputEvent” on page 212 for more detail of this data structure.)

To consume an event, the application should turn on (set to TRUE) the *consumed* field in the *call_data* argument when a given event is processed.

OlAddCallback() must be used instead of *XtAddCallback()* when adding callbacks to the *XtNconsumeEvent* callback list. (It is possible to use *XtAddCallback()* for Primitive and Manager widgets, but not for VendorShell widgets; therefore, it is recommended that applications be written consistently with the *OlAddCallback()* function for all widget classes.)

XtNfont

Class	Type	Default	Access
XtCFont	OlFont	XtDefaultFont	SGID

Synopsis: The default font used by labels in a widget.

Values: Any valid XFontStruct pointer or XFontSet value.

The interpretation of this resource is dependent upon the value of the *XtNtextFormat* resource of the widget.

If the *XtNtextFormat* resource has the value *OL_SB_STR_REP*, then the *XtNfont* resources will be an *OlFont* reference to an XFontStruct *; for other values of the *XtNtextFormat* resource, the value of the *XtNfont* resource will be an *OlFont* reference to an XFontSet.

The default value for *XtNfont* is determined using the following algorithm:

Primitive Resources

- If the widget's text format is `OL_SB_STR_REP`, and if the `XtNxtDefaultFont` resource is specified, an attempt is made to load `XtNxtDefaultFont`. If the loading of `XtNxtDefaultFont` is unsuccessful, or if `XtNxtDefaultFont` is not specified, the above step is repeated with `XtNolDefaultFont`.
- If the widget's text format is not `OL_SB_STR_REP`, and if the `XtNxtDefaultFontSet` resource is specified, an attempt is made to load `XtNxtDefaultFontSet`. If the loading of `XtNxtDefaultFontSet` is unsuccessful, or if `XtNxtDefaultFontSet` is not specified, the above step is repeated with `XtNolDefaultFont`.

XtNfontColor

Class	Type	Default	Access
<code>XtCFontColor</code>	<code>Pixel</code>	<code>XtDefaultForeground</code>	<code>SGID</code>

Synopsis: The font color used by objects that display text.

Values: Any `Pixel` value valid for the current display, or any name from the `$OPENWINHOME/lib/rgb.txt` file. (Pixel values are used in C programs, `rgb.txt` values in X resource files. See "Resource Files" on page 7.)

XtNforeground

Class	Type	Default	Access
<code>XtCForeground</code>	<code>Pixel</code>	<code>XtDefaultForeground</code>	<code>SGID</code>

Synopsis: The foreground color used by objects to draw non-textual content, provided that the value of the `XtNinputFocusFeedback` toolkit resource (see page 11) is `OL_INPUT_FOCUS_COLOR`.

Values: Any `Pixel` value valid for the current display, or any name from the `$OPENWINHOME/lib/rgb.txt` file. (Pixel values are used in C programs, `rgb.txt` values in X resource files. See "Resource Files" on page 7.)

XtNinputFocusColor

Class	Type	Default	Access
<code>XtCInputFocusColor</code>	<code>Pixel</code>	<code>Red</code>	<code>SGID</code>

Synopsis: The color used to show that the widget has input focus.

Primitive Resources

Values: Any `Pixel` value valid for the current display, or any name from the `$OPENWINHOME/lib/rgb.txt` file. (Pixel values are used in C programs, `rgb.txt` values in X resource files. See “Resource Files” on page 7.)

Normally, the color used to show input focus is derived from the value of the `XtNinputFocusColor` resource and is dynamically maintained. This dynamic behavior is abandoned if the application explicitly sets this resource either at initialization or through a call to `XtSetValues()`.

For various widgets, the default is dependent on the value of other resources. For the `FileChooser`, `FontChooser`, `TextEdit`, `TextField`, and `TextLine` widgets, if the application resource `XtNmouseless = TRUE` and the application resource `XtNinputFocusFeedback = OL_INPUT_FOCUS_COLOR` (see page 11), `XtNinputFocusColor` defaults to “Red”; otherwise, it defaults to the value of `XtNfontColor`.

XtNmnemonic

Class	Type	Default	Access
<code>XtCMnemonic</code>	unsigned char	'\0'	SGI

Synopsis: The mnemonic for keyboard operation.

Values: Any single-byte displayable character that is in the associated widget’s label, or a character capable of being displayed in the widget’s label.

Typing this character modified with the `XtNmnemonicPrefix` is equivalent to activating the widget with the `OL_SELECT` activation type.

XtNreferenceName

Class	Type	Default	Access
<code>XtCReferenceName</code>	String	NULL	GI

Synopsis: The position for inserting this widget in the traversal list of its closest shell ancestor.

Values: The name of a widget already created as a descendant of its closest shell ancestor.

If the named widget exists in the managing ancestor’s traversal list, this widget will be inserted in front of it. Otherwise, this widget will be inserted at the end of the list.

Primitive Resources

If both the `XtNreferenceName` and `XtNreferenceWidget` resources are set, they must refer to the same widget. If not, a warning is issued and the widget referred to by name is used.

XtNreferenceWidget

Class	Type	Default	Access
<code>XtCReferenceWidget</code>	Widget	NULL	GI

Synopsis: The position for inserting this widget in the traversal list of its closest shell ancestor.

Values: The widget ID of a widget already created as a descendant of its closest shell ancestor.

If the referenced widget is non-null and exists in the managing ancestor's traversal list, this widget will be inserted in front of it. Otherwise, this widget will be inserted at the end of the list.

If both the `XtNreferenceName` and `XtNreferenceWidget` resources are set, they must refer to the same widget. If not, a warning is issued and the widget referred to by name is used.

XtNscale

Class	Type	Default	Access
<code>XtCScale</code>	int	12	SGI

Synopsis: The size of graphical elements (widgets), proportioned to the size of text, measured in points (1/72 inch).

The toolkit supports sizes of 10, 12, 14, 16, 19, 20, and 24 points; other values may not display correctly.

XtNtextFormat

Class	Type	Default	Access
<code>XtCTextFormat</code>	OlStrRep	OL_SB_STR_REP	GI

Synopsis: The expected data format of all the textual resources of a widget.

Values:

- OL_SB_STR_REP - Single-byte character representation.
- OL_WC_STR_REP - Wide character representation.
- OL_MB_STR_REP - Multibyte character representation.

Primitive Resources

`XtNtextFormat` can only be set when the widget is created. This can be achieved by passing it as an argument to the Xt function used to create the widget, for example `XtVaCreateManagedWidget()`. Alternatively, the toolkit has a built-in default value for this resource, which applications can change using the function `OlSetDefaultTextFormat()`. Unless changed by the application, the default is `OL_SB_STR_REP`. `XtNtextFormat` cannot be set with `XtSetValues()`.

The widget subsequently manipulates and renders all data specified by the application for its textual resources, assuming the specified data format. For instance, if `XtNtextFormat` is set to `OL_MB_STR_REP`, the widget might render a label using the Xlib function `XmbDrawString()`. If `XtNtextFormat` were `OL_WC_STR_REP`, the widget would use `XwcDrawString()`.

XtNtraversalOn

Class	Type	Default	Access
<code>XtCTraversalOn</code>	Boolean	TRUE	SGI

Synopsis: The accessibility of this widget through keyboard traversal.

Values: TRUE/"true" - The widget is accessible.
 FALSE/"false" - The widget is not accessible.

Note - This resource affects only an individual widget, and in the case of Manager widgets, their children. Setting `*traversalOn: false` in a resource control file is not quite equivalent to turning off mouseless operation.

XtNuserData

Class	Type	Default	Access
<code>XtCUserData</code>	XtPointer	NULL	SGI

Synopsis: Storage for application-specific data.

The toolkit does not modify the value in the storage area pointed to by `XtNuserData`. The application is responsible for allocating and freeing this area.

Manager Resources

The following resources are available to the widgets that are a subclass of the Manager class. See the diagram in “OLIT Class Hierarchy” on page 3.

Table 2-5 Manager Resources Summary

Name	Type	Default	Access
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNinputFocusColor	Pixel	Red	SGID
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNtraversalOn	Boolean	TRUE	SGI
XtNuserData	XtPointer	NULL	SGI

All of these Manager resources are equivalent to those defined in the Primitive class; see:

- “XtNconsumeEvent” on page 26,
- “XtNinputFocusColor” on page 27,
- “XtNreferenceName” on page 28,
- “XtNreferenceWidget” on page 29,
- “XtNtraversalOn” on page 30,
- “XtNuserData” on page 30.

Shell Resources

These are resources that are common to all widget classes that are subclasses of Shell. See the diagram in “OLIT Class Hierarchy” on page 3.

Base Windows and Popup Windows

To create OPEN LOOK base windows use `OlToolkitInitialize()` or `XtCreateApplicationShell()`. (`FileChooserShell`, `FontChooserShell`, `PopupWindowShell`, `MenuShell`, and `NoticeShell` widgets are created using `XtCreateApplicationShell()`.) An application can define other popup windows that can be created using `XtCreatePopupShell()`. The following resources are typical of base windows and generic popup windows, but not all are available for the popup windows defined in this toolkit. See the list of

Shell Resources

resources for the `PopupWindow`, `Menu`, and `Notice` widgets to see which are available. The “Access” column in this table identifies the access for base windows only.

Table 2-6 Shell Resources Summary

Name	Type	Default	Access
<code>XtNallowShellResize</code>	Boolean	TRUE	SGI
<code>XtNcreatePopupChildProc</code>	<code>XtCreatePopupChildProc</code>	NULL	SGI
<code>XtNgeometry</code>	String	NULL	GI
<code>XtNoverrideRedirect</code>	Boolean	FALSE	SGI
<code>XtNpopdownCallback</code>	<code>XtCallbackList</code>	NULL	SGIO
<code>XtNpopupCallback</code>	<code>XtCallbackList</code>	NULL	SGIO
<code>XtNsaveUnder</code>	Boolean	FALSE	SGI
<code>XtNvisual</code>	Visual *	(parent's)	GIO
<code>XtNwidthInc</code>	int	<code>XtUnspecifiedShellInt</code>	SGI

XtNallowShellResize

Class	Type	Default	Access
<code>XtCallowShellResize</code>	Boolean	TRUE	SGI

Synopsis: The resize results of a child's geometry request.

Values: TRUE – The widget will attempt to resize itself as requested by the child. The attempt may be refused by the window manager, which will cause the shell widget to refuse the geometry management request of its child. Otherwise, it accepts the request.
 FALSE – A Shell widget will immediately refuse the geometry management request.

XtNcreatePopupChildProc

Class	Type	Default	Access
<code>XtCCreatePopupChildProc</code>	<code>XtCreatePopupChildProc</code>	TRUE	SGI

Synopsis: The single function (not a callback list) called during popup.

The function indicated by this resource is called after the `XtNpopupCallback` callbacks are issued (see page 33), but before the shell widget is realized and mapped. The function is passed a single argument, the ID of the shell widget.

XtNgeometry

Class	Type	Default	Access
XtCGeometry	String	NULL	GI

Synopsis: The size and position of the shell widget when it pops up.

Values: Any syntactically correct argument to the `XParseGeometry()` function (see the *XLib Reference Manual*).

XtNoverrideRedirect

Class	Type	Default	Access
XtCOverrideRedirect	Boolean	FALSE	SGI

Synopsis: The manager of a shell widget's window.

Values: `TRUE/"true"` - The window manager does *not* manage the shell widget's window.
`FALSE/"false"` - The window manager manages the shell widget's window.

Do not set this resource for any of the OLIT shell widgets: `FileChooserShell`, `FontChooserShell`, `MenuShell`, `NoticeShell`, or `PopupWindowShell`.

XtNpopdownCallback

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGIO

Synopsis: The callback list invoked just after the shell widget's window pops down.

XtNpopupCallback

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGIO

Synopsis: The callback list invoked just before the shell widget's window pops up.

Shell Resources

XtNsaveUnder

Class	Type	Default	Access
XtCSaveUnder	Boolean	FALSE	SGI

Synopsis: Whether the shell widget should instruct the server to attempt to save the contents of windows obscured by the shell when it is mapped, and to restore the contents when the shell widget is unmapped.

Values: TRUE/"true" - The server will attempt to save and restore the contents.
 FALSE/"false" - The server will not attempt to save and restore the contents.

XtNvisual

Class	Type	Default	Access
XtCVisual	Visual *	(parent's)	GIO

Synopsis: The visual used to create the widget's window.

Values: A pointer to any visual structure supported by the current display and compatible with the widget's depth and colormap.

Only Shell and DrawArea widgets have a visual resource. All other widgets are created using their parent's visual.

If not initialized, Shell and DrawArea widgets use their depth resource and parent's visual class to find the widget's visual.

The preferred method of setting a Shell or DrawArea widget's visual resource is to use the Intrinsics *typed args* interface. A string containing the desired Visual Class Name should be passed to the String to Visual resource converter.

To get the visual associated with any object use the function `OlVisualOfObject()` (see page 156).

WMShell Resources

These are resources defined in `WMShellWidgetClass`. See the diagram in “OLIT Class Hierarchy” on page 3.

Table 2-7 WMShell Resources Summary

Name	Type	Default	Access
XtNbaseHeight	int	XtUnspecifiedShellInt	SGI
XtNbaseWidth	int	XtUnspecifiedShellInt	SGI
XtNheightInc	int	XtUnspecifiedShellInt	SGI
XtNiconMask	Pixmap	NULL	SGI
XtNiconPixmap	Pixmap	NULL	SGI
XtNiconWindow	Window	NULL	SGI
XtNiconX	int	XtUnspecifiedShellInt	SGI
XtNiconY	int	XtUnspecifiedShellInt	SGI
XtNinitialState	InitialState	NormalState	SGI
XtNinput	Bool	FALSE	G
XtNmaxAspectX	int	XtUnspecifiedShellInt	SGI
XtNmaxAspectY	int	XtUnspecifiedShellInt	SGI
XtNmaxHeight	int	OL_IGNORE	SGI
XtNmaxWidth	int	OL_IGNORE	SGI
XtNminAspectX	int	XtUnspecifiedShellInt	SGI
XtNminAspectY	int	XtUnspecifiedShellInt	SGI
XtNminHeight	int	OL_IGNORE	SGI
XtNminWidth	int	OL_IGNORE	SGI
XtNtitle	String	NULL	SGI
XtNtitleEncoding	Atom	XA_STRING	SGI
XtNtransient	Boolean	TRUE	SGI
XtNwaitForWm	Boolean	TRUE	SGI
XtNwidthInc	int	XtUnspecifiedShellInt	SGI
XtNwindowGroup	Window	XtUnspecifiedWindow	SGI
XtNwinGravity	int	XtUnspecifiedShellInt	SGI
XtNwmTimeout	int	5000 (msec)	SGI

WMShell Resources

*XtNbaseHeight/
XtNbaseWidth*

Class	Type	Default	Access
XtCBaseHeight	Int	XtUnspecifiedShellInt	SGI
XtCBaseWidth	Int	XtUnspecifiedShellInt	SGI

Synopsis: The base values to which the `XtNheightInc` and `XtNwidthInc` size increments are added.

XtNheightInc

Class	Type	Default	Access
XtCHeightInc	int	XtUnspecifiedShellInt	SGI

Synopsis: The resizing increment for Shell widgets.

Values: $0 \leq XtNheightInc$

This resource defines an arithmetic progression of sizes, from `XtNminHeight` to `XtNmaxHeight` into which the shell widget prefers to be resized by the window manager.

XtNiconMask

Class	Type	Default	Access
XtCIconMask	Pixmap	NULL	SGI

Synopsis: The mask applied to `XtNiconPixmap` to give the base window's icon.

Values: A single plane pixmap.

XtNiconPixmap

Class	Type	Default	Access
XtCIconPixmap	Pixmap	NULL	SGI

Synopsis: The image to be used as the base window's icon.

Values: A single plane pixmap.

XtNiconWindow

Class	Type	Default	Access
XtCIconWindow	Window	NULL	SGI

Synopsis: The ID of a window to be used as the base window's icon.

Values: An ID of an existing window.

The `XtNiconWindow` takes precedence over the `XtNiconPixmap` resource.

XtNiconX/ XtNiconY

Class	Type	Default	Access
XtCIconX	int	XtUnspecifiedShellInt	SGI
XtCIconY	int	XtUnspecifiedShellInt	SGI

Synopsis: The x- and y-coordinates of where the base window's icon should appear.

Values: $-1 \leq \text{XtNiconX}$
 $-1 \leq \text{XtNiconY}$

If the value of either of these resource is `-1`, the window manager automatically picks a value, according to its icon placement requirements.

XtNInitialState

Class	Type	Default	Access
XtCInitialState	InitialState	NormalState	SGI

Synopsis: The appearance of the base window (and associated popup windows) when the application starts up.

Values: `NormalState` - The application starts up with its base window open.
`IconicState` - The application starts up with its base window closed into an icon.

Other values are defined by the X Window System for this resource, but the OPEN LOOK window manager recognizes only the iconic and normal states.

WMShell Resources

XtNinput

Class	Type	Default	Access
XtCInput	Boolean	FALSE	G

Synopsis: The application's input focus behavior.

This resource should not be set by an application.

*XtNmaxAspectX/
XtNmaxAspectY/
XtNminAspectX/
XtNminAspectY*

Class	Type	Default	Access
XtCMaxAspectX	int	XtUnspecifiedShellInt	SGI
XtCMaxAspectY	int	XtUnspecifiedShellInt	SGI
XtCMinAspectX	int	XtUnspecifiedShellInt	SGI
XtCMinAspectY	int	XtUnspecifiedShellInt	SGI

Synopsis: The range of aspect ratios allowed for the size of the shell widget's window.

Values: -1 = XtNmaxAspectX or $1 \leq XtNmaxAspectX$
 -1 = XtNmaxAspectY or $1 \leq XtNmaxAspectY$
 -1 = XtNminAspectX or $1 \leq XtNminAspectX$
 -1 = XtNminAspectY or $1 \leq XtNminAspectY$

$$\frac{XtNminAspectX}{XtNminAspectY} \leq \frac{XtNmaxAspectX}{XtNmaxAspectY}$$

Assuming the width and height of the window are given by *width* and *height* the following relation shows how the window size is constrained:

$$\frac{XtNminAspectX}{XtNminAspectY} \leq \frac{width}{height} \leq \frac{XtNmaxAspectX}{XtNmaxAspectY}$$

If the user tries to resize the window to a narrower or wider aspect ratio than allowed by these resources, the window manager adjusts the window to the closest allowed aspect ratio. If possible, it will do this by increasing the width or height to compensate.

WMShell Resources

The `XtNmaxHeight` and `XtNmaxWidth` resources may force the window manager to reduce the width or height instead. If the values of these resources are `-1`, the window manager does not constrain the size of the window to any aspect ratio.

Note – An application should either set all values to `-1` (the default) or should set all to a positive value. An application should never set a value of zero to any of these resources.

***XtNmaxHeight/
XtNmaxWidth/
XtNminHeight/
XtNminWidth***

Class	Type	Default	Access
<code>XtCMaxHeight</code>	int	OL_IGNORE	SGI
<code>XtCMaxWidth</code>	int	OL_IGNORE	SGI
<code>XtCMinHeight</code>	int	OL_IGNORE	SGI
<code>XtCMinWidth</code>	int	OL_IGNORE	SGI

Synopsis: The range allowed for the size of the shell widget's window.

Values: $XtNminHeight \leq XtNmaxHeight$
 $XtNminWidth \leq XtNmaxWidth$
 (or OL_IGNORE for any of these resources)

If the user tries to resize the window smaller or larger than these values allow, the window manager adjusts the width and/or height to compensate.

The default value of OL_IGNORE keeps the window manager from constraining the window's size.

XtNtitle

Class	Type	Default	Access
<code>XtCTitle</code>	String	NULL	SGI

Synopsis: The title to include in the header of the base or popup window.

Widgets of classes other than Shell may have a resource with this name.

WMShell Resources

XtNtitleEncoding

Class	Type	Default	Access
XtCTitleEncoding	Atom	XA_STRING	SGI

Synopsis: The character set used in the `XtNtitle` resource.

Values: ICCCM defines only one valid value for this resource: XA_STRING. Individual window managers may specify other values.

XtNtransient

Class	Type	Default	Access
XtCTransient	Boolean	TRUE	SGI

Synopsis: The unmapping of a shell widget's window when the associated base window is iconified.

Values: TRUE/"true" - The window is unmapped.
FALSE/"false" - The window is mapped.

A transient window is one that is unmapped when its associated base window is iconified. This resource controls this behavior.

No application should set this resource for any of the OLIT shell widgets. See `XtNwindowGroup` on page 41.

XtNwaitForWm

Class	Type	Default	Access
XtCWaitForWm	Boolean	TRUE	SGI

Synopsis: Whether the shell's geometry manager waits for the window manager to respond to a request. For details on this resource, please refer to the *Xt Intrinsic Programming Manual*.

XtNwidthInc

Class	Type	Default	Access
XtCWidthInc	int	XtUnspecifiedShellInt	SGI

Synopsis: The resizing increment for Shell widgets.

Values: $0 \leq \text{XtNwidthInc}$

WMShell Resources

This resource defines the increment in an arithmetic progression of sizes, from `XtNminWidth` to `XtNmaxWidth`, into which the shell widget prefers to be resized by the window manager.

XtNwindowGroup

Class	Type	Default	Access
<code>XtCWindowGroup</code>	Window	<code>XtUnspecifiedWindow</code>	SGI

Synopsis: The base window associated with this shell widget's window.

Values: An ID of an existing window

When the user closes the base window, all its associated windows are unmapped (popup windows or other shell widget windows with `XtNtransient` set to TRUE) or closed (base windows with `XtNtransient` set to FALSE).

XtNwinGravity

Class	Type	Default	Access
<code>XtCWinGravity</code>	int	(see description)	SGI

Synopsis: The corner of the application window critical for placement.

Values: `WestGravity/"west"`
`CenterGravity/"center"`
`NorthGravity/"north"`
`NorthEastGravity/"northEast"`
`NorthWestGravity/"northWest"`
`SouthGravity/"south"`
`SouthEastGravity/"southEast"`
`SouthWestGravity/"southWest"`

If `XtNgeometry` is NULL, the default value is `NorthWestGravity`. If `XtNgeometry` is not NULL, the default value is the gravity implied by the geometry string. Consult the *Xt Intrinsic Reference Manual* for further details on `XtNwinGravity`.

VendorShell Resources

XtNwmTimeout

Class	Type	Default	Access
XtCWmTimeout	int	5000 (msec)	SGI

Synopsis: The time interval the shell’s geometry manager waits for the window manager to respond to a request. See “XtNwaitForWm” on page 40.

VendorShell Resources

Table 2-6 on page 32 listed generic resources available to most shells. Table 2-8, however, lists resources necessary to support the OPEN LOOK look and feel. These resources are implemented in the `VendorShell` widget class; and therefore, apply only to shells that are subclasses of the `VendorShell` widget class (i.e., `TransientShell`, `MenuShell`, `PopupWindowShell`, `NoticeShell`, `FontChooserShell`, `TopLevelShell`, and `ApplicationShell`). Since some of these resources do not apply to all shells (e.g., `XtNresizeCorners` on menus), see the individual widget descriptions for more accurate descriptions of the applicable resources and their default values.

Table 2-8 VendorShell Resources Summary

Name	Type	Default	Access
XtNbusy	Boolean	FALSE	SGI
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNdefaultImName	String	NULL	SGI
XtNfooterPresent	Boolean	FALSE	SGI
XtNfocusWidget	Widget	(see description)	SGI
XtNimFontSet	OIFont	XtDefaultFontSet	SGI
XtNimStatusStyle	OImStatusStyle	OL_NO_STATUS	GI
XtNleftFooterString	OIStr	NULL	SGI
XtNleftFooterVisible	Boolean	TRUE	SGI
XtNmenuButton	Boolean	(see description)	GI
XtNmenuType	OIDefine	(see description)	SGI
XtNpushpin	OIDefine	(see description)	SGI
XtNresizeCorners	Boolean	(see description)	SGI
XtNrightFooterString	OIStr	NULL	SGI
XtNrightFooterVisible	Boolean	TRUE	SGI

Table 2-8 VendorShell Resources Summary (Continued)

Name	Type	Default	Access
XtNshellTitle	OIStr	NULL	SGI
XtNuserData	XtPointer	NULL	SGI
XtNwindowHeader	Boolean	(see description)	GI
XtNwmProtocol	XtCallbackList	NULL	SGIO
XtNwmProtocolInterested	int	OL_WM_DELETE_WINDOW OL_WM_TAKE_FOCUS	I

XtNbusy

Class	Type	Default	Access
XtCBusy	Boolean	FALSE	SGI

Synopsis: The marking of the shell's window as busy.

Values: TRUE – Marks as busy the application window associated with this shell. When a window becomes busy, the window manager grays the window header (if there is one).
 FALSE – Causes the window to return to its normal appearance and event processing. Neither the window manager nor the toolkit grabs mouse or keyboard events when an application window becomes busy.

XtNconsumeEvent

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGIO

Synopsis: The callback list invoked to consume an XEvent. This resource is equivalent to the one defined for the Primitive class; see “XtNconsumeEvent” on page 26.

XtNdefaultImName

Class	Type	Default	Access
XtCDefaultImName	String	NULL	SGI

Synopsis: The name of the default input method.

Values: Name of an input method suitable for the locale of the application. See “Input Method” on page 80.

VendorShell Resources

XtNfocusWidget

Class	Type	Default	Access
XtCFocusWidget	Widget	(see description)	SGI

Synopsis: The widget that gets input focus when the user selects a window.

If not initialized by the programmer, this resource defaults to the first widget created among its descendants capable of accepting input focus.

As focus changes within the shell, this resource is updated to reflect the widget with focus. Focus will be set to this widget when the VendorShell loses and then regains focus.

A resource converter will translate widget names specified in a resource file to a widget ID for this resource.

XtNfooterPresent

Class	Type	Default	Access
XtCFooterPresent	Boolean	FALSE	SGI

Synopsis: The presentation of a shell footer area.

Values: TRUE/"true" - The footer area is created and/or mapped.
 FALSE/"false" - If the footer area already exists, it is unmapped; otherwise, nothing is created.

XtNimFontSet

Class	Type	Default	Access
XtCImFontSet	OlFont	XtDefaultFontSet	SGI

Synopsis: The font set used by the input method to display status feedback.

Values: Any fontset suitable for the locale of the application.

XtNimStatusStyle

Class	Type	Default	Access
XtCImStatusStyle	OImStatusStyle	OL_NO_STATUS	GI

Synopsis: The location of the input method status feedback. See "Setting the Input Method Pre-Edit and Status Styles (Asian Locales Only)" on page 82.

VendorShell Resources

Values: OL_IM_DISPLAYS_IN_CLIENT/"imDisplaysInClient" - The input method displays the status in the footer of the shell's window.
 OL_IM_DISPLAYS_IN_ROOT/"imDisplaysInRoot" - The input method displays the status in a separate window that is a child of the root window.
 OL_NO_STATUS/"none" - The input method provides no status feedback.

XtNleftFooterString

Class	Type	Default	Access
XtCLeftFooterString	OlStr	NULL	SGI

Synopsis: The left footer string.

Values: Any OlStr value valid in the current locale.

Both XtNfooterPresent and XtNleftFooterVisible must be TRUE for this string to be visible.

XtNleftFooterVisible

Class	Type	Default	Access
XtCLeftFooterVisible	Boolean	TRUE	SGI

Synopsis: The visibility of the left footer area.

Values: TRUE/"true" - The left footer area is made visible.

FALSE/"false" - The left footer area is made invisible.

If the XtNfooterPresent resource is FALSE, the XtNleftFooterVisible resource has no effect.

XtNmenuButton

Class	Type	Default	Access
XtCMenuButton	Boolean	(see description)	GI

Synopsis: The placement of the menu button decoration in the upper left corner of the shell window's header.

Values: TRUE/"true" - The menu button decoration should be drawn. This is the default for TopLevel shells.

FALSE/"false" - The menu button decoration should not be drawn. This is the default for Transient shells.

XtNmenuType

Class	Type	Default	Access
XtCMenuType	OlDefine	(see description)	SGI

Synopsis: The type of window menu that the window manager creates.

Values: `OL_MENU_FULL/"full"` - This is the default value for a base shell. This full menu contains the following entries: Close, Full Size, Move and Resize, Properties, Back, Refresh, and Quit.
`OL_MENU_LIMITED/"limited"` - Setting this value results in a window menu with the following buttons: Dismiss (a MenuButton) Move and Resize, Back, Refresh, Owner?. `OL_MENU_LIMITED` is the default for `PopupWindow` and `Help` shells.
`OL_MENU_CANCEL/"cancel"` - This value provides the same menu as the `OL_MENU_LIMITED` with the exception that the Dismiss button is replaced with a Cancel button.
`OL_NONE/"none"` - The window manager does not create a menu or a menu mark. The `NoticeShell` widget sets this value.

XtNpushpin

Class	Type	Default	Access
XtCPushPin	OlDefine	(see description)	SGI

Synopsis: The inclusion of the pushpin in the window's decorations.

Values: `OL_NONE/"none"` - A pushpin is not included in the window's decorations. This is the default for base window shells and `NoticeShells`.
`OL_OUT/"out"` - A pushpin is included in the window's decorations, with its state set to be unpinned. This is the default for `PopupWindowShells`.
`OL_IN/"in"` - A pushpin is included in the window's decorations, with its state set to be pinned. A `MenuShell` widget should not set `XtNpushpin` to `OL_IN` at initialization time.

Applications can query the state of the pushpin by getting the value of this resource, since it is updated when the pushpin's state changes.

If the shell does not have an OPEN LOOK header (`XtNwindowHeader` is set to `FALSE`), then `XtNpushpin` is always `OL_NONE`, and attempts to change the value are ignored.

Once created, a widget supports transitions of out-to-in and in-to-out, but other transitions are implementation dependent.

XtNresizeCorners

Class	Type	Default	Access
XtCResizeCorners	Boolean	(see description)	SGI

Synopsis: The inclusion of resize corners as part of the window decorations.
 Values: TRUE – Default for the base shell; resize corners are present.
 FALSE – Default for other Shell widgets; resize corners are not present.

XtNrightFooterString

Class	Type	Default	Access
XtCRightFooterString	OlStr	NULL	SGI

Synopsis: The right footer string.
 Values: Any OlStr value valid in the current locale.

Both `XtNfooterPresent` and `XtNrightFooterVisible` must be TRUE for this string to be visible.

XtNrightFooterVisible

Class	Type	Default	Access
XtCRightFooterVisible	Boolean	TRUE	SGI

Synopsis: The visibility of the right footer area.
 Values: TRUE/"true" – The right footer area is visible.
 FALSE/"false" – The right footer area is not visible.

If the `XtNfooterPresent` resource is FALSE, the `XtNrightFooterVisible` resource has no effect.

XtNshellTitle

Class	Type	Default	Access
XtCTitle	OlStr	NULL	SGI

Synopsis: The title for a shell widget.
 Values: Any OlStr value valid in the current locale.

VendorShell Resources

The value of this resource is internally kept consistent with the value of the `XtNtitle` resource. Changing either of the two resources affects the other. The essential difference between `XtNtitle` and `XtNshellTitle` lies in their types.

XtNuserData

Class	Type	Default	Access
<code>XtCUserData</code>	<code>XtPointer</code>	NULL	SGI

Synopsis: Storage for application-specific data. This resource is equivalent to the one defined for the Primitive class; see “`XtNuserData`” on page 30.

XtNwindowHeader

Class	Type	Default	Access
<code>XtCWindowHeader</code>	Boolean	(see description)	GI

Synopsis: The presence of a window header provided by the window manager.

Values: `TRUE/”true”` - Default for base windows, `PopupWindows`, and `Help` shells, indicating the window has a header.
`FALSE/”false”` - Default for `Notice` shell, indicating the window has no header.

The header is the area of the window that contains the pushpin, title, and window mark.

Note - This resource can only be set at initialization.

XtNwmProtocol

Class	Type	Default	Access
<code>XtCWmProtocol</code>	<code>XtCallbackList</code>	NULL	SGIO

Synopsis: The callback list invoked when a vendor shell widget receives `WM_PROTOCOL` messages.

This resource controls the action that is taken whenever a shell widget receives `WM_PROTOCOL` messages matching the types of protocol messages specified in the `XtNwmProtocolInterested` resource. If no callback list is specified, the

VendorShell Resources

shell performs its default action(s). If a callback list is specified, it is invoked and no default action(s) is taken. The application can, however, simulate the default action(s) at its convenience by calling `OlWMProtocolAction()` with the `action` parameter set to `OL_DEFAULTACTION`. (See “Protocol Function” on page 160 for more information on this routine.)

When the application’s callback list is invoked, the `call_data` field is a pointer to the following structure:

```
typedef struct {
    int      msgtype; /* type of WM msg */
    XEvent   *xevent;
} OlWMProtocolVerify;
```

The field `msgtype` is an integer constant indicating the type of protocol message that invoked the callback and has a range of values of:

```
OL_WM_TAKE_FOCUS
OL_WM_SAVE_YOURSELF
OL_WM_DELETE_WINDOW
```

`OlAddCallback()` must be used instead of `XtAddCallback()` when adding callbacks to the `XtNwmProtocol` callback list.

XtNwmProtocolInterested

Class	Type	Default	Access
XtCWMProtocolInterested	int	OL_WM_DELETE_WINDOW OL_WM_TAKE_FOCUS	SIG

Synopsis: The types of protocol messages that interest the application.

Values: The bitwise inclusive OR of the following values:

`OL_WM_DELETE_WINDOW` - Requests any protocol messages associated with `WM_DELETE_WINDOW`. If this value is not set by the application, undefined results occur.

`OL_WM_TAKE_FOCUS` - Requests any protocol messages associated with `WM_TAKE_FOCUS`. If this value is not set by the application, undefined results occur.

`OL_WM_SAVE_YOURSELF` - Requests any protocol messages associated with `WM_SAVE_YOURSELF`.

TransientShell Resources

TransientShell Resources

This resource is defined in `TransientShellWidgetClass`. See the diagram in “OLIT Class Hierarchy” on page 3.

Table 2-9 TransientShell Resources Summary

Name	Type	Default	Access
XtNtransientFor	Widget	NULL	SGI

XtNtransientFor

Class	Type	Default	Access
XtCTransientFor	Widget	NULL	SGI

Synopsis: The widget the shell is a transient for if the shell has the `XtNtransient` resource TRUE and is a transient shell.

Values: The widget the shell is transient for.

TopLevelShell Resources

These are resources defined in `TopLevelShellWidgetClass`. See the diagram in “OLIT Class Hierarchy” on page 3.

Table 2-10 TopLevelShell Resources Summary

Name	Type	Default	Access
XtNiconic	Boolean	FALSE	SGI
XtNiconName	String	NULL	SGI
XtNiconNameEncoding	Atom	XA_STRING	SGI

XtNiconic

Class	Type	Default	Access
XtCIconic	Boolean	FALSE	SGI

Synopsis: The iconic state of the base window.

Values: TRUE/"true" - Iconifies the base window.

FALSE/"false" - De-iconifies the base window.

This resource also provides an alternative way to set the `XtNInitialState` resource to `IconicState`.

XtNiconName

Class	Type	Default	Access
XtCIconName	String	NULL	SGI

Synopsis: The name that the window manager displays in the shell widget's icon.

If the `XtNtitle` resource is not defined or is NULL, this resource is used instead. If this resource is NULL, the name of the application is used in its place.

XtNiconNameEncoding

Class	Type	Default	Access
XtCIconNameEncoding	Atom	XA_STRING	SGI

Synopsis: The character set used in the `XtNiconName` resource.

Values: ICCCM defines only one valid value for this resource: `XA_STRING`. Individual window managers may specify other values.

ApplicationShell Resources

These are resources defined in `ApplicationShellWidgetClass`. See the diagram in “OLIT Class Hierarchy” on page 3.

Table 2-11 ApplicationShell Resources Summary

Name	Type	Default	Access
XtNargc	int	0	I
XtNargv	String *	NULL	I

*XtNargc/
XtNargv*

Class	Type	Default	Access
XtCargc	int	0	I
XtCargv	String *	NULL	I

Synopsis: The setting for the `WM_COMMAND` property.

Flat Resources

The application shell uses `XtNargc` and `XtNargv` to set the `WM_COMMAND` property. The `WM_COMMAND` property specifies the command line used to invoke the program. If an application uses `XtAppInitialize()`, the intrinsics set `XtNargc` and `XtNargv` to the values of `argc` and `argv` passed to `XtAppInitialize()`.

Flat Resources

Flat widgets are described starting on page 321. All of the flat containers have the same layout characteristics. The superclass of all flat widgets is a generic row/column manager metaclass called Flat. Although each column has its own width and each row has its own height, all columns can have the same width and all rows can have the same height, if desired. The efficiency in both processing steps and data requirements increases as the grid becomes more regular in shape. For example, a grid specifying that all rows must have the same height and all columns must have the same width is the most efficient configuration. The Flat row/column manager widget lays out the items within the container, driven by the layout attributes of the container and starting in the NorthWest corner. Row-major order implies every column in the current row is filled before filling any columns in the next row. Column-major order implies every row in the current column is filled before filling any rows in the next column.

Items of flat containers are placed within the grid. If the item's width (or height) is less than the column's width (or row's height), the item is positioned in accordance to the `XtNitemGravity` resource. The following table lists the layout resources of all flat containers. See the resource tables for each flat container widget for a more accurate accounting of the default and allowable values for each layout resource.

Table 2-12 Flat Resources Summary

Name	Type	Default	Access
<code>XtNgravity</code>	int	CenterGravity	SGI
<code>XtNhPad</code>	Dimension	0	SGI
<code>XtNhSpace</code>	Dimension	0	SGI
<code>XtNitemFields</code>	String *	NULL	SGI
<code>XtNitemGravity</code>	int	NorthWestGravity	SGI
<code>XtNitemMaxHeight</code>	Dimension	OL_IGNORE	SGI
<code>XtNitemMaxWidth</code>	Dimension	OL_IGNORE	SGI

Table 2-12 Flat Resources Summary (Continued)

Name	Type	Default	Access
XtNitemMinHeight	Dimension	OL_IGNORE	SGI
XtNitemMinWidth	Dimension	OL_IGNORE	SGI
XtNitems	XtPointer	NULL	SGI
XtNitemsTouched	Boolean	FALSE	SG
XtNlabel	OlStr	NULL	SGI
XtNlabelImage	XImage *	NULL	SGI
XtNlabelJustify	OlDefine	OL_LEFT	SGI
XtNlabelTile	Boolean	FALSE	SGI
XtNlayoutHeight	OlDefine	OL_MINIMIZE	SGI
XtNlayoutType	OlDefine	OL_FIXEDROWS	SGI
XtNlayoutWidth	OlDefine	OL_MINIMIZE	SGI
XtNmeasure	int	1	SGI
XtNnumItemFields	Cardinal	0	SGI
XtNnumItems	Cardinal	0	SGI
XtNsameHeight	OlDefine	OL_ALL	SGI
XtNsameWidth	OlDefine	OL_COLUMNS	SGI
XtNvPad	Dimension	0	SGI
XtNvSpace	Dimension	4	SGI

XtNgravity

Class	Type	Default	Access
XtCCgravity	int	CenterGravity	SGI

Synopsis: The locus in the container where an undersized group of items will be placed during layout.

Values: WestGravity/"west"
CenterGravity/"center"
NorthGravity/"north"
NorthEastGravity/"northEast"
NorthWestGravity/"northWest"
SouthGravity/"south"
SouthEastGravity/"southEast"
SouthWestGravity/"southWest"

The gravity resource specifies the position of all items (i.e., as a group) whenever a tight-fitting bounding box that surrounds the items has a width, or

Flat Resources

height, less than the container's width or height, respectively. Essentially, this resource specifies how the items, as a group, float within their container.

*XtNhPad/
XtNvPad*

Class	Type	Default	Access
XtCHPad	Dimension	0	SGI
XtCVPad	Dimension	0	SGI

Synopsis: The minimum horizontal and vertical space to leave around the edges of the container.

Values: $0 \leq XtNhPad$
 $0 \leq XtNvPad$

*XtNhSpace/
XtNvSpace*

Class	Type	Default	Access
XtCHSpace	Dimension	0	SGI
XtCVSpace	Dimension	4	SGI

Synopsis: The amount of horizontal and vertical space to leave between items.

Values: $0 \leq XtNhSpace$
 $0 \leq XtNvSpace$

If the items are of different sizes in a row or column, the spacing applies to the widest or tallest dimension of all items in the row or column.

XtNitemFields

Class	Type	Default	Access
XtCitemFields	String *	NULL	SGI

Synopsis: The list of resource names used to identify the records in the `XtNitems` list.

Values: A pointer to an application-defined structure containing a list of resources.

The application must ensure that this value points to a static list since flat containers reference this list after initialization, but do not copy its information.

XtNitemGravity

Class	Type	Default	Access
XtCItemGravity	int	NorthWestGravity	SGI

Synopsis: The region in its cell within the container in which an undersized item will be placed during layout.

Values: WestGravity/"west"
CenterGravity/"center"
NorthGravity/"north"
NorthEastGravity/"northEast"
NorthWestGravity/"northWest"
SouthGravity/"south"
SouthEastGravity/"southEast"
SouthWestGravity/"southWest"

This resource is used whenever the item's width or height is less than the column width or the row height of the place it is to occupy. The values of the XtNsameWidth and XtNsameHeight resources govern the column's width and the row's height.

*XtNitemMaxHeight/**XtNitemMaxWidth/**XtNitemMinHeight/**XtNitemMinWidth*

Class	Type	Default	Access
XtCItemMaxHeight	Dimension	OL_IGNORE	SGI
XtCItemMaxWidth	Dimension	OL_IGNORE	SGI
XtCItemMinHeight	Dimension	OL_IGNORE	SGI
XtCItemMinWidth	Dimension	OL_IGNORE	SGI

Synopsis: The maximum/minimum allowable height/width of items.

If any of these resources has the value OL_IGNORE (the default), the corresponding maximum/minimum height/width constraint is ignored.

Flat Resources

XtNItems

Class	Type	Default	Access
XtCItems	XtPointer	NULL	SGI

Synopsis: A list of application-defined structures, each representing an item.
 Values: A pointer to the list of application-defined item structures.

An item structure contains fields corresponding to the resources in the XtNitemFields list. The number of items in this list is contained in the XtNnumItems resource (see page 59).

The application must ensure that this value points to a static list since flat containers reference this list after initialization, and do not copy its information.

XtNItemsTouched

Class	Type	Default	Access
XtCItemsTouched	Boolean	FALSE	SG

Synopsis: The update status of the contents of the container.
 Values: TRUE – The contents need updating.
 FALSE – The contents do not need updating.

Whenever the application modifies an item list directly, this resource must be set to TRUE, to signal the flat container widget to update its contents. The flat container will relayout and redisplay its entire list of items, as if the list were new. This may result in geometry negotiations with the container’s parent widget.

After the container completes the processing associated with setting this resource to TRUE, it will reset the resource value to FALSE, indicating the integrity of the widget state with what is being displayed. This means XtGetValues() on XtNItemsTouched will always return FALSE.

It is not necessary to use this resource if the application modifies the list with the OlFlatSetValues() procedure (see page 354), nor is it necessary to use this resource whenever the application supplies a new list to the flat container.

XtNlabel

Class	Type	Default	Access
XtCLabel	OlStr	NULL	SG

Synopsis: The text string that appears in the item.

Values: Any OlStr value valid in the current locale.

XtNlabelImage

Class	Type	Default	Access
XtCLabelImage	XImage *	NULL	SGI

Synopsis: An XImage for display in an item label region.

The toolkit will ignore this resource if XtNlabel is non-NULL. The XImage will not be copied.

XtNlabelJustify

Class	Type	Default	Access
XtCLabelJustify	OlDefine	OL_LEFT	SGI

Synopsis: The justification of the label or image within the item.

Values: OL_LEFT/"left" - Left-justify the label or image.
 OL_CENTER/"center" - Center the label or image.
 OL_RIGHT/"right" - Right-justify the label or image.

XtNlabelTile

Class	Type	Default	Access
XtCLabelTile	Boolean	FALSE	SGI

Synopsis: The drawing of the label image as a single image or as a tiled pattern.

Values: TRUE/"true" - If the image will fit within the item, the label area will be filled with multiple renditions of the image in a tiled pattern.
 FALSE/"false" - A single image will be drawn as the item's label, justified as specified by the XtNlabelJustify resource.

The XtNlabelTile resource is ignored if XtNlabel is non-NULL.

Flat Resources

*XtNlayoutHeight/
XtNlayoutWidth*

Class	Type	Default	Access
XtCLayoutHeight	OlDefine	OL_MINIMIZE	SGI
XtCLayoutWidth	OlDefine	OL_MINIMIZE	SGI

Synopsis: The resize policy of flat containers when items change.

Values: OL_MINIMIZE/"minimize" - The container will modify its height or width to be just large enough to tightly wrap around its items. Thus, the container will grow and shrink depending on the size needs of its items. This policy will override any width or height resources that the application has set previously.

OL_MAXIMIZE/"maximize" - The container will increase its height or width to be just large enough to tightly wrap around its items, regardless of its current height or width, but will not give up extra space. Thus, the container will grow, but never shrink, depending on the size needs of its items.

XtNlayoutType

Class	Type	Default	Access
XtCLayoutType	OlDefine	OL_FIXEDROWS	SGI

Synopsis: The axis in the grid of items that is considered the major axis for the layout policy.

Values: OL_FIXEDCOLS/"fixedcols" - The layout will have a maximum number of columns equal to the value specified by the XtNmeasure resource, and there will be enough rows to hold all items. Items are placed in row-major order; i.e., the columns of the current row are filled before filling any columns in the next row.

OL_FIXEDROWS/"fixedrows" - The layout will have a maximum number of rows equal to the value specified by the XtNmeasure resource, and there will be enough columns to hold all items. Items are placed in column-major order; i.e., the rows of the current column are filled before filling any rows in the next column.

XtNmeasure

Class	Type	Default	Access
XtCMeasure	int	1	SGI

Synopsis: The number of items allowed in the major direction for the layout policy.

Values: $0 < \text{XtNmeasure}$

The major direction is determined by the `XtNlayoutType` resource. For a column-major layout, at most `XtNmeasure` columns will be displayed, and as many rows as are needed to display all items within this number of columns. For a row-major layout, at most `XtNmeasure` rows will be displayed, and as many columns as are needed to display all items within this number of rows.

XtNnumItemFields

Class	Type	Default	Access
XtCNumItemFields	Cardinal	0	SGI

Synopsis: The number of resource names contained in `XtNitemFields`.

XtNnumItems

Class	Type	Default	Access
XtCNumItems	Cardinal	0	SGI

Synopsis: The number of items.

Values: The number of elements in the `XtNitems` list (see page 56).

XtNsameHeight

Class	Type	Default	Access
XtCSameHeight	OlDefine	OL_ALL	SGI

Synopsis: The items forced to be the same height within the container.

Values: OL_ALL/"all" - All items will be the same height.

OL_ROWS/"rows" - All items appearing in the same row will be the same height.

OL_NONE/"none" - Items will be placed in fixed-height rows, but the height of each item will be unaffected. The height of each row will be the height of the tallest item.

*Flat Resources**XtNsameWidth*

Class	Type	Default	Access
XtCSameWidth	OlDefine	OL_COLUMNS	SGI

Synopsis: The items forced to be the same width within the container.

Values: OL_ALL/"all" - All items will be the same width.

OL_COLUMNS/"columns" - All items appearing in the same column will be the same width.

OL_NONE/"none" - Items will be placed in fixed-width columns, but the width of each item will be unaffected. The width of each column will be the width of the widest item.

Activation Types



This chapter explains what Activation Types are, explains how they are used in the toolkit, and describes those Activation Types that are common to several OLIT widgets.

Activation Type Description

What is an Activation Type?

OPEN LOOK defines a set of semantics that a user can invoke to control the user interface. Some examples of these semantics are:

MENU	Popup a menu
SCROLLEDOWN	Scroll the view down one screen
NEXTFIELD	Move focus to the next object

See the *OPEN LOOK GUI Functional Specification* and the *OPEN LOOK Mouseless Specification* for more information on GUI semantics.

Currently, OLIT maps these semantics to *virtual events* inside the toolkit (termed “virtual” because they do not necessarily correspond to a particular X11 input event). OLIT’s convention is to add a prefix of “OL_” to the semantic name to name the type of virtual event, which is called the Activation Type.

Activation Type Description

For example, for the semantics listed previously, the corresponding OLIT Activation Types are:

```
OL_MENU and OL_MENUKEY (for Mouseless mode)
OL_SCROLLDOWN
OL_NEXTFIELD
```

Each OLIT widget supports a set of Activation Types for which it knows how to respond. Some Activation Types are supported by all widgets, such as OL_HELP, and those used for Mouseless operation (described later in this chapter), and some are specific to a certain widget.

So, when a user performs an input action, OLIT translates the generated XEvent (or combination of XEvents) into an Activation Type defined by a set of configurable toolkit bindings (described later) and then delivers the corresponding “virtual event” to the widget. If the widget supports that Activation Type, it performs the corresponding action. For example, if the user presses the MENU mouse button on a MenuButton widget, OLIT translates this to the OL_MENU Activation Type, delivers an OL_MENU activation to the MenuButton widget, and the MenuButton responds by popping up its menu.

Interposing on Activation Types

If an application wishes to monitor or interpose on the virtual events delivered to a particular widget, it can register an XtnConsumeEvent callback on the widget. (For detailed information on this callback, see “XtnConsumeEvent” on page 26 for the Primitive or Manager classes.) This callback will get called just *before* the Activation Type is delivered to the widget. If the application wishes to prevent this Activation from being delivered to the widget, it can change the *consumed* Boolean field in the *call_data* to TRUE inside the callback. This callback allows the application to perform some custom action in replacement of, or in addition to, the standard widget behavior.

Programmatically Activating Widgets

Since the widgets respond to Activation Types (not just X11 events), the application can easily simulate these user semantics by calling `OLActivateWidget()` with the desired Activation Type. For example, if an application wants text to scroll down in response to an OblongButton being pressed, it simply needs to put the following statement in the `XtnSelect` callback for the button:

Activation Type Description

```
OlActivateWidget(scrolledwindow, OL_SCROLLDOWN, NULL);
```

OLIT also supports “associating” widgets with each other such that if a widget is activated with an Activation Type that it does not support, the Activation will be automatically passed on to an associated widget, called a “follower.” Applications can associate widgets using the `OlAssociateWidget()` routine. For example, if an application wants any scrolling Activation Types to be passed from an `OblongButton` widget to a `Scrollbar`, the application simply needs to make the `Scrollbar` a “follower” of the `OblongButton`:

```
OlAssociateWidget(button, scrollbar, FALSE);
```

See “Initialization and Activation Functions” on page 92 for more information on programmatically activating widgets.

Mapping X11 Events to Activation Types

By making the widget respond to high-level Activation Types, as opposed to having it respond to particular X11 input events, this model allows more flexibility for both the user and the programmer. The Activation Types are mapped to actual X11 input events through a set of toolkit resources. These toolkit resources have a set of default bindings that the user can easily change using the standard resource mechanism. For example, the resources and default bindings for the Activation Types mentioned previously are the following:

Activation Type	Resource	Default Binding	Description
OL_MENU	XtNmenuBtn	<Button3>	(MENU mouse button)
OL_MENUKEY	XtNmenuKey	a<Space>	(Alt+Space key)
OL_SCROLLDOWN	XtNscrollDownKey	a<Down>	(Alt+R14 key)
OL_NEXTFIELD	XtNnextFieldKey	<Tab>,c<Tab>	(Tab or Control+Tab)

The syntax of the “Default Bindings” column is explained below. The `OL_MENU` Activation Type could be mapped instead to the `MENU` mouse button by putting the following line in a Resource file:

```
*menuBtn: <Button2>
```

Activation Type Description

A complete table of the OLIT Activation Types and their default bindings is in Table 3-1 on page 65. The following abbreviations for modifier keys are used to shorten the “Default Bindings” column of the table, and can also be used in resource specifications:

No Modifiers	n	Meta key	m	Mod2 key	2
Alt key	a	Hyper key	h	Mod3 key	3
Ctrl key	c	Super key	su	Mod4 key	4
Shift key	s	Mod1 key	1	Mod5 key	5

There are three different ways to specify key bindings (and keyboard accelerators) in a resource file:

1. A key event specified using a subset of the Xt translation manager syntax. The syntax is specified in EBNF notation, following the same conventions used in the *Xt Intrinsic Reference Manual*, Appendix B.

```
keyseq      = [modifier_list] "<Key>" <keysym_name>
modifier_list = {modifier_name} | "None" | "n"
modifier_name = <see modifier names table above>
```

2. The existing OLIT syntax, which is similar to the Xt translation manager syntax and maintained for backward compatibility.

```
keyseq      = [modifier_list] "<" <keysym_name> ">"
```

3. A general OpenWindows syntax that is understood by OLIT, `xview(1)`, and `olwm(1)`.

```
keyseq      = {[modifier_name] "+"} <keysym_name>
```

For example, to bind the SELECTKEY command to be activated when the space bar is pressed and the Meta modifier key (⌘) is held down, any of the following may be used:

```
*selectKey: Meta<Key>space ; Xt translation syntax
*selectKey: Meta<space>    ; OLIT syntax
*selectKey: Meta+space     ; generic syntax
```

Note – On keyboards that don’t have an actual meta key, the meta-key event is generated by pressing the Control and Alt keys at the same time.

Up to two bindings may be specified for a virtual event as a comma-separated list; for example:

```
*selectKey: Meta<Key>space, Ctrl<Key>space
```

Activation Type Description

Keyboard accelerators may also be specified using any of the three methods, but only one binding can be associated with an accelerator.

Table 3-1 OLIT Activation Types

Activation Type	Semantic	Resource Name	Default Binding
OL_ADJUST	ADJUST	XtNadjustBtn	<Button2>
OL_ADJUSTKEY	ADJUSTKEY	XtNadjustKey	a<Insert>
OL_CANCEL	CANCEL	XtNcancelKey	<Escape>
OL_CHARBAK	CHARBAK	XtNcharBakKey	<Left>
OL_CHARFWD	CHARFWD	XtNcharFwdKey	<Right>
OL_CONSTRAIN	CONSTRAIN	XtNconstrainBtn	s<Button1>
OL_COPY	COPY	XtNcopyKey	<F16>
OL_CUT	CUT	XtNcutKey	<F20>
OL_DEFAULTACTION	DEFAULTACTION	XtNdefaultActionKey	<Return>, c<Return>
OL_DELCHARBAK	DELCHARBAK	XtNdelCharBakKey	<BackSpace>, <Delete>
OL_DELCHARFWD	DELCHARFWD	XtNdelCharFwdKey	s<BackSpace>, c<d>
OL_DELLINE	DELLINE	XtNdelLineKey	m<BackSpace>, m<Delete>
OL_DELLINEBAK	DELLINEBAK	XtNdelLineBakKey	c<BackSpace>, c<v>
OL_DELLINEFWD	DELLINEFWD	XtNdelLineFwdKey	c<Delete>, c<k>
OL_DELWORDBAK	DELWORDBAK	XtNdelWordBakKey	s<BackSpace>, c<w>
OL_DELWORDFWD	DELWORDFWD	XtNdelWordFwdKey	c s<Delete>
OL_DOCEND	DOCEND	XtNdocEndKey	c<R13>
OL_DOCSTART	DOCSTART	XtNdocStartKey	c<R7>
OL_DRAG	DRAG	XtNdragKey	<F5>
OL_DROP	DROP	XtNdropKey	<F2>
OL_DUPLICATE	DUPLICATE	XtNduplicateBtn	c<Button1>
OL_DUPLICATEKEY	DUPLICATEKEY	XtNduplicateKey	s<space>
OL_HELP	HELP	XtNhelpKey	<Help>
OL_HSBMENU	HSBMENU	XtNhorizSBMenuKey	a<h>
OL_LINEEND	LINEEND	XtNlineEndKey	<R13>, c<e>

3

Activation Type Description

Table 3-1 OLIT Activation Types (Continued)

Activation Type	Semantic	Resource Name	Default Binding
OL_LINESTART	LINESTART	XtNlineStartKey	<R7>, c<a>
OL_MENU	MENU	XtNmenuBtn	<Button3>
OL_MENUDEFAULT	MENUDEFAULT	XtNmenuDefaultBtn	c<Button3>
OL_MENUDEFAULTKEY	MENUDEFAULTKEY	XtNmenuDefaultKey	c<space>
OL_MENUKEY	MENUKEY	XtNmenuKey	a<space>
OL_MOVEDOWN	MOVEDOWN	XtNdownKey	<Down>
OL_MOVELEFT	MOVELEFT	XtNleftKey	<Left>
OL_MOVERIGHT	MOVERIGHT	XtNrightKey	<Right>
OL_MOVEUP	MOVEUP	XtNupKey	<Up>
OL_MULTIDOWN	MULTIDOWN	XtNmultiDownKey	c<Down>
OL_MULTILEFT	MULTILEFT	XtNmultiLeftKey	c<Left>
OL_MULTIRIGHT	MULTIRIGHT	XtNmultiRightKey	c<Right>
OL_MULTIUP	MULTIUP	XtNmultiUpKey	c<Up>
OL_NEXT_FIELD	NEXT_FIELD	XtNnextFieldKey	<Tab>, c<Tab>
OL_PAGEDOWN	PAGEDOWN	XtNpageDownKey	a<R15>
OL_PAGELEFT	PAGELEFT	XtNpageLeftKey	a c<R9>
OL_PAGERIGHT	PAGERIGHT	XtNpageRightKey	a c<R15>
OL_PAGEUP	PAGEUP	XtNpageUpKey	a<R9>
OL_PAN	PAN	XtNpanBtn	c s <Button1>
OL_PANEEND	PANEEND	XtNpaneEndKey	c s<R13>
OL_PANESTART	PANESTART	XtNpaneStartKey	c s<R7>
OL_PASTE	PASTE	XtNpasteKey	<F18>, c<y>
OL_PREV_FIELD	PREV_FIELD	XtNprevFieldKey	s<Tab>, c s<Tab>
OL_PROPERTY	PROPERTY	XtNpropertiesKey	<F13>
OL_RETURN	RETURN	XtNreturnKey	<Return>
OL_ROWDOWN	ROWDOWN	XtNrowDownKey	<Down>
OL_ROWUP	ROWUP	XtNrowUpKey	<Up>

*Activation Type Description**Table 3-1* OLIT Activation Types (Continued)

Activation Type	Semantic	Resource Name	Default Binding
OL_SCROLLBOTTOM	SCROLLBOTTOM	XtNscrollBottomKey	a c<R13>
OL_SCROLLDOWN	SCROLLDOWN	XtNscrollDownKey	a<Down>
OL_SCROLLLEFT	SCROLLLEFT	XtNscrollLeftKey	a<Left>
OL_SCROLLLEFTEDGE	SCROLLLEFTEDGE	XtNscrollLeftEdgeKey	a <R7>
OL_SCROLLRIGHT	SCROLLRIGHT	XtNscrollRightKey	a<Right>
OL_SCROLLRIGHTEDGE	SCROLLRIGHTEDGE	XtNscrollRightEdgeKey	a <R13>
OL_SCROLLTOP	SCROLLTOP	XtNscrollTopKey	a c<R7>
OL_SCROLLUP	SCROLLUP	XtNscrollUpKey	a<up>
OL_SELCHARBAK	SELCHARBAK	XtNselCharBakKey	s<Left>, s c
OL_SELCHARFWD	SELCHARFWD	XtNselCharFwdKey	s<Right>, s c<f>
OL_SELECT	SELECT	XtNselectBtn	<Button1>
OL_SELECTKEY	SELECTKEY	XtNselectKey	<space>
OL_SELFLIPENDS	SELFLIPENDS	XtNselFlipEndsKey	a<Insert>
OL_SELLINE	SELLINE	XtNselLineKey	c a<Left>
OL_SELLINEBAK	SELLINEBAK	XtNselLineBakKey	s<R7>, s c<p>
OL_SELLINEFWD	SELLINEFWD	XtNselLineFwdKey	s<R13>, s c<n>
OL_SELWORDBAK	SELWORDBAK	XtNselWordBakKey	c s<Left>
OL_SELWORDFWD	SELWORDFWD	XtNselWordFwdKey	c s<Right>
OL_STOP	STOP	XtNstopKey	<F11>
OL_TOGGLEPUSHPIN	TOGGLEPUSHPIN	XtNtogglePushpinKey	c<t>
OL_UNDO	UNDO	XtNundoKey	<F14>
OL_VSBMENU	VSBMENU	XtNvertSBMenuKey	a<v>
OL_WORDBAK	WORDBAK	XtNwordBakKey	c<Left>
OL_WORDFWD	WORDFWD	XtNwordFwdKey	c<Right>

Common Activation Types

Common Activation Types

All OLIT widget classes that accept input focus support the semantics described in section 2.3 of the *OPEN LOOK Mouseless Specification* and at least the Activation Types shown in Table 3-2.

Table 3-2 Common Activation Types

Activation Type	Semantics	Resource Name
OL_CANCEL	CANCEL	XtNcancelKey
OL_DEFAULTACTION	DEFAULTACTION	XtNdefaultActionKey
OL_HELP	HELP	XtNhelpKey
OL_MOVEDOWN	MOVEDOWN	XtNdownKey
OL_MOVELEFT	MOVELEFT	XtNleftKey
OL_MOVERIGHT	MOVERIGHT	XtNrightKey
OL_MOVEUP	MOVEUP	XtNupKey
OL_NEXTFIELD	NEXTFIELD	XtNnextFieldKey
OL_PREVFIELD	PREVFIELD	XtNprevFieldKey
OL_TOGGLEPUSHPIN	TOGGLEPUSHPIN	XtNtogglePushpinKey

These activation types have the following meanings:

OL_CANCEL

Redirect the OL_CANCEL activation type, delivering it to the first widget ancestor inclusive of the object receiving the activation that is an instance of `vendorShellWidgetClass` or a subclass thereof.

OL_DEFAULTACTION

Redirect the OL_DEFAULTACTION activation type, delivering it to the first widget ancestor inclusive of the object receiving the activation that is an instance of `vendorShellWidgetClass` or a subclass thereof.

OL_HELP

Invoke the system help facility. If the value of the toolkit resource `XtNhelpModel` is OL_POINTER, then help will be invoked for the object currently under the pointer at the time of activation; however, if the value of `XtNhelpModel` is OL_INPUTFOCUS, then help will be invoked for the object that currently holds input focus.

OL_MOVEDOWN

Move the input focus from the current object to the object below the current holder of input focus in the traversal order. The definition of the object below the current holder of the input focus in the traversal order is context sensitive and system dependent.

***OL_MOVELEFT/
OL_PREVFIELD***

Move the input focus from the current object to the previous object in the traversal order. The definition of the previous object in the traversal order is context sensitive and system dependent.

***OL_MOVERIGHT/
OL_NEXTFIELD***

Move the input focus from the current object to the next object in the traversal order. The definition of the next object in the traversal order is context sensitive and system dependent.

OL_MOVEUP

Move the input focus from the current object to the object above the current holder of input focus in the traversal order. The definition of the object above the current holder of the input focus in the traversal order is context sensitive and system dependent.

OL_TOGGLEPUSHPIN

Redirect the `OL_TOGGLEPUSHPIN` activation type, delivering it to the first widget ancestor inclusive of the object receiving the activation that is an instance of `VendorShellWidgetClass` or a subclass thereof.

≡ 3

Common Activation Types

Internationalization Features



This chapter focuses on the general issues of internationalizing applications, and how OLIT address these issues. It also provides a simple example application, which demonstrates how OLIT makes it easy to create and use internationalized applications.

Introduction

As the international market for software becomes increasingly important, software manufacturers need a way to *internationalize* their applications without having to re-engineer or re-compile the application for each language. Ideally, a single version of an application should be able to support any number of languages.

Internationalization refers to the ability of a Graphical User Interface (GUI) to display text in various languages and conventions, and to accept textual input in those languages. *Locale* or *localization* refers to the alphabet and conventions of a particular language or cultural environment. Date, time, and monetary format are examples of locale conventions.

The OLIT toolkit, a user-interface toolkit based on the X Window System and the OPEN LOOK graphical user interface, allows developers to simply and easily create internationalized applications without having to modify the source code for each supported language. An application developed and compiled with OLIT will be able to operate in any of the supported languages and process data according to the rules of that language.

System Requirements

OLIT features that support internationalized applications are referred to as *international OLIT* in this chapter. This internationalized version of OLIT supports both European languages (French, German, Italian, and Swedish) and Asian languages (Japanese, Korean, Taiwanese Chinese, and PRC Chinese).

Adding new languages to an international OLIT application consists primarily of setting the default text format and changing messages, labels, and other strings in resource files. For Asian locales, one must also provide localized *input methods* (IM) to allow for multibyte character composition. The input method is the method by which text is entered into the system. Input methods are specific to each Asian language and are provided by Sun Microsystems. They can, however, be redefined and changed by OEMs or independent software vendors.

System Requirements

The first release of international OLIT enables developers to localize to several European and Asian languages. To run this release of international OLIT, you need Solaris®, OpenWindows™, and the *Feature Package* for the locale in which you intend to run it. For example, you need the Japanese Feature Package (JFP) for Japan, the Korean localization package for Korea, and the French localization package for France. These packages consist of extensions to SunOS and incorporate numerous facilities for handling local linguistic and cultural conventions.

Issues Involved in Internationalizing Applications

To internationalize your application, you must address the following issues.

1. **Locale Setting.** Locale Setting is the method by which the language or cultural environment is set. See “Locale Setting” on page 73.
2. **Character Encoding.** Conventional applications use 7-bit ASCII encoding to represent each character. However, some languages have larger character sets that require more than the 128 character range permitted by 7-bit encoding; for example, European locales use an 8-bit encoding and Asian locales use extended encoding mechanisms. Character encoding is the method by which a language’s character set is represented. See “Character Encoding and Text Formats” on page 74 and the Sun *Software Internationalization Guide* for further details on character encoding.

Locale Setting

3. **Font Set Handling.** Font handling is simple for the U.S. and European languages that OLIT supports because they only use one character set. However, some languages use multiple character sets and therefore require multiple fonts or a *font set*. See “Font Set Handling” on page 77.
4. **Localized Text Handling.** The developer needs to be able to use application strings (i.e., error messages, menu text, button text, etc.) in the native language, and have those strings retrieved in the language specified by the locale. See “Localized Text Handling” on page 79.
5. **Input Method (Asian Locales Only).** This is the method by which users enter the text of a language. To enter data into a conventional application, the user simply types in the information to be processed. Some languages, however, consist of multiple alphabets which require several keystrokes to create one character. This special handling is called the input method. See “Input Method” on page 80.
6. **Standards.** Software internationalization is supported by a number of standards organizations. These include IEEE (POSIX), ANSI, X/Open, and the MIT X Consortium. In order to make applications portable across a wide variety of hardware platforms, it is important to use a toolkit that follows these standards as much possible, as international OLIT does. See “Standards” on page 89.

This chapter describes each of these issues in detail and discusses how these issues are addressed by international OLIT.

Locale Setting

The X Toolkit Intrinsic, on which OLIT is layered, provides an application resource called `XnlLanguage` (class `XnlLanguage`) that announces the user’s locale to the toolkit and the operating system.

Resource	Type	Default	Access
<code>XnlLanguage</code>	<code>XtNstring</code>	NULL	I

The currently supported values for this resource are: `de`, `fr`, `it`, `ja`, `ko`, `sv`, `zh`, and `zh_TW`. There are three ways to establish the locale, as shown in the following list:

Character Encoding and Text Formats

1. Specify `-xnllanguage` on the command line of an OLIT application. For example, to set the locale for an application called *myapplication* to Korean, you type:

```
% myapplication -xnllanguage ko
```

2. Specify `*xnllanguage: language` in a X11 Resource Manager database file. For example, to set the locale to traditional Chinese, you add the following line to your `.Xdefaults` file:

```
*xnllanguage: zh_TW
```

3. Set the `LANG` environment variable in the shell that you are starting the application from. For example, to set the locale to Japanese, you type the following:

```
% setenv LANG ja
```

To set the locale to French, you type the following:

```
% setenv LANG fr
```

Establishing the OS locale is the responsibility of OLIT and Xt. You should not use the OS function `setlocale(3)` in your OLIT applications, since OLIT already calls this function internally.

Character Encoding and Text Formats

OLIT supports three character encoding types, or *text formats*:

- Single byte, used in the USA and Europe, which represents each character with one byte
- Multibyte, used in Asian, which represents each character with a variable number of bytes
- Wide character, used in Asian, which represents each character with a fixed number of bytes

Applications can create single-byte, multibyte, or wide character OLIT objects, or a combination of these. If you are writing an application intended for single-byte locales only, you may want to use single-byte text format to avoid the performance overhead incurred by processing multibyte text. However, note that if you use single-byte format, it will be harder to internationalize your application to Asian locales at a later date.

Character Encoding and Text Formats

The multibyte text format is fully compatible with ASCII. Since each different character can potentially be a different size, programming with multibyte text can be difficult. However, the multibyte format uses memory efficiently.

Wide character text format, on the other hand, is easier to program since all characters are represented with the same number of bytes. However, the wide character format consumes more storage since it represents all characters with a fixed number of bytes. If all characters are ASCII, many of the bytes will be superfluous.

When deciding which text format is appropriate for a user interface, you should take the following considerations into account:

- Conversion between formats reduces performance.
- OLIT honors the requested text format inside the object implementation.
- Processing multibyte data is inherently more time consuming than processing single-byte or wide character data. Objects that perform intensive data manipulation (for example text-editing object such as OLIT's `TextEdit` or `TextField` widgets) will perform better if created as wide character objects. (If an object is certain to only handle 8-bit data, the optimal solution is to create single-byte objects.)

You should design your application so that conversion between formats is minimized. For example, you may want to decide on an object-by-object basis whether textual data will be processed intensively and use wide character format if it is.

Note – The widget makes assumptions based on the text format. For example, if you specify the single-byte format and supply a wide character label, it causes an error.

OLIT widget resources for presentation text are of type `OlStr`. For more information on this type, see “Setting the Default Text Format for an Entire Application” on page 76. To provide support for multiple text formats, international OLIT introduces the `XtNtextFormat` resource. This resource allows you to inform widgets what text format to expect for resources associated with presentation text. This is an interface for programmers only; there is no equivalent interface for users to specify widget text formats. The text format of an object is persistent for the lifetime of the object; it is established at object creation time and cannot be changed.

Character Encoding and Text Formats

There are several ways to set the text format:

- Do nothing. If you do nothing, the text format defaults to single byte.
- Set the default text format for the entire application with the `OlSetDefaultTextFormat()` function.
- Set the default text format for an individual widget by changing the widget's `XtNtextFormat` resource.

Setting the Default Text Format for an Entire Application

As a convenience, OLIT maintains a default text format, which widgets inherit when they are created without their text format explicitly specified in the argument list passed to one of the `XtCreateWidget()` family of functions.

For compatibility with previous OLIT releases, the default text format is single-byte unless you change it. To change it, use the `OlSetDefaultTextFormat()` function, which is defined as follows:

```
void OlSetDefaultTextFormat(OlStrRep format);
```

where *format* specifies the character representation of the text. The *format* argument can have the following values:

format Value	Meaning
<code>OL_SB_STR_REP</code>	Single-byte character representation
<code>OL_WC_STR_REP</code>	Wide character representation
<code>OL_MB_STR_REP</code>	Multibyte character representation

Your application should call `OlSetDefaultTextFormat()` immediately after `OlToolkitInitialize()` and before creating the widget hierarchy. Objects subsequently created will have the text format specified in the most recent call to `OlSetDefaultTextFormat()`, unless overridden by explicit arguments. Note that if the application is single-byte only, it does not need to call `OlSetDefaultTextFormat()`.

You can create a widget hierarchy consisting of a combination of multibyte, wide character, and single byte objects. You can do this either by changing the default text format between widget creation calls or by specifying an object's text format explicitly when creating it.

Setting the Text Format for an Individual Widget

OLIT provides a resource, `XtNtextFormat`, which allows you to set the text format for individual widgets. See “`XtNtextFormat`” on page 29.

Font Set Handling

To represent data that consists of multiple character sets, Release 5 of the X Window System provides the notion of a *Font Set*. An `XFontSet` is a X11R5 data structure that supports this notion. From the user’s perspective, an `XFontSet` represents a list of X11 Logical Font Description (XLFD) fonts that allows the application to fully represent the characters used in a particular locale. For full details on Font Sets, refer to the X11R5 documentation.

To specify a font set in a resource file, use a comma-separated list of fonts. For example, you can enter the following in your `.Xdefaults` file:

```
*font: -misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-1, \
-jis-fixed-medium-r-normal--16-150-75-75-c-160-jisx0208.1983-0, \
-misc-fixed-medium-r-normal--0-0-75-75-c-0-jisx0201.1976-0
```

When you internationalize your application, you should specify fonts in a resource file. If you specify them within your application, it will be impossible for others to localize it.

OlFont

OLIT supports both the font and font set notions by introducing the `OlFont` type for objects that display text. `OlFont` is an opaque pointer type whose interpretation depends on the setting of its associated `XtNtextFormat` resource. If you create an object as multibyte or wide character, the value of the `OlFont` field will be a valid `XFontSet` identifier. If you create an object as single byte, `OlFont` field will be a valid pointer to an `XFontStruct`.

If the text format is `OL_SB_STR_REP` and a font set has been specified (using a comma-separated list), the first font in the list will be used to construct an `Xfont struct`.

Setting the Default Font or Font Set

The OLIT widget set provides an `XtNolDefaultFont` resource that specifies the default font or fontset for an application. This resource is discussed in “`XtNolDefaultFont`” on page 14.

If no font is specified for a widget’s `XtNfont` resource, `XtNolDefaultFont` determines the widget’s font. In international OLIT, the default value of `XtNolDefaultFont` is determined as follows:

- In the C locale and European locales, the default value of `XtNolDefaultFont` is Lucida sans serif with `Resolution_X` and `Resolution_Y` set to 75.
- In Asian locales, the default value of `XtNolDefaultFont` is the font set required to display all the characters in the code set of the locale.
- An internationalized OLIT application running in Asian locales must define the `XtNolDefaultFont` resource in its locale-specific `app-defaults` file. If the `XtNolDefaultFont` resource is not specified with a list of fully-defined font names the application’s startup time will be increased significantly. The application’s startup time is dependent on the number of wildcards in the XLFD font names. Use completely specified XLFD font names to optimize performance at startup time. The following is an example of how to set the `XtNolDefaultFont` resource in the Japanese locale:

```
*olDefaultFont: \
-sun-gothic-medium-r-normal--16-140-75-75-c-70-jisx0201.1976-0, \
-sun-gothic-medium-r-normal--16-140-75-75-c-140-jisx0208.1983-0
```

Getting the Default Font or Font Set

The `OlGetDefaultFont()` convenience routine enables you to get the font or font set that will be used if you do not set the `XtNfont` resource for a widget. The syntax of `OlGetDefaultFont()` is:

```
OlFont OlGetDefaultFont(widget w);
```

where `w` is a widget in the application for which you want to get the default font.

If `XtNolDefaultFont` specifies a font that is not available, `OlGetDefaultFont()` returns a null pointer.

Localized Text Handling

OLIT provides resources that enable you to set various text strings (messages, menu labels, button labels, etc.) in an application. For information on the text string resources for a particular widget, see the reference section for the widget. When you create an internationalized application, you should remove all these resources from your application and keep them in locale-specific resource files.

Using Internationalized Help

You can register multibyte help text for a widget with OLIT's `OlRegisterHelp()` function. To register single-byte help, use multibyte help. The OLIT Help API currently supports multibyte only. A wide character API will be added in the future. The syntax for `OlRegisterHelp()` is as follows:

```
void OlRegisterHelp(
    OlDefine    id_type,
    XtPointer   id,
    String      tag,
    OlDefine    source_type,
    XtPointer   source);
```

To use `OlRegisterHelp()` with international OLIT, specify one of the following values for the *id_type* argument.

<i>id_type</i> Value	Meaning
OL_WIDGET_HELP	Specifies multibyte format help text for an individual widget (for backward compatibility).
OL_FLAT_HELP	Specifies multibyte format help text for a flat widget (for backward compatibility)

If you specify `OL_DISK_SOURCE` for the *source_type* argument, you must specify a single-byte or multibyte filename for the *source* argument.

Help searches the directories specified by `XFILESEARCHPATH` for the specified filename. If you set the filename from within your program as an absolute path, help ignores `XFILESEARCHPATH`. If the `XFILESEARCHPATH` expansion does not find a file for help, the current directory is searched. Within the value

Input Method

of XFILESEARCHPATH, any instance of the string “%T” is expanded to “help” and “%N” is expanded to the filename the programmer specifies as the source parameter to `OlRegisterHelp()`.

For a description of the other arguments of `OlRegisterHelp()`, see “Help Function” on page 146.

Localized Messages

OLIT issues textual error message in the program’s locale at startup. To do this, OLIT registers a private language procedure with the Intrinsics during `OlToolkitInitialize()`; see page 92.

Do not register an application-specific language procedure. If you do, it will interfere with OLIT’s locale-announcement mechanism and you will have to take responsibility for it in your application. If your application requires processing before OLIT’s language procedure is called, you can provide your own procedure and call OLIT’s procedure from it. To get the language procedure registered by OLIT, call:

```
olit_proc = XtSetLanguageProc(NULL, NULL, NULL);
```

Multibyte and Wide Character Text Buffer Functions

International OLIT provides multibyte and wide character equivalents to the single-byte text buffer functions in previous releases. A list of these functions and their syntax are provided in “Text Buffer Functions for Internationalization” on page 176.

Input Method

The input method (IM) is the algorithm by which users enter the text of a language. The input method for each language may be different, depending on the linguistic structure and conventions of that language.

International OLIT follows the *X Window System Version 11, Input Method Specification, Draft 3.0*. This specification was derived as a result of discussions among X Consortium members on standardizing the input handling of characters in various languages by X clients.

Input Method

For many languages, there isn't a one-to-one key to character mapping, regardless of how the keyboard is configured. In order to support such languages, an input method is required.

In English and European languages, users enter the desired text by typing in a sequence of letters to create a word. However, for Asian languages based on ideographic characters, input is more complicated. For example, there are two phonetic alphabets in Japanese—Hiragana and Katakana—and the traditional ideogrammatic alphabet, Kanji. In any piece of writing, all three alphabets may be used. Japanese words can also be spelled out phonetically in English. This is called Romaji.

To handle European language characters where there is no key on the standard keyboard that maps to the desired character, use the Compose key to initiate a composed character sequence (e.g., Compose key, <letter>, <accent or diacritical mark>).

To handle languages for which there isn't a one-to-one key to character mapping, input methods provide features such as the following:

- A control key sequence, which selects the *input mode*
- A pre-edit region, which displays characters as the user enters them but before the user *commits* them
- A lookup choice region, which displays ideographic characters and allows the user to choose one
- A status region, which provides information such as whether conversion is activated and the state or mode of the input method

Text input widgets, in conjunction with some input methods, also can provide advanced, language-specific, pre-editing features. For instance, the OLIT TextEdit widget can detect certain conditions under which it will commit any uncommitted pre-edit text without the user having to take further action. This technique is known as *implicit commit*.

Example: In a mail application the user enters a message in Japanese and presses a “send” button to dispatch the composed message. If there is uncommitted pre-edit text, the user's intention is that it be part of the message. The pre-edit text has not, however, been committed to the text buffer. In this case it is useful for the toolkit to intervene and cause a commit to occur before the application processes the buffer to send the message.

Input Method

Details of which operations trigger implicit commit semantics in OLIT widgets can be found in the localization documentation for the appropriate languages.

The use of these features varies with the input method. For more information, see the documentation for the input method you are using.

Figure 4-1 shows the input method screen regions.

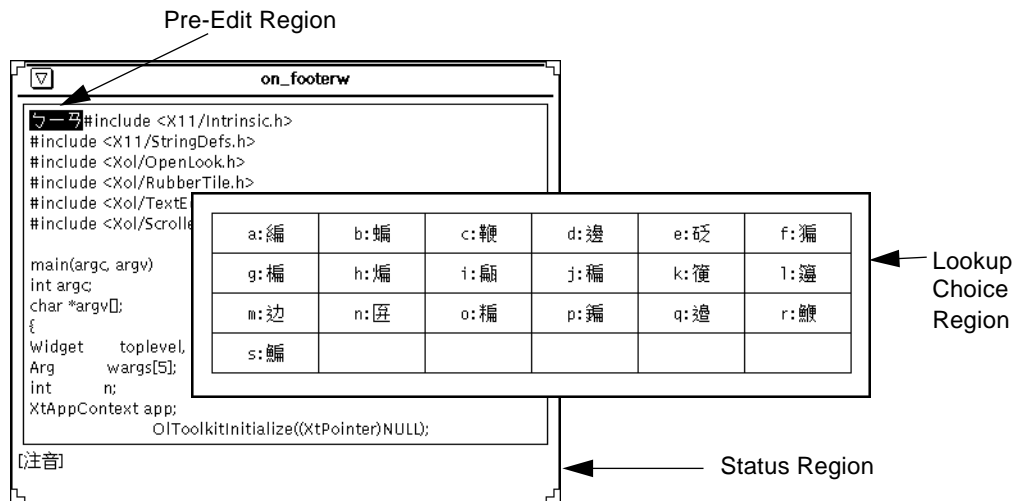


Figure 4-1 Input Method Screen Regions for zh_TW locale

Setting the Input Method Pre-Edit and Status Styles (Asian Locales Only)

There are two aspects of the input method that an OLIT application can control:

- The *pre-edit style*, which specifies where and how pre-edit data is presented. The pre-edit style can vary from widget to widget within a shell.
- The *status style*, which specifies where status feedback is presented. The status style, unlike the pre-edit, is an attribute of the shell and is expected to remain the same across all widgets inside the shell.

OLIT provides two new resources that specify the pre-edit and status styles: `XtNimPreeditStyle` and `XtNimStatusStyle`, described in the following sections.

XtNimPreeditStyle (Asian Locales Only)

The `XtNimPreeditStyle` resource selects the pre-edit style. This resource is supported by all the OLIT widgets that allow text input. See, for example, `TextEdit` (“`XtNimPreeditStyle`” on page 635).

Resource	Type	Default	Access
<code>XtNimPreeditStyle</code>	<code>OlImPreeditStyle</code>	<code>OL_NO_PREEDIT</code>	<code>GI</code>

If the specified style is not supported by the input method, the ability to pre-edit is lost. The currently supported pre-edit styles are:

<code>XtNimPreeditStyle</code> Value	Meaning
<code>OL_ON_THE_SPOT/</code> “ <code>onTheSpot</code> ”	IM directs the application to display the pre-edit data
<code>OL_OVER_THE_SPOT/</code> “ <code>overTheSpot</code> ”	IM displays pre-edit data in its own window
<code>OL_ROOT_WINDOW/</code> “ <code>rootWindow</code> ”	IM displays pre-edit data outside the application in a window that is a child of the base window
<code>OL_NO_PREEDIT/</code> “ <code>none</code> ”	IM does not display pre-edit data

The following figure shows an example of the `onTheSpot` pre-edit style. The pre-edit data is shown in reverse video. When the user commits the data, it is sent to the client and displayed in normal video.

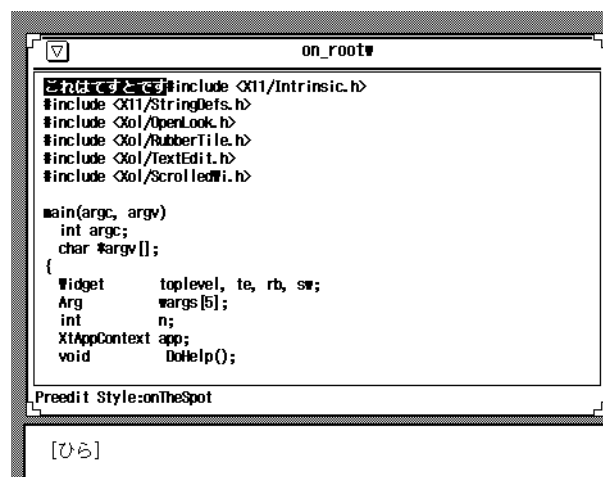


Figure 4-2 onTheSpot Pre-Edit Style for ja locale

The following figure shows an example of the `overTheSpot` pre-edit style.

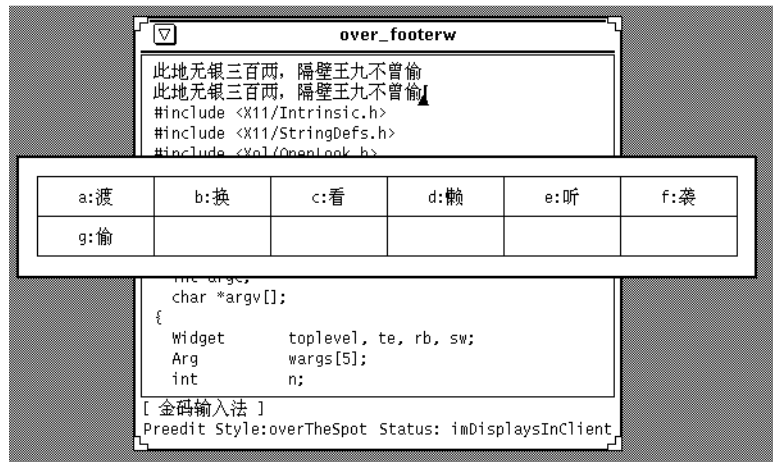


Figure 4-3 `overTheSpot` Pre-Edit Style for zh locale

The following figure shows an example of the `rootWindow` pre-edit style.



Figure 4-4 `rootWindow` Pre-Edit Style for zh locale

See the X11R5 documentation for a full description of each of the pre-edit styles.

XtNimStatusStyle (Asian Locales Only)

The `XtNimStatusStyle` resource, defined in the `VendorShell` class, determines the style of IM status feedback.

Resource	Type	Default	Access
<code>XtNimStatusStyle</code>	<code>OlImStatusStyle</code>	<code>OL_NO_STATUS</code>	GI

The supported styles are:

XtNimStatusStyle Value	Meaning
<code>OL_IM_DISPLAYS_IN_CLIENT/</code> <code>"imDisplaysInClient"</code>	The IM generates status feedback in the footer of the shell window
<code>OL_IM_DISPLAYS_IN_ROOT/</code> <code>"imDisplaysInRoot"</code>	The IM generates status feedback in a separate window
<code>OL_NO_STATUS/"none"</code>	The IM doesn't generate any status feedback

When `XtNimStatusStyle` is set to `OL_IM_DISPLAYS_IN_CLIENT`, a number of other resources are available to set characteristics of the application footer. See "Application Resources for the IM Footer (Asian Locales Only)" on page 87.

The following figure shows an example of the `imDisplaysInClient` status style.

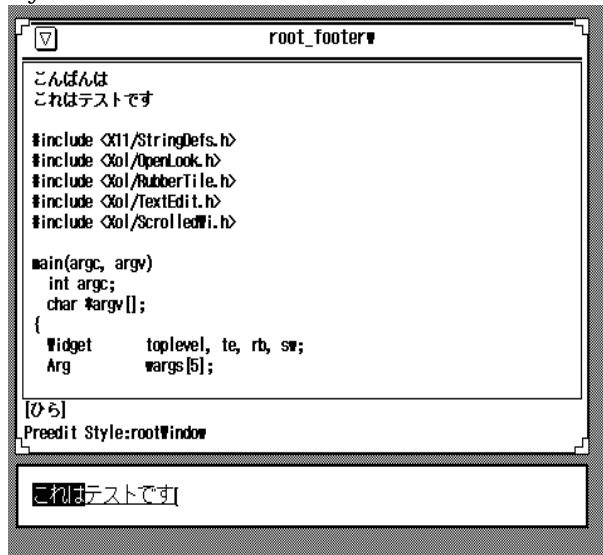


Figure 4-5 `imDisplaysInClient` Status Style for ja locale

The following figure shows an example of the `imDisplaysInRoot` status style.

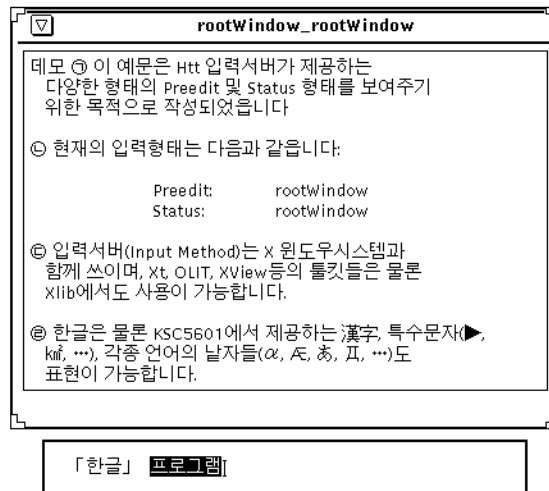


Figure 4-6 `imDisplaysInRoot` Status Style for ko locale

Application Resources for the IM Footer (Asian Locales Only)

Resource	Type	Default	Access
XtNimFontSet	OlFont	XtDefaultFont	SGI
XtNdefaultImName	String	NULL	SGI

XtNimFontSet (Asian Locales Only)

The `XtNimFontSet` resource specifies the internationalization IM status footer's font set.

XtNdefaultImName

The `XtNdefaultImName` resource specifies the string to identify the IM Server.

XtNshellTitle

`XtNshellTitle` is an `OlStr` resource that allows the title of shell widgets to be set. The shell being set must be a subclass of `VendorShell`. The usage of this resource is analogous to the `XtNtitle` resource defined by the Intrinsic classes. `XtNshellTitle` and `XtNtitle` are synchronized by OLIT; calling `XtSetValues()` on either will cause both to be updated.

Example

To internationalize a simple OLIT application for the European or Asian locales, you need to:

1. Remove any resources that contain display text from the application code and put them in a resource file.
2. Specify the text format for widgets that display text (by specifying a default text format for the application or by specifying the text format for individual widgets).
3. Specify a font set for resources that contain `OlFont` values (such as `XtNfont`) in the resource file.

Suppose you want to write an internationalized application that displays a `StaticText` widget with some wide character text in it. To do this, you would write code similar to the following.

```
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <Xol/OpenLook.h>
#include <Xol/StaticText.h>

main(argc, argv)
    int    argc;
    char   *argv[];
{
    Widget toplevel, msg_widget;
    XtAppContext app;

    /* Initialize the OLIT toolkit */
    OlToolkitInitialize((XtPointer)NULL);

    /* Set the default text format to wide character */
    OlSetDefaultTextFormat(OL_WC_STR_REP);

    toplevel = XtAppInitialize(&app, "Memo",
                              (XrmOptionDescList)NULL,
                              0, &argc, argv, NULL,
                              (ArgList) NULL, 0);

    /* Create a staticText widget. */
    msg_widget = XtVaCreateManagedWidget("msg",
                                          staticTextWidgetClass,
                                          toplevel,
                                          NULL);

    /* Realize the widgets and enter the event loop. */
    XtRealizeWidget(toplevel);
    XtAppMainLoop(app);
}
```

Note that this application does not specify the string that appears in the `StaticText` widget. You specify this text in the resource file as follows:

```
*StaticText.string: <text>
```

Also note that this application is different from a conventional OLIT application in that it calls `OlSetDefaultTextFormat()` to set the default text format. `OlSetDefaultTextFormat()` sets the default text format for any widgets in the application that display text.

If you only want to set the text format for an individual widget, you set its `XtNtextFormat` resource. For example, in the application above you would create the `StaticText` widget as follows:

```
msg_widget = XtVaCreateManagedWidget("msg",
                                       staticTextWidgetClass,
                                       toplevel,
                                       XtNtextFormat, OL_WC_STR_REP,
                                       NULL);
```

For some locales, you may also need to specify a font set for the application. To do this, you add the following line to the resources file:

```
*StaticText.font: font set
```

The following figures show the application with Korean and Japanese text.



Figure 4-7 “Hello World” in Korean



Figure 4-8 “Hello World” in Japanese

Standards

There are three standards that international OLIT follows:

- Internationalized UNIX
- MIT X11 Internationalization Standard
- Internationalized Extension of OPEN LOOK Specification

≡ 4

Standards

Toolkit Functions



This chapter describes functions that are not related to any particular widget.

<i>Initialization and Activation Functions</i>	<i>page 92</i>
<i>Buffer Functions</i>	<i>page 95</i>
<i>Cursor and Pixmap Functions</i>	<i>page 99</i>
<i>Display Functions</i>	<i>page 108</i>
<i>Drag and Drop Functions</i>	<i>page 109</i>
<i>Dynamic Resource Functions</i>	<i>page 140</i>
<i>Error Functions</i>	<i>page 142</i>
<i>Help Function</i>	<i>page 146</i>
<i>Input Focus Functions</i>	<i>page 150</i>
<i>Multiple Visual Functions</i>	<i>page 154</i>
<i>Packed Widget Function</i>	<i>page 156</i>
<i>Pixel Conversion Functions</i>	<i>page 158</i>
<i>Protocol Function</i>	<i>page 160</i>
<i>Regular Expression Functions</i>	<i>page 161</i>
<i>Text Buffer Functions</i>	<i>page 163</i>
<i>Text Buffer Functions for Internationalization</i>	<i>page 176</i>
<i>Text Selection Operations</i>	<i>page 204</i>
<i>Toolkit Resource Functions</i>	<i>page 206</i>
<i>Virtual Event Functions</i>	<i>page 207</i>

Initialization and Activation Functions

Initialization and Activation Functions

OlToolkitInitialize

```
#include <Xol/OpenLook.h>
void OlToolkitInitialize(
    XtPointer NULL);
```

`OlToolkitInitialize()` must be called by each application before any OPEN LOOK widgets are created or other OPEN LOOK routines are used.

The suggested method of initializing an OLIT application is to use `OlToolkitInitialize()` followed by some combination of:

- `XtAppInitialize()`
- `XtToolkitInitialize()`
- `XtCreateApplicationContext()`
- `XtOpenDisplay()` or `XtInitializeDisplay()`
- `XtAppCreateShell()`

or the corresponding variable argument functions.

OlInitialize

Note - The following initialization routine is now obsolete:

```
#include <Xol/OpenLook.h>
void OlInitialize(
    String                shell_name,
    String                application_class,
    XrmOptionDescRec     *options,
    Cardinal              num_options,
    Cardinal              *argc,
    String                argv[]);
```

The arguments to the obsolete `OlInitialize()` routine were similar to the arguments to the X Window `XtInitialize()` routine. Use the `OlToolkitInitialize()` routine instead, but note the differences in arguments in that new version.

OlActivateWidget

```
#include <Xol/OpenLook.h>
Boolean OlActivateWidget(
    Widget      widget,
    OlVirtualName activation_type,
    XtPointer    data);
```

`OlActivateWidget()` programmatically activates a widget in accordance with the supplied activation type (see Chapter 3, “Activation Types”). The precise semantics of activation and activation type are widget-dependent and are described in each widget section. `OlActivateWidget()` returns TRUE if the activation type was accepted by the initially supplied widget, or one of its associated *follower* widgets; see `OlAssociateWidget()`. Otherwise, the function returns FALSE. If the initially supplied widget does not accept the activation type, `OlActivateWidget()` recursively attempts to activate associated *follower* widgets until one of them accepts the supplied activation type.

`OlActivateWidget()` also accepts gadget arguments.

OlAssociateWidget

```
#include <Xol/OpenLook.h>
Boolean OlAssociateWidget(
    Widget      leader,
    Widget      follower,
    Boolean     disable_traversal)
```

`OlAssociateWidget()` associates a widget (the *follower*) with another widget (the *leader*). Associating a widget with a leader widget effectively expands the number of ways the leader widget can be activated since `OlActivateWidget()` automatically activates any follower widgets if the lead widget does not accept the supplied activation type. This routine returns TRUE if the association was successful; otherwise, it returns FALSE. Attempts to create an association-cycle are invalid and produce a warning.

It is typically desirable to prevent keyboard traversal among widgets associated with one another. The *disable_traversal* parameter is a convenient interface for setting the *follower* widget’s `XtNtraversalOn` resource to FALSE.

`OlAssociateWidget()` also accepts gadget arguments.

Initialization and Activation Functions

OlUnassociateWidget

```
#include <Xol/OpenLook.h>
void OlUnassociateWidget(
    Widget follower);
```

`OlUnassociateWidget()` removes a *follower* widget from a previous association with another lead widget. No warning is generated if the supplied widget was not previously associated with another widget.

`OlUnassociateWidget()` also accepts gadget arguments.

Buffer Functions

These functions manipulate a generic “Buffer” object.

AllocateBuffer

```
#include <Xol/buffutil.h>
Buffer *AllocateBuffer(
    int    element_size,
    int    initial_size);
```

AllocateBuffer() allocates a Buffer for elements of the given *element_size*. The *used* member of the Buffer is set to zero and the size member is set to the value of *initial_size*. If *initial_size* is zero, the pointer *p* is set to NULL; otherwise, the amount of space required (*initial_size* × *element_size*) is allocated and the pointer *p* is set to point to this space. The function returns the pointer to the allocated Buffer. It is the responsibility of the caller to free this storage (using FreeBuffer()) when it is no longer needed.

The Buffer structure is defined as follows:

```
typedef struct _Buffer {
    int    size;
    int    used;
    int    esize;
    BufferElement *p;
} Buffer;
```

Buffer Macros

The following macros are provided for use with the Buffer functions.

Table 5-1 Buffer Utilities Macros

Macro	Returns
BufferFilled(<i>buffer</i>)	Indicates whether <i>buffer</i> is filled
BufferLeft(<i>buffer</i>)	Evaluates to the number of unused elements in <i>buffer</i>
BufferEmpty(<i>buffer</i>)	Indicates whether <i>buffer</i> is empty

CopyBuffer

```
#include <Xol/buffutil.h>
Buffer *CopyBuffer(
    Buffer *buffer);
```

CopyBuffer() allocates a new Buffer with the same attributes as the given *buffer* and copies the data associated with the given *buffer* into the new Buffer. A pointer to the newly allocated and initialized Buffer is returned. It is the responsibility of the caller to free this storage (using FreeBuffer()) when it is no longer needed.

FreeBuffer

```
#include <Xol/buffutil.h>
void FreeBuffer(
    Buffer *buffer);
```

FreeBuffer() deallocates (frees) storage associated with the given *buffer* pointer.

GrowBuffer

```
#include <Xol/buffutil.h>
void GrowBuffer(
    Buffer *buffer,
    int increment);
```

GrowBuffer() expands (or compresses) a given *buffer* size by *increment* elements. If the increment is negative, the operation results in a reduction in the size of the Buffer.

InsertIntoBuffer

```
#include <Xol/buffutil.h>
int InsertIntoBuffer(
    Buffer *target,
    Buffer *source,
    int offset);
```

InsertIntoBuffer() inserts the elements stored in the *source* Buffer into the *target* Buffer before the element stored at *offset*. If the *offset* is invalid or if the *source* Buffer is empty, the function returns zero; otherwise, it returns 1 after

completing the insertion. The `GrowBuffer()` function will be used as needed to ensure that the *target* Buffer is large enough to hold the contents of the *source* Buffer.

ReadFileIntoBuffer

```
#include <Xol/buffutil.h>
int ReadFileIntoBuffer(
    FILE      *fp,
    Buffer     *buffer);
```

`ReadFileIntoBuffer()` reads a previously opened file associated with *fp* and adds the characters read to the end of the *buffer*. The `GrowBuffer()` function will be used as needed to ensure that the Buffer is large enough to hold the contents of the file. The function returns when either an end-of-file is detected or a newline character is encountered while reading the file. The function returns EOF if end-of-file is detected and ‘\n’ if a newline character is encountered.

ReadStringIntoBuffer

```
#include <Xol/buffutil.h>
int ReadStringIntoBuffer(
    Buffer     *sp,
    Buffer     *buffer);
```

`ReadStringIntoBuffer()` reads a previously opened Buffer (see “`stropen`” on page 98) associated with *sp* and adds the characters read to the end of *buffer*. The `GrowBuffer()` function will be used as needed to ensure that the Buffer is large enough to hold the data to be copied. The function returns when either a newline character is encountered or an end-of-buffer condition is detected while reading the Buffer associated with *sp*. The function returns a ‘\n’ if it encounters a newline character and EOF if it detects an end-of-buffer condition.

strclose

```
#include <Xol/buffutil.h>
void strclose(
    Buffer *sp);
```

The `strclose()` function closes a string Buffer that was opened using the `stropen()` function. It frees the Buffer allocated by `stropen()`.

Buffer Functions

strgetc

```
#include <Xol/buffutil.h>
int strgetc(
    Buffer *sp);
```

The `strgetc()` function reads the next character stored in the string *sp*. The function returns the next character in the Buffer. When no characters remain, the function returns EOF.

stropen

```
#include <Xol/buffutil.h>
Buffer *stropen(
    char *string);
```

The `stropen()` function allocates a Buffer large enough for *string* and copies *string* into this Buffer. A pointer to the newly allocated Buffer is returned. It is the responsibility of the caller to close this Buffer (using `strclose()`) when it is no longer needed.

See Also

“Text Buffer Functions” on page 163,
“Text Buffer Functions for Internationalization” on page 176.

Cursor and Pixmap Functions

These functions fall into several categories:

- `OlGetxyzCursor` - These are Version 3 functions for obtaining the cursors defined for Drag and Drop functionality. They take a `Widget` argument, unlike the `GetOl`-style cursor functions. The middle part of the name indicates the OPEN LOOK-specified function of the cursor. These `OlGet*` functions are the preferred functions to use in every case, especially if you are not using the standard colormap for the root window of your display.
- `GetOlxyzCursor` - These are not the preferred interface, but are included to provide compatibility with the Version 2 style of functions, taking a `Screen *screen` argument. They are listed separately, without figures. The cursors for the `GetOl*` functions are identical to their `OlGet*` counterparts.
- Other Version 2 functions, also with names of the form `GetOlxyzCursor`, for cursors unrelated to Drag and Drop. Figures for these cursors are in the *OPEN LOOK GUI Functional Specification*.
- Two functions returning gray Pixmap IDs.

Version 3 Cursors



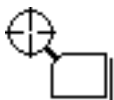
OlGetDataDupeDragCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataDupeDragCursor(
    Widget widget);
```



OlGetDataDupeDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataDupeDropCursor(
    Widget widget);
```



OlGetDataDupeInsertCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataDupeInsertCursor(
    Widget widget);
```

Cursor and Pixmap Functions



OlGetDataDupeNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataDupeNoDropCursor(
    Widget widget);
```



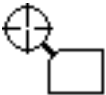
OlGetDataMoveDragCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataMoveDragCursor(
    Widget widget);
```



OlGetDataMoveDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataMoveDropCursor(
    Widget widget);
```



OlGetDataMoveInsertCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataMoveInsertCursor(
    Widget widget);
```



OlGetDataMoveNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDataMoveNoDropCursor(
    Widget widget);
```



OlGetDocCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDocCursor(
    Widget widget);
```



OlGetDocStackCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDocStackCursor(
    Widget widget);
```



OlGetDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDropCursor(
    Widget widget);
```



OlGetDupeDocCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeDocCursor(
    Widget widget);
```



OlGetDupeDocDragCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeDocDragCursor(
    Widget widget);
```



OlGetDupeDocDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeDocDropCursor(
    Widget widget);
```



OlGetDupeDocNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeDocNoDropCursor(
    Widget widget);
```



OlGetDupeStackCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeStackCursor(
    Widget widget);
```



OlGetDupeStackDragCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeStackDragCursor(
    Widget widget);
```

Cursor and Pixmap Functions



OlGetDupeStackDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeStackDropCursor(
    Widget widget);
```



OlGetDupeStackNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetDupeStackNoDropCursor(
    Widget widget);
```



OlGetFolderCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetFolderCursor(
    Widget widget);
```



OlGetFolderStackCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetFolderStackCursor(
    Widget widget);
```



OlGetMoveDocCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveDocCursor(
    Widget widget);
```



OlGetMoveDocDragCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveDocDragCursor(
    Widget widget);
```



OlGetMoveDocDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveDocDropCursor(
    Widget widget);
```




OlGetMoveDocNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveDocNoDropCursor(
    Widget widget);
```



OlGetMoveStackCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveStackCursor(
    Widget widget);
```



OlGetMoveStackDragCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveStackDragCursor(
    Widget widget);
```



OlGetMoveStackDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveStackDropCursor(
    Widget widget);
```



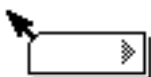
OlGetMoveStackNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetMoveStackNoDropCursor(
    Widget widget);
```



OlGetNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetNoDropCursor(
    Widget widget);
```



OlGetTextDupeDragCursor

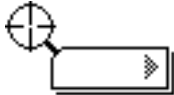
```
#include <Xol/OlCursors.h>
Cursor OlGetTextDupeDragCursor(
    Widget widget);
```

Cursor and Pixmap Functions



OlGetTextDupeDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetTextDupeDropCursor(
    Widget widget);
```



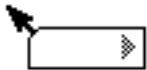
OlGetTextDupeInsertCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetTextDupeInsertCursor(
    Widget widget);
```



OlGetTextDupeNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetTextDupeNoDropCursor(
    Widget widget);
```



OlGetTextMoveDragCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetTextMoveDragCursor(
    Widget widget);
```



OlGetTextMoveDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetTextMoveDropCursor(
    Widget widget);
```



OlGetTextMoveInsertCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetTextMoveInsertCursor(
    Widget widget);
```



OlGetTextMoveNoDropCursor

```
#include <Xol/OlCursors.h>
Cursor OlGetTextMoveNoDropCursor(
    Widget widget);
```

Version 2 Drag and Drop Cursors

All of these version 2 drag and drop cursor functions have similar synopses:

```
#include <Xol/OlCursors.h>
Cursor GetOl<cursor-name>Cursor(
    Screen      *screen);
```

The functions are:

GetOlDataDupeDragCursor()	GetOlFolderCursor()
GetOlDataDupeDropCursor()	GetOlFolderStackCursor()
GetOlDataDupeInsertCursor()	GetOlMoveDocCursor()
GetOlDataDupeNoDropCursor()	GetOlMoveDocDragCursor()
GetOlDataMoveDragCursor()	GetOlMoveDocDropCursor()
GetOlDataMoveDropCursor()	GetOlMoveDocNoDropCursor()
GetOlDataMoveInsertCursor()	GetOlMoveStackCursor()
GetOlDataMoveNoDropCursor()	GetOlMoveStackDragCursor()
GetOlDocCursor()	GetOlMoveStackDropCursor()
GetOlDocStackCursor()	GetOlMoveStackNoDropCursor()
GetOlDropCursor()	GetOlNoDropCursor()
GetOlDupeDocCursor()	GetOlTextDupeDragCursor()
GetOlDupeDocDragCursor()	GetOlTextDupeDropCursor()
GetOlDupeDocDropCursor()	GetOlTextDupeInsertCursor()
GetOlDupeDocNoDropCursor()	GetOlTextDupeNoDropCursor()
GetOlDupeStackCursor()	GetOlTextMoveDragCursor()
GetOlDupeStackDragCursor()	GetOlTextMoveDropCursor()
GetOlDupeStackDropCursor()	GetOlTextMoveInsertCursor()
GetOlDupeStackNoDropCursor()	GetOlTextMoveNoDropCursor()

Other Version 2 Cursors

GetOlBusyCursor

```
#include <Xol/OlCursors.h>
Cursor GetOlBusyCursor(
    Screen      *screen);
```

`GetOlBusyCursor()` obtains the cursor ID for *screen* that complies with the *OPEN LOOK GUI Functional Specification* description of the **Busy** cursor.

Cursor and Pixmap Functions

GetOlDuplicateCursor

```
#include <Xol/OlCursors.h>
Cursor GetOlDuplicateCursor(
    Screen      *screen);
```

GetOlDuplicateCursor() obtains the cursor ID for *screen* that complies with the *OPEN LOOK GUI Functional Specification* description of the **Duplicate** cursor.

GetOlMoveCursor

```
#include <Xol/OlCursors.h>
Cursor GetOlMoveCursor(
    Screen      *screen);
```

GetOlMoveCursor() obtains the cursor ID for *screen* that complies with the *OPEN LOOK GUI Functional Specification* description of the **Move** cursor.

GetOlPanCursor

```
#include <Xol/OlCursors.h>
Cursor GetOlPanCursor(
    Screen      *screen);
```

GetOlPanCursor() obtains the cursor ID for *screen* that complies with the *OPEN LOOK GUI Functional Specification* description of the **Pan** cursor.

GetOlQuestionCursor

```
#include <Xol/OlCursors.h>
Cursor GetOlQuestionCursor(
    Screen      *screen);
```

GetOlQuestionCursor() obtains the cursor ID for *screen* that complies with the *OPEN LOOK GUI Functional Specification* description of the **Question** cursor.

GetOlStandardCursor

```
#include <Xol/OlCursors.h>
Cursor GetOlStandardCursor(
    Screen      *screen);
```

Cursor and Pixmap Functions

`GetOlStandardCursor()` obtains the cursor ID for *screen* that complies with the *OPEN LOOK GUI Functional Specification* description of the **Standard** cursor.

GetOlTargetCursor

```
#include <Xol/OlCursors.h>
Cursor GetOlTargetCursor(
    Screen      *screen);
```

`GetOlTargetCursor()` obtains the cursor ID for *screen* that complies with the *OPEN LOOK GUI Functional Specification* description of the **Target** cursor.

Pixmap Functions

OlGet50PercentGrey

```
#include <Xol/OlCursors.h>
Pixmap OlGet50PercentGrey(
    Screen      *screen);
```

`OlGet50PercentGrey()` obtains the ID of a 50% grey Pixmap for *screen*.

OlGet75PercentGrey

```
#include <Xol/OlCursors.h>
Pixmap OlGet75PercentGrey(
    Screen      *screen);
```

`OlGet75PercentGrey()` obtains the ID of a 75% grey Pixmap for *screen*.

Return Values

Each cursor function returns a cursor ID. Each Pixmap function returns a Pixmap.

See Also

“Drag and Drop Functions” on page 109.

Display Functions
Display Functions

OlUpdateDisplay

```
#include <Xol/OpenLook.h>
void OlUpdateDisplay(
    Widget      w);
```

`OlUpdateDisplay()` processes all pending exposure events so that the appearance of a given widget can be updated immediately. Normally, an operation is accomplished by a set of callback functions. If one of the callback functions performs a time-consuming action, it is possible that some portion of an application window will not be redrawn right away after an `XtSetValues()` call. This is because normal exposure processing does not occur until all callback functions have been invoked. This situation can be resolved by calling this function before starting a time-consuming action.

Example

```
extern Widget  status_area;      /* a staticText widget */
void fooCB(
    Widget      w,
    XtPointer   client_data,
    XtPointer   call_data);
{
    ...
    Arg        args[5];
        /* display the status in the footer area */
        /* before the actual operation */
    XtSetArg(args[0], XtNstring,
        "Start the operation, please wait ...");
    XtSetValues(status_area, args, 1);
        /* show the status in the footer area right away */
    OlUpdateDisplay(status_area);
        /* now we can start the actual operation */
    ...
}
```

See Also

`XtSetValues()` in *Xt Intrinsic Reference Manual*.

Drag and Drop Functions

Drag and Drop is a direct-manipulation data transfer operation with the following steps:

1. The user “picks up” an object (a graphical representation of data inside a client application) by pressing the SELECT mouse button on the object.
2. The user “drags” the object across the display with the SELECT mouse button pressed.
3. The user “drops” the object over an eligible drop site by releasing the SELECT mouse button over the drop site.

An example of a drag and drop operation is picking up a file from a file manager and dropping it on a trash can icon to delete the file. The terminology associated with the drag and drop operation is described below in detail.

Drop Rectangle

A drop rectangle is a rectangular area of the screen selected by some application to be eligible for drops. It is the building block of a drop site. There may or may not be any graphical feedback associated with a drop rectangle.

Drop Site

A drop site is a list of possibly overlapping drop rectangles. Drop sites need to be registered with the toolkit before they can participate in the Drag and Drop “protocol.” A drop site may be registered with the toolkit by calling either of the functions `OldNdRegisterWidgetDropSite()` or `OldNdRegisterWindowDropSite()`.

Drag and Drop Functions

Drop Target

A drop target is a visible receptacle for a drop. It is marked by one of the glyphs (empty or full versions) in the following figure:



Figure 5-1 Drop Target Glyphs

See “DropTarget Widget” on page 266 for details on creating a drop target.

Owner of Drop Site

The owner of a drop site is the window or widget that registered the drop site with the toolkit. The owner of a drop site is notified (see Preview Message Notify Procedure) when the user moves the pointer over one of its drop rectangles during a drag operation. Drop sites die when the owner dies.

Do not confuse the widget owner of a drop site with the widget owner of an X11 selection (see “Drop and Data Transfer” on page 112).

Preview Message Notify Procedure

When a drop site is registered with the toolkit, the owner of the drop site may provide Preview Hints (see “OIDnDSitePreviewHints” on page 117) and a Preview Message Notify Procedure (see “OIDnDPMNotifyProc” on page 119). During the drag operation, when the user moves the pointer over a drop site, the drop site’s Preview Message Notify Procedure may be invoked in a manner consistent with the Preview Hints.

Trigger Message Notify Procedure

When a drop site is registered with the toolkit, the owner of the drop site may provide a Trigger Message Notify Procedure (see “OIDnDTMNotifyProc” on page 121). When a drop occurs on the drop site, the toolkit informs the drop site of the drop by invoking the drop site’s Trigger Message Notify Procedure.

Drag and Drop Functions

A brief description follows of the interaction between a source and destination during a Drag and Drop operation.

Setup

Source

The source may or may not be registered with the toolkit. The source must arrange to be notified of mouse SELECT events in whatever manner it deems appropriate.

Destination

The destination must register its drop site(s) with the toolkit by calling `OldnDRegisterWidgetDropSite()` if the destination is a widget, or `OldnDRegisterWindowDropSite()` if the destination is a window. Registering the drop site may involve providing the toolkit with a set of Preview Hints (see “`OldnDSitePreviewHints`” on page 117), a Preview Message Notify Procedure (see “`OldnDPMNotifyProc`” on page 119), and a Trigger Message Notify Procedure (see “`OldnDTMNotifyProc`” on page 121).

Begin Drag

The user “picks” an object by pressing the mouse SELECT button over an object. The owner of the object is deemed to be the source. With the mouse SELECT button pressed, the user moves the pointer over the display, “dragging” the picked object away from the source in search of a destination.

Source

Once the source is notified of a mouse SELECT event and the source is in an internal state consistent with the beginning of a Drag and Drop operation, the source should take the following steps:

1. Allocate an X11 selection atom using `OldnDAllocateTransientAtom()`.
2. Grab the pointer by calling `OlGrabDragPointer()`.

Drag and Drop Functions

3. Begin the Drag and Drop operation by calling `OldnDDragAndDrop()`. `OldnDDragAndDrop()` does not return until a drop occurs or the Drag and Drop operation is aborted. `OldnDDragAndDrop()` returns TRUE if a successful “drop” occurs. Do not confuse a successful drop with a successful transfer of data associated with a dropped object.

Destination

The Destination does not exist at this point.

Preview And Animate

Each time the user moves the pointer over a drop site during the drag operation, the source and the drop site under the pointer may be notified, depending on whether they have registered suitable notify procedures.

Source

When calling `OldnDDragAndDrop()` (see “Begin Drag” on page 111), the source may provide a Preview and Animate Callback Procedure (see “`OldnDPreviewAnimateCbP`” on page 118). The toolkit calls the source’s Preview and Animate Callback Procedure as the cursor moves over each potential drop site, giving the source the opportunity to change the cursor appearance as needed to provide the appropriate user feedback.

Drop Site Under The Pointer

As the cursor moves over each potential drop site, the toolkit will also call the Preview Message Notify Procedure of the drop site under the pointer, as appropriate for the drop site’s Preview Hints. This affords the drop site the opportunity to change its appearance to indicate its ability (or inability) to actually receive the drop.

Drop and Data Transfer

A “drop” occurs when the user releases the mouse SELECT button over a drop site. The drop site over which the drop occurs is deemed to be the destination.

Source

`OldnDDragAndDrop()` (see page 124) returns TRUE if a successful drop occurs. It has three arguments used to return values:

```
Window      *window,  
Position    *x, *y
```

These arguments contain the values of the drop window ID and the x- and y-coordinates of the drop. At this point the source may:

1. Obtain ownership of the X11 selection atom allocated during the Begin Drag step. This is done by calling `OldnDnDownSelection()`. See page 126.
2. Inform the toolkit of the ID of the X11 selection atom by calling `OldnDDeliverTriggerMessage()`. See page 123.
3. Release the pointer by calling `OlUngrabDragPointer()`. See page 128.

Destination

When `OldnDDeliverTriggerMessage()` is called, the toolkit invokes the Trigger Message Notify Procedure of the destination drop site, passing along the ID of the X11 selection atom obtained from the source. If the destination is interested in the drop, it may call `XGetSelection()` or `XtGetSelectionValue()` to obtain the contents of the selection. Refer to Section 10.2 of the *Xt Intrinsic Reference Manual* for details on obtaining the contents of a selection.

Closing Handshake

Source

The source may provide an `OldnDTransactionStateCallback()` function as an argument to `OldnDnDownSelection()` (see “Drop and Data Transfer” on page 112). This function is invoked by the toolkit with its `OldnDTransactionState` argument set to an appropriate type, when the destination invokes any of the following functions:

- `OldnDBeginSelectionTransaction()`
- `OldnDEndSelectionTransaction()`
- `OldnDErrorDuringSelectionTransaction()`
- `OldnDDragNDropDone()`

Drag and Drop Functions

Destination

If the toolkit invokes the Trigger Message Notify Procedure with the *send_done* argument set to TRUE, the destination is expected to call `OldDnDBeginSelectionTransaction()` at the beginning of the selection transfer and call `OldDnDEndSelectionTransaction()` at the end of the selection transfer. The destination may invoke `OldDnDErrorDuringSelectionTransaction()` to indicate an error during selection transfer.

Cleanup

Source

The source releases the selection with `OldDnDDisownSelection()`, and frees the X11 selection atom it originally allocated (see “Setup” on page 111) with `OldDnDFreeTransientAtom()`. This is typically done from within the source’s Transaction State Callback when it is invoked with an `OldDnDTransactionState` value of `OldDnDTransactionDone`.

Destination

The destination calls `OldDnDDragNDropDone()`. If the source specified a Transaction State Callback to `OldDnDDownSelection()`, the toolkit will now call it with an `OldDnDTransactionState` value of `OldDnDTransactionDone`.

Common Arguments

The following arguments are used in most drag and drop functions. When an argument to a specific function differs in interpretation from what is listed here, it is noted for that individual function.

<i>client_data</i>	Application-defined data
<i>dropsiteid</i>	ID of the drop site over which the drop occurred
<i>detail</i>	An XEvent type: either <code>EnterNotify</code> , <code>MotionNotify</code> , or <code>LeaveNotify</code>
<i>num_sites</i>	The number of drop rectangles defining the drop site
<i>operation</i>	Either <code>OldDnDTriggerCopyOp</code> or <code>OldDnDTriggerMoveOp</code>

Drag and Drop Functions

<i>pmnotify</i>	A pointer to a Preview Notify Procedure, of type <code>OldnDPMNotifyProc()</code> . The Preview Notify Procedure is given by the destination when it registers its drop site(s). It is called when the cursor passes over it following the start of a drag, subject to the conditions given in the <i>preview_hints</i> .
<i>preview_hints</i>	An enumerated data type <code>OldnDnSitePreviewHints</code> ; possible values are listed on page 117.
<i>root</i>	The root window.
<i>rootx</i>	The horizontal (x-) coordinate at which the drop occurred, relative to the root window.
<i>rooty</i>	The vertical (y-) coordinate at which the drop occurred, relative to the root window.
<i>selection</i>	The X11 selection atom actually used to transfer the data.
<i>site_rects</i>	A list of <code>OldnDnSiteRect</code> structures, which defines the drop rectangles, in the coordinate system of the widget that registers the drop site.
<i>timestamp</i>	The current server time. Often, this argument indicates the time of an event such as the cursor entering a drop site. In these cases it will be explicitly noted.
<i>tmnotify</i>	A pointer to a Trigger Message Notify Procedure, of type <code>OldnDTMNotifyProc()</code> .
<i>widget</i>	The widget associated with the owner of a drop site.
<i>window</i>	The window associated with the owner of a drop site.

Function Groups

The Drag and Drop API contains the following function groups:

Drop Site Manipulation Functions

<code>OldnDRegisterWidgetDropSite()</code>	Register a widget-based drop site
<code>OldnDRegisterWindowDropSite()</code>	Register a window-based drop site
<code>OldnDSetDropSiteInterest()</code>	Activate or inactivate a drop site
<code>OldnDSetInterestInWidgetHier()</code>	Activate or inactivate drop sites in a widget hierarchy

Drag and Drop Functions

<code>OldDnUpdateDropSiteGeometry()</code>	Update a drop site's geometry
<code>OldDnChangeDropSitePreviewHints()</code>	Update a drop site's preview hints
<code>OldDnDestroyDropSite()</code>	Delete an existing drop site
<code>OldDnQueryDropSiteInfo()</code>	Query a drop site's geometry
<code>OldDnGetWindowOfDropSite()</code>	Get a window associated with a drop site
<code>OldDnGetWidgetOfDropSite()</code>	Get a widget associated with a drop site
<code>OldDnGetDropSitesOfWidget()</code>	Get the drop sites for a widget
<code>OldDnGetDropSitesOfWindow()</code>	Get the drop sites for a window

Message Functions

`OldDnDeliverPreviewMessage()`
`OldDnDeliverTriggerMessage()`
`OldDnPreviewAndAnimate()`

Handshake Functions

`OldDnBeginSelectionTransaction()`
`OldDnEndSelectionTransaction()`
`OldDnErrorDuringSelectionTransaction()`

Selection Functions

`OldDnAllocTransientAtom()`
`OldDnDisownSelection()`
`OldDnFreeTransientAtom()`
`OldDnDownSelection()`
`OldDnDownSelectionIncremental()`
`OldDnGetCurrentSelectionsForWidget()`

General Purpose Functions

`OlGrabDragPointer()`
`OldDnDragAndDrop()`
`OlUngrabDragPointer()`
`OldDnDragNDropDone()`
`OldDnWidgetConfiguredInHier()`

Data Structures

The following data structures are used by several Drag and Drop functions.

OldnDDropSiteID

```
#include <Xol/OldnDVCX.h>
typedef struct oldnd_drop_site *OldnDDropSiteID;
```

`OldnDDropSiteID` is an opaque reference to a particular instance of a drop site.

OldnDSiteRect

```
#include <Xol/OldnDVCX.h>
typedef XRectangle OldnDSiteRect, *OldnDSiteRectPtr;
```

The `OldnDSiteRect` structure describes the drop site rectangle. Drop sites can include multiple rectangles.

OldnDSitePreviewHints

```
#include <Xol/OldnDVCX.h>
typedef enum oldnd_site_preview_hints {
    OldnDSitePreviewNone,
    OldnDSitePreviewEnterLeave = (1 << 0),
    OldnDSitePreviewMotion = (1 << 1),
    OldnDSitePreviewBoth = (OldnDSitePreviewEnterLeave |
                             OldnDSitePreviewMotion),
    OldnDSitePreviewDefaultSite = (1 << 2),
    OldnDSitePreviewForwarded = (1 << 3),
    OldnDSitePreviewInsensitive = (1 << 4)
} OldnDSitePreviewHints;
```

The `OldnDSitePreviewHints` enumerated type specifies the conditions under which the drop site is interested in receiving notification through its preview callback:

<code>OldnDSitePreviewNone</code>	The drop site does no previewing; its callback will not be invoked.
<code>OldnDSitePreviewEnterLeave</code>	The drop site Preview Message Notify Callback will be invoked for Enter/Leave events.

Drag and Drop Functions

<code>OldnDnSitePreviewMotion</code>	The drop site Preview Message Notify Callback will be invoked for Motion events.
<code>OldnDnSitePreviewBoth</code>	The drop site Preview Message Notify Callback will be invoked for Enter/Leave and Motion events.
<code>OldnDnSitePreviewDefaultSite</code>	The drop site is the default site for drop site forwarding on this application shell. A default drop site is the site nominated to receive drops forwarded by the window manager decorations or icons.
<code>OldnDnSitePreviewForwarded</code>	The drop site is acting as a “proxy” on behalf of some other object or client.
<code>OldnDnSitePreviewInsensitive</code>	The drop site is currently “insensitive”; this information will be passed to the source’s Preview Animate Callback so it can animate the cursor appropriately as it passes over the drop site.

Notify Procedure Prototypes

OldnDnPreviewAnimateCbP

```
#include <Xol/OldnDVCX.h>
typedef void (*OldnDnPreviewAnimateCbP)(
    Widget      widget,
    int         eventcode,
    Time        timestamp,
    Boolean      insensitive,
    XtPointer    client_data);
```

eventcode Event code: LeaveNotify, EnterNotify, or MotionNotify.

insensitive TRUE means the drop site under the cursor is insensitive.

client_data Application-defined data passed to `OldnDDragAndDrop()` when the drag and drop operation began.

`OldnDnPreviewAnimateCbP` is the function prototype for the source’s Preview and Animate Callback.

OldnDPMNotifyProc

```
#include <Xol/OldnDVCX.h>
typedef void (*OldnDPMNotifyProc)(
    Widget          widget,
    Window          window,
    Position        root_x,
    Position        root_y,
    int             detail,
    Time            timestamp,
    OldnDDropSiteID drop_site,
    Boolean         forwarded,
    XtPointer       client_data);
```

- root_x* The root-relative x-coordinate of the preview “event”
- root_y* The root-relative y-coordinate of the preview “event”
- timestamp* The time of the preview “event”
- drop_site* The ID of the drop site on which the preview occurred
- forwarded* TRUE means the drop has been forwarded to this target from another drop site
- client_data* Application-defined data passed to the toolkit when the drop site was registered.

The other arguments to this function are described in “Common Arguments” on page 114.

OldnDPMNotifyProc is the function prototype for the drop site Preview Message Notify Procedure. This procedure is associated with a particular drop site at the time of registration of the drop site with the toolkit. It may be invoked by the toolkit during a “drag” operation when the pointer enters, leaves, or moves across the drop site in a manner consistent with the Preview Hints (see “*OldnDSitePreviewHints*” on page 117) of the drop site.

OldnDProtocolActionCbP

```
#include <Xol/OldnDVCX.h>
typedef void (*OldnDProtocolActionCbP) (
    Widget          widget,
    Atom            selection,
    OldnDProtocolAction protocol_action,
    Boolean         flag,
    XtPointer       client_data);
```

Drag and Drop Functions

<i>protocol_action</i>	An enumerated type indicating the protocol action that has occurred.
<i>success</i>	Success/failure flag; if TRUE, the protocol notification was successfully received.

The other arguments to this function are described in “Common Arguments” on page 114.

The `OldNdProtocolAction` enumerated type is defined as:

```
typedef enum oldnd_protocol_action {
    OldNdSelectionTransactionBegins,
    OldNdSelectionTransactionEnds,
    OldNdSelectionTransactionError,
    OldNdDragNDropTransactionDone
} OldNdProtocolAction;
```

`OldNdProtocolActionCbP` the function prototype for a Protocol Action Callback Procedure. This callback will be invoked by the toolkit to inform the caller of the success or failure of these selection transaction handshake functions. It is supplied by the requester of a selection as an argument to the selection transaction handshake functions (see “`OldNdBeginSelectionTransaction`” on page 128, “`OldNdEndSelectionTransaction`” on page 130, “`OldNdErrorDuringSelectionTransaction`” on page 131, and “`OldNdDragNDropDone`” on page 129).

OldNdTransactionStateCallback

```
#include <Xol/OldNdVCX.h>
typedef void (*OldNdTransactionStateCallback) (
    Widget          widget,
    Atom            selection,
    OldNdTransactionState state,
    Time           timestamp,
    XtPointer       client_data);
```

<i>widget</i>	The selection holder.
<i>selection</i>	The selection atom
<i>state</i>	The protocol event that has occurred
<i>timestamp</i>	When the event occurred.
<i>client_data</i>	Application-defined data passed to <code>OldNdDownSelection()</code> or <code>OldNdDownSelectionIncremental()</code> .

Drag and Drop Functions

The `OldnDTransactionState` enumerated type is defined as:

```
typedef enum oldnd_transaction_state {
    OldnDTransactionBegins,
    OldnDTransactionEnds,
    OldnDTransactionDone,
    OldnDTransactionRequestorError,
    OldnDTransactionRequestorWindowDeath,
    OldnDTransactionTimeout,
} OldnDTransactionState;
```

`OldnDDownSelection()` and `OldnDDownSelectionIncremental()` have an `OldnDTransactionStateCallback()` function pointer as one of their parameters. This function is invoked as a result of drag and drop protocol events during the drag and drop selection transaction.

The callback is invoked with the following `OldnDTransactionState` values when the requester of the selection (the destination drop site) invokes the following functions:

Function Invoked By Destination	OldnDTransactionState Value
<code>OldnDBeginSelectionTransaction()</code>	<code>OldnDTransactionBegins</code>
<code>OldnDEndSelectionTransaction()</code>	<code>OldnDTransactionEnds</code>
<code>OldnDDragNDropDone()</code>	<code>OldnDTransactionDone</code>
<code>OldnDErrorDuringSelectionTransaction()</code>	<code>OldnDTransactionRequestorError</code>

If the requesting client is lost during the selection transfer because its window dies, the state callback will be invoked with a state value of `OldnDTransactionRequestorWindowDeath`.

OldnDTMNotifyProc

```
#include <Xol/OldnDVCX.h>
typedef void (*OldnDTMNotifyProc)(
    Widget          widget,
    Window          window,
    Position        root_x,
    Position        root_y,
    Atom            selection,
    Time            timestamp,
    OldnDDropSiteID dropsite,
    OldnDTriggerOperation operation,
```

Drag and Drop Functions

	Boolean	<i>send_done</i> ,
	Boolean	<i>forwarded</i> ,
	XtPointer	<i>client_data</i>);
<i>root_x</i>	The root-relative x-coordinate at which the drop occurred	
<i>root_y</i>	The root-relative y-coordinate at which the drop occurred	
<i>timestamp</i>	The timestamp of the trigger message	
<i>dropsite</i>	The ID of the drop site on which the drop occurred	
<i>send_done</i>	If TRUE, the selection holder expects to be notified at the end of the selection transaction that it has been completed and that no further transactions associated with this drop will occur. This notification is achieved by calling <code>OldnDDragNDropDone()</code> when the selection transaction is completed successfully.	
<i>forwarded</i>	If TRUE, the site rectangle dropped upon is a forwarded site rectangle associated by a third party (such as a window manager) with the default drop site of a top level window.	
<i>client_data</i>	Application-defined data passed to the toolkit when the drop site was registered.	

The other arguments to this function are described in “Common Arguments” on page 114.

`OldnDTMNotifyProc` is the function prototype for the drop operation Trigger Message Notify Procedure. This notify procedure, associated with a particular drop site at registration, is invoked when a drop operation occurs on its associated drop site.

Source Functions

OldnDAllocTransientAtom

```
#include <Xol/OldnDVCX.h>
Atom OldnDAllocTransientAtom(
    Widget widget);
```

widget The ID of the widget that will own the transient atom returned by this call.

`OldnDAllocTransientAtom()` allocates a reusable “transient” atom suitable for use in a drag and drop selection transaction for this widget.

OldnDClearDragState

```
#include <Xol/OldnDVCX.h>
void OldnDClearDragState(
    Widget widget);
```

widget The widget ID of the selection holder

`OldnDClearDragState()` is called upon completion of the previewing phase of a drag and drop gesture to clear internal state within the drag and drop system. This function is called implicitly by `OldnDDragAndDrop()` and therefore is not ordinarily called directly by the OLIT programmer.

OldnDDeliverPreviewMessage

```
#include <Xol/OldnDVCX.h>
Boolean OldnDDeliverPreviewMessage(
    Widget      widget,
    Window      root,
    Position    rootx,
    Position    rooty,
    Time        timestamp);
```

widget The widget ID of the selection owner.

The other arguments to this function are described in “Common Arguments” on page 114.

`OldnDDeliverPreviewMessage()` attempts to deliver Enter, Leave, and Motion events to any drop sites currently under the *(rootx,rooty)* position on the root window specified. `OldnDDeliverPreviewMessage()` returns TRUE if it finds a drop site to deliver an event to; otherwise, it returns FALSE.

OldnDDeliverTriggerMessage

```
#include <Xol/OldnDVCX.h>
Boolean OldnDDeliverTriggerMessage(
    Widget      widget,
    Window      root,
    Position    rootx,
    Position    rooty,
    Atom        selection,
    OldnDTriggerOperation operation,
    Time        timestamp);
```

Drag and Drop Functions

widget The widget ID of the selection holder

The other arguments to this function are described in “Common Arguments” on page 114.

`OldnDDeliverTriggerMessage()` is called by the dragging client to deliver a trigger message to a target drop site on the root window at the coordinates specified.

The calling client is responsible for establishing a timeout period. If the drop target doesn't send selection conversion requests during this period, it should take appropriate action. `OldnDDeliverTriggerMessage()` returns TRUE if it finds a drop site to dispatch a trigger message to at the root (x,y); otherwise, it returns FALSE.

OldnDDisownSelection

```
#include <Xol/OldnDVCX.h>
void OldnDDisownSelection(
    Widget widget,
    Atom selection,
    Time timestamp);
```

`OldnDDisownSelection()` is identical in semantics to the Xt function `XtDisownSelection()`. The source widget should call this function to relinquish ownership of the selection when the drag and drop operation has been completed.

OldnDDragAndDrop

```
#include <Xol/OldnDVCX.h>
Boolean OldnDDragAndDrop(
    Widget widget,
    Window *window,
    Position *x,
    Position *y,
    OldnDDragDropInfoPtr drop_info,
    OldnDPreviewAnimateCbP proc,
    XtPointer client_data);
```

widget The ID of the source widget initiating the drag and drop operation

window Returns the ID of the window containing the cursor (pointer)

x The x-coordinate of the cursor relative to the containing window

Drag and Drop Functions

- y* The y-coordinate of the cursor relative to the containing window
- drop_info* A pointer to a structure of type `OldnDDragNDropInfo` containing information about the location of the drop, root-relative:
- ```
typedef struct _ol_dnd_root_info {
 Window root_window;
 Position root_x;
 Position root_y;
 Time drop_timestamp;
} OldnDDragNDropInfo, *OldnDDragNDropInfoPtr;
```
- proc* The animate function that is called when the cursor enters a drop site.
- client\_data* Application-defined data to be passed to the animate callback.

`OldnDDragAndDrop()` provides a simple interface for processing the mouse and keyboard events during a drop and drop operation. Before calling this function, you should call `OlGrabDragPointer()` or `XGrabPointer()` to effectively grab pointer events.

`OldnDDragAndDrop()` issues an `XGrabKeyboard()` to obtain keystrokes during the drag operation. It then inserts a raw event handler on the widget specified for the pointer and key events and initializes the drag and drop system with `OldnDInitializeDragState()`. Then it proceeds to process the event stream, delivering preview messages where appropriate via `OldnDDeliverPreviewMessage()` until the drag completes or is aborted. The function returns the x, y location and the window that the pointer was in when the operation completed. It also returns the necessary root information.

### *OldnDFreeTransientAtom*

```
#include <Xol/OldnDVCX.h>
void OldnDFreeTransientAtom(
 Widget widget,
 Atom transient);
```

- widget* The widget with which the transient atom was associated.
- transient* The selection atom.

`OldnDFreeTransientAtom()` frees the transient atom obtained from `OldnDAllocTransientAtom()`.

### *OldnDInitializeDragState*

```
#include <Xol/OldnDVCX.h>
Boolean OldnDInitializeDragState(
 Widget widget);
```

*widget*    The ID of the source widget initiating the drag and drop operation

`OldnDInitializeDragState()` is called prior to commencing delivery of preview messages to cause the drag and drop system to download drop site previewing information from the OPEN LOOK Window Manager. It returns TRUE if the download was successful. Otherwise, it returns FALSE. This function is called implicitly by `OldnDDragAndDrop()` and therefore is not ordinarily called directly by the OLIT programmer.

### *OldnDDownSelection*

```
#include <Xol/OldnDVCX.h>
Boolean OldnDDownSelection(
 Widget widget,
 Atom selection,
 Time timestamp,
 XtConvertSelectionProc convert_proc,
 XtLoseSelectionProc lose_proc,
 XtSelectionDoneProc done_proc,
 OldnDTransactionStateCallback state_cb,
 XtPointer client_data);
```

`OldnDDownSelection()` is identical in semantics to the Xt function `XtOwnSelection()` except for the additional parameters *state\_cb* and *client\_data* (see “`OldnDTransactionStateCallback`” on page 120).

### *OldnDDownSelectionIncremental*

```
#include <Xol/OldnDVCX.h>
Boolean OldnDDownSelectionIncremental(
 Widget widget,
 Atom selection,
 Time timestamp,
 XtConvertSelectionIncrProc convert_incr_proc,
 XtLoseSelectionIncrProc lose_incr_proc,
 XtSelectionDoneIncrProc done_incr_proc,
```



## Drag and Drop Functions

```
XtCancelConvertSelectionProc cancel_proc,
XtPointer client_data,
OldnDTransactionStateCallback state_cb);
```

`OldnDDownSelectionIncremental()` function is identical in semantics to the Xt function `XtOwnSelectionIncremental()` except for the additional parameter *state\_cb* (see “OldnDTransactionStateCallback” on page 120).

### OldnDPreviewAndAnimate

```
#include <Xol/OldnDVCX.h>
Boolean OldnDPreviewAnimate(
 Widget widget,
 Window root,
 Position rootx,
 Position rooty,
 Time timestamp,
 OldnDPreviewAnimateCbP animate_proc,
 XtPointer client_data);
```

*animate\_proc*    The animate callback function.

This function is called implicitly by `OldnDDragAndDrop()` and therefore is not ordinarily called by the OLIT programmer directly.

### OlGrabDragPointer

```
#include <Xol/OpenLook.h>
void OlGrabDragPointer(
 Widget w,
 Cursor cursor,
 Window confine_to_window);
```

*w*                    The ID of the source widget initiating the drag and drop operation.

*cursor*                The cursor to be displayed.

*confine\_to\_window*    Specifies the window to confine the drag pointer to, or None. None is typically what is desired for a drag and drop operation.

`OlGrabDragPointer()` effects an active grab of the mouse pointer. This function is normally called after a mouse drag operation has begun and prior to calling the `OldnDDragAndDrop()` procedure, which is used to monitor the drag operation.

## Drag and Drop Functions

`OlGrabDragPointer()` does not return until it has successfully grabbed the drag pointer. If another widget in this client application has already grabbed the pointer, calling this function overrides any such previous grab. If another client application has already grabbed the pointer, this function blocks until the other client ungrabs the pointer and this client subsequently grabs the pointer.

### *OlUngrabDragPointer*

```
#include <Xol/OpenLook.h>
void OlUngrabDragPointer(
 Widget w);
```

`OlUngrabDragPointer()` relinquishes the active pointer grab that was initiated by the `OlGrabDragPointer()` procedure. It simply ungrabs the pointer.

For `OlUngrabDragPointer()` to succeed, the widget passed to it must be on the same display as the widget used to grab the pointer.

## Destination Functions

### *OldnDBeginSelectionTransaction*

```
#include <Xol/OldnDVCX.h>
void OldnDBeginSelectionTransaction(
 Widget widget,
 Atom selection,
 Time timestamp,
 OldnDProtocolActionCbP proc,
 XtPointer client_data);
```

- widget*     The requesting widget or the drop site owner.
- selection*   The selection atom passed in the trigger notify function.
- timestamp*   The server timestamp for the current time.
- proc*        The callback to inform the requester whether the selection owner has successfully received the begin notification. When this callback is invoked, the *protocol\_action* argument (see “OldnDProtocolActionCbP” on page 119) is set to `OldnDSelectionTransactionBegins`.
- client\_data*   Application-defined data to be passed to *proc*.

---

## Drag and Drop Functions

`OldnDBeginSelectionTransaction()` is used in conjunction with the `OldnDEndSelectionTransaction()` function to provide a positive handshake indicating a selection transaction. It invokes the selection holder's transaction state callback (specified by the `OldnDownSelection()` and `OldnDownSelectionIncremental()` functions) with a transaction state parameter value of `OldnDTransactionBegins`.

### *OldnDChangeDropSitePreviewHints*

```
#include <Xol/OldnDVCX.h>
Boolean OldnDChangeDropSitePreviewHints(
 OldnDDropSiteID dropsiteid;
 OldnDSitePreviewHints preview_hints);
```

The arguments to this function are described in “Common Arguments” on page 114.

`OldnDUpdateSitePreviewHints()` updates a drop site's preview hints. During the lifetime of a drop site it may be necessary to alter the nature of its previewing interest. Use `OldnDUpdateSitePreviewHints()` to overwrite the existing preview hints for a drop site and update the drop site interest list appropriately.

### *OldnDDestroyDropSite*

```
#include <Xol/OldnDVCX.h>
void OldnDDestroyDropSite(
 OldnDDropSiteID dropsiteid);
```

*dropsiteid* The ID of the drop site

`OldnDDestroyDropSite()` explicitly destroys a drop site. When a drop site's widget or window is destroyed, all drop sites associated with that widget or window are automatically destroyed.

### *OldnDDragNDropDone*

```
#include <Xol/OldnDVCX.h>
void OldnDDragNDropDone(
 Widget widget,
 Atom selection,
```

## Drag and Drop Functions

```

Time timestamp,
OldDnDProtocolActionCbP proc,
XtPointer client_data);

```

*proc*            The callback to inform the requester whether the selection owner has successfully received the done notification. When it is invoked, the *protocol\_action* argument (see “OldDnDProtocolActionCbP” on page 119) is set to OldDnDDragNDropTransactionDone.

*client\_data*    Application-defined data to be passed to *proc*.

The other arguments to this function are described in “Common Arguments” on page 114.

OldDnDDragNDropDone() is called to inform the source (selection holder) of the completion of the drag and drop operation; this is a notification to the source that it may clean up any state associated with the selection atom, as described in “Cleanup” on page 114. It invokes the source’s transaction state callback (registered with OldDnDDownSelection() or OldDnDDownSelectionIncremental()) with a transaction state parameter value of OldDnDTransactionDone.

### OldDnDEndSelectionTransaction

```

#include <Xol/OldDnDVCX.h>

void OldDnDEndSelectionTransaction(
 Widget widget,
 Atom selection,
 Time timestamp,
 OldDnDProtocolActionCbP proc,
 XtPointer client_data);

```

*proc*            The callback to inform the requester whether the selection owner has successfully received the end notification. When it is invoked, the *protocol\_action* (see “OldDnDProtocolActionCbP” on page 119) argument is set to OldDnDSelectionTransactionEnds.

*client\_data*    Application-defined data to be passed to *proc*.

The other arguments to this function are described in “Common Arguments” on page 114.

---

## Drag and Drop Functions

`OldDnEndSelectionTransaction()` provides a positive handshake between the selection requester and holder.

It invokes the selection holder's transaction state callback (registered with the `OldDnDownSelection()` and `OldDnDownSelectionIncremental()` functions) with a transaction state parameter value of `OldDnDTransactionEnds`.

### *OldDnErrorDuringSelectionTransaction*

```
#include <Xol/OldnDVCX.h>

void OldDnErrorDuringSelectionTransaction(
 Widget widget,
 Atom selection,
 Time timestamp,
 OldDnDProtocolActionCbP proc,
 XtPointer client_data);
```

*proc*            The callback to inform the requester whether the selection owner has successfully received the error notification. When it is invoked, the *protocol\_action* (see `OldDnDProtocolActionCbP`) argument is set to `OldDnDSelectionTransactionError`.

*client\_data*    Application-defined data to be passed to *proc*.

The other arguments to this function are described in “Common Arguments” on page 114.

`OldDnErrorDuringSelectionTransaction()` can be called at any time during the selection transfer by the requester to inform the selection holder that there is an error. The subsequent behavior of the holder is undefined by this protocol. `OldDnErrorDuringSelectionTransaction()` invokes the selection holder's transaction state callback (registered with `OldDnDownSelection()` or `OldDnDownSelectionIncremental()`) with a transaction state parameter value of `OldDnDTransactionRequestorError`.

### *OldDnDGetCurrentSelectionsForWidget*

```
#include <Xol/OldnDVCX.h>

Boolean OldDnDGetCurrentSelectionsForWidget(
 Widget widget,
 Atom **atoms_return,
 Cardinal *num_sites_return);
```

## Drag and Drop Functions

|                         |                                                                         |
|-------------------------|-------------------------------------------------------------------------|
| <i>widget</i>           | The ID of the widget being investigated.                                |
| <i>atoms_return</i>     | Points to an array of atoms currently held as selections by the widget. |
| <i>num_sites_return</i> | Points to a variable containing the number of atoms returned.           |

`OldDnDGetCurrentSelectionsForWidget()` returns a list of atoms currently held as drag and drop selections for the specified widget. If `OldDnDGetCurrentSelectionsForWidget()` finds any, it returns TRUE; otherwise, it returns FALSE.

The caller must call `XtFree()` on the pointer returned in the *atoms\_return* parameter to free the storage allocated when it is no longer required.

### *OldDnDGetDropSitesOfWidget*

```
#include <Xol/OldnDVCX.h>
OldnDDropSiteID *OldDnDGetDropSitesOfWidget(
 Widget widget,
 Cardinal *num_sites_return);
```

|                         |                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------|
| <i>widget</i>           | The widget associated with the owner of the drop site.                                |
| <i>num_sites_return</i> | A pointer to a variable into which the function will return the number of drop sites. |

`OldDnDGetDropSitesOfWidget()` obtains the currently registered list of drop sites for a particular widget instance. The function returns a pointer to an `OldnDDropSiteID` array that is an enumeration of the drop sites currently registered for the widget. Clients should use `XtFree()` on this return value to deallocate the array when it is no longer needed. If there are no drop sites registered or the function fails, `OldDnDGetDropSitesOfWidget()` returns NULL.

### *OldDnDGetDropSitesOfWindow*

```
#include <Xol/OldnDVCX.h>
OldnDDropSiteID *OldDnDGetDropSitesOfWindow(
 Display *dpy,
 Window window,
 Cardinal *num_sites_return);
```

---

## Drag and Drop Functions

|                         |                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------|
| <i>dpy</i>              | The display pointer.                                                                  |
| <i>window</i>           | The window associated with the owner of the drop site.                                |
| <i>num_sites_return</i> | A pointer to a variable into which the function will return the number of drop sites. |

`OldNnDGetDropSitesOfWindow()` obtains the currently registered list of drop sites for a particular window. The function returns a pointer to an `OldNnDDropSiteID` array that is an enumeration of the drop sites currently registered for the window. Clients should use `XtFree()` on this return value to deallocate the array when it is no longer needed. If there are no drop sites registered or the function fails, `OldNnDGetDropSitesOfWindow()` returns `NULL`.

### *OldNnDGetWidgetOfDropSite*

```
#include <Xol/OldNnDVCX.h>
Widget OldNnDGetWidgetOfDropSite(
 OldNnDDropSiteID dropsiteid);
```

`OldNnDGetWidgetOfDropSite()` returns the ID of the widget associated with the drop site, specified by the *dropsiteid* argument. If the drop site was registered with `OldNnDRegisterWindowDropSite()`, `OldNnDGetWidgetOfDropSite()` returns the ID of the widget that is the most immediate ancestor of the associated window.

### *OldNnDGetWindowOfDropSite*

```
#include <Xol/OldNnDVCX.h>
Window OldNnDGetWindowOfDropSite(
 OldNnDDropSiteID dropsiteid);
```

`OldNnDGetWindowOfDropSite()` returns the window ID that the drop site specified by the *dropsiteid* argument is associated with. If the drop site was registered with a gadget, then `OldNnDGetWindowOfDropSite()` returns the window ID of the gadget's windowed parent.

### OldnDQueryDropSiteInfo

```
#include <Xol/OldnDVCX.h>
Boolean OldnDQueryDropSiteInfo(
 OldnDDropSiteID dropsiteid,
 Widget *widget,
 Window *window,
 OldnDSitePreviewHints *preview_hints,
 OldnDSiteRectPtr *site_rects,
 unsigned int *num_rects,
 Boolean *on_interest);
```

- widget*            The address of a variable of type `Widget` that returns the ID of the widget that owns the drop site. Set this parameter to `NULL` if no query on the widget ID is required. If the drop site was registered with `OldnDRegisterWindowDropSite()`, this is the widget ID of the associated window's most immediate ancestor.
- window*            The address of a variable of type `Window` that returns the ID of the window that owns the drop site. Set this parameter to `NULL` if no query on the window ID is required. For gadgets, this is the window ID of its windowed ancestor.
- preview\_hints*    The address of a variable of type `OldnDSitePreviewHints` that returns the current hints for the drop site. Set this parameter to `NULL` if no query on the preview hints is required.
- site\_rects*        The address of a variable of type `OldnDSiteRectPtr` that returns a pointer to an array that contains the current geometry of the drop site. This parameter may be set to `NULL` if no query on the site geometry is required. Clients must use `XtFree()` to deallocate the memory used by the array when they no longer require it.
- num\_rects*         The address of an `unsigned int` variable that returns the number of `OldnDSiteRect` structures specified for the drop site. Set this parameter to `NULL` if no query on the number of rectangles.
- on\_interest*      The address of a `Boolean` variable that returns a value indicating whether the drop site is currently active (`TRUE`) or inactive (`FALSE`). Set this parameter to `NULL` if this value is not required.



---

## Drag and Drop Functions

`OldnDQueryDropSiteInfo()` retrieves information about the drop site specified by the *dropsiteid* argument. The function returns TRUE if the query was successful; otherwise, it returns FALSE.

### *OldnDRegisterWidgetDropSite*

```
#include <Xol/OldnDVCX.h>
OldnDDropSiteID OldnDRegisterWidgetDropSite(
 Widget widget,
 OldnDSitePreviewHints preview_hints,
 OldnDSiteRectPtr site_rects,
 unsigned int num_sites,
 OldnDTMNotifyProc tmnotify,
 OldnDPMNotifyProc pmnotify,
 Boolean on_interest,
 XtPointer client_data);
```

*client\_data* Application-defined data that is passed to the *tmnotify* and *pmnotify* functions when they are called.

*on\_interest* Specifies whether the drop site is active (i.e., “interested” in responding to drops). TRUE means the drop site is active, FALSE means it is inactive. An inactive drop site is ignored during a Drag and Drop operation; its Preview Message Notify Procedure is not called when the cursor passes over it, nor is the source’s Preview Animate Callback. This drop site attribute may be changed at any time during the existence of the drop site using the function `OldnDSetDropSiteInterest()`.

The other arguments to this function are described in “Common Arguments” on page 114.

`OldnDRegisterWidgetDropSite()` creates a drop site associated with a particular widget. The widget must be realized; that is, it must have a window associated with it before you can create a drop site for it. Gadgets can support drop sites and use their windowed ancestor’s window in association with the registered drop site. Drop sites are automatically destroyed when their owning widgets die.

### *OldnDRegisterWindowDropSite*

```
#include <Xol/OldnDVCX.h>
OldnDDropSiteID OldnDRegisterWindowDropSite(
 Display *dpy,
 Window window,
 OldnDSitePreviewHints preview_hints,
 OldnDSiteRectPtr site_rects,
 unsigned int num_sites,
 OldnDTMNotifyProc tmnotify,
 OldnDPMNotifyProc pmnotify,
 Boolean on_interest,
 XtPointer client_data);
```

*dpy*            The display pointer.

*client\_data*   Application-defined data that is passed to the *tmnotify* and *pmnotify* functions when they are called.

The other arguments to this function are described in “Common Arguments” on page 114.

*OldnDRegisterWindowDropSite()* registers a window-based drop site. It creates a drop site associated with a particular X Window and is useful for toolkit applications that mix “raw” X windows with widgets. Drop sites are automatically destroyed when their owning windows die. The window must be an inferior of a widget’s window.

### *OldnDSetDropSiteInterest*

```
#include <Xol/OldnDVCX.h>
void OldnDSetDropSiteInterest(
 OldnDDropSiteID dropsiteid,
 Boolean on_interest);
```

*on\_interest*   TRUE means the drop site is made active, FALSE means the drop site is made inactive.

*OldnDSetDropSiteInterest()* activates or inactivates a drop site by exporting its existence. Active drop sites respond to drops. Inactive drop sites do not respond to drops.

### *OldnDSetInterestInWidgetHier*

```
#include <Xol/OldnDVCX.h>
void OldnDSetInterestInWidgetHier(
 Widget widget,
 Boolean on_interest);
```

*on\_interest* TRUE means the drop sites are made active, FALSE means the drop sites are made inactive.

`OldnDSetInterestInWidgetHier()` activates or inactivates all drop sites belonging to this widget and its children.

### *OldnDUpdateDropSiteGeometry*

```
#include <Xol/OldnDVCX.h>
Boolean OldnDUpdateDropSiteGeometry(
 OldnDDropSiteID dropsiteid,
 OldnDSiteRectPtr site_rects,
 unsigned int num_sites);
```

*dropsiteid* The ID of the drop site to be updated.

*site\_rects* The new list of site rectangles for the drop site.

*num\_sites* The number of rectangles in the new rectangle list.

`OldnDUpdateDropSiteGeometry()` alters the geometry of a drop site. Changes in the geometry of a drop site are caused by changes in the geometry of the widget or window that owns the drop site. To reduce client-server traffic, the toolkit does not automatically track changes in windows that own drop sites. The creator of a drop site is responsible for maintaining the geometry of the site to reflect any changes in the widget or window that owns the site.

### *OldnDWidgetConfiguredInHier*

```
#include <Xol/OldnDVCX.h>
void OldnDWidgetConfiguredInHier(
 Widget widget);
```

This function is primarily for use by developers of Composite widgets.

## Drag and Drop Functions

Since drop sites are separate from the server window hierarchy, drop site owners must attempt to maintain their drop sites clipped to their visible region(s), as defined by the server window hierarchy associated with the widget hierarchy that contains the drop sites.

In order to achieve this clipping, `Composite` widgets and their subclasses must inform the Drag and Drop system that they have configured some widgets in their subtree, as a result of a call to that `Composite` widget's `ChangeManaged()` or `GeometryManager()` methods (hence potentially changing the visible region(s) of drop sites in that subtree). Calling this function will cause the Drag and Drop system to recalculate the clipping region(s) of any drop sites under the configuring widget in the widget hierarchy.

In order to eliminate multiple recalculations of drop site clipping region(s) due to configures propagating down a widget hierarchy, a mechanism exists to suppress such multiple calculations; developers should take advantage of this in order to optimize performance.

The following is an example of the usage of this function in a simple geometry manager:

```
static XtGeometryResult GeometryManager(
 Widget requester;
 XtWidgetGeometry *request,
 XtWidgetGeometry *reply);
{
 CompositeWidget comp = requester->core.parent;
 Widget vendor = comp;
 Arg args[2];
 Boolean configured_others = False;
 XtSetArg(args[0], XtNconfiguringWidget, (XtPointer)requester);
 XtSetArg(args[1], XtNdisableDSClipping, True);
 while (!XtIsVendorShell(w))
 vendor = vendor->core.parent;
 XtSetValues(vendor, args, XtNumber(args));
 /* Disable clipping in my subtree while I configure. */
 /* Consider the geometry request received and then maybe
 * configure one or more of the managed set and/or perhaps
 * request that my parent reconfigure me as a result of the
 * request being made by the requester widget.
 */
}
```

---

*Drag and Drop Functions*

```
* Set configured_others True if the Composite made a successful
* geometry request to its parent, or if it moved siblings of
* the requester */
if ((request->request_mode & CWX) == CWX)
 requester->core.x = request->x;
if ((request->request_mode & CWY) == CWY)
 requester->core.y = request->y;
if ((request->request_mode & CWWidth) == CWWidth)
 requester->core.width = request->width;
if ((request->request_mode & CWHeight) == CWHeight)
 requester->core.height = request->height;
XtSetArg(args[1], XtNdisableDSClipping, False);
XtSetValues(vendor, args, XtNumber(args));
/* enable clipping again */
/* Inform the drag and drop system to clip any drop sites in
 * the widget hierarchy under the configuring widget. */
if (configured_others)
 OldnDWidgetConfiguredInHier((Widget)comp);
else
 OldnDWidgetConfiguredInHier(requester);
return XtGeometryYes;
}
```

*See Also*

“Cursor and Pixmap Functions” on page 99,  
“DropTarget Widget” on page 266.

---

## Dynamic Resource Functions

### Dynamic Resource Functions

OLIT supports Dynamic Resources. This means that for selected resource classes (those resources that include a D in the “Access” column), OLIT will detect dynamic changes in the server’s resource database (by looking for updates on the RESOURCE\_MANAGER property of the RootWindow) and automatically update the resource inside the widget.

The following routines allow applications to use this Dynamic Resource functionality.

---

**Note** – The `OlGetApplicationResources()` and `LookupOlColors()` routines previously included with the dynamic settings functions are no longer supported.

---

#### *OlCallDynamicCallbacks*

```
#include <Xol/Dynamic.h>
void OlCallDynamicCallbacks(void)
```

`OlCallDynamicCallbacks()` triggers the calling of the functions registered on the dynamic callback list. This procedure is called automatically whenever the RESOURCE\_MANAGER property of the RootWindow is updated. It may also be called to force a synchronization of the dynamic settings.

#### *OlRegisterDynamicCallback*

```
#include <Xol/Dynamic.h>
void OlRegisterDynamicCallback(
 OlDynamicCallbackProc CB,
 XtPointer data);
```

`OlRegisterDynamicCallback()` adds a function to the list of registered callbacks to be called whenever the procedure `OlCallDynamicCallbacks()` is invoked. `OlCallDynamicCallbacks()` is invoked whenever the RESOURCE\_MANAGER property of the Root Window is updated. `OlCallDynamicCallbacks()` may also be called directly by either the

---

## *Dynamic Resource Functions*

application or other routines in the widget libraries. The callbacks registered are guaranteed to be called in first-in-first-out (FIFO) order of registration and will be called as:

```
(*CB)(data);
```

### ***OlUnregisterDynamicCallback***

```
#include <Xol/Dynamic.h>
int OlUnregisterDynamicCallback(
 OlDynamicCallbackProc CB
 XtPointer data);
```

**OlUnregisterDynamicCallback()** removes a function from the list of registered callbacks to be called whenever **OlCallDynamicCallbacks()** is invoked. It returns zero if the dynamic callback cannot be removed; otherwise, it returns 1.

## Error Functions

The following functions provide error and warning message services.

Most programs should not use `OlError()` and `OlWarning()` since they do not allow for customization or internationalization.

The `OpenLook.h` header does not include `stdarg.h` or `varargs.h`. An application using `OlSetVaDisplayErrorMsgHandler()` or `OlSetVaDisplayWarningMsgHandler()` should include one of these two headers before including `OpenLook.h` to ensure the correct function prototype will be used for the application's error/warning handler.

### *OlError*

```
#include <Xol/OpenLook.h>
void OlError(
 String msg);
```

`OlError()` writes a string to *stderr* and then exits.

### *OlWarning*

```
#include <Xol/OpenLook.h>
void OlWarning(
 String msg);
```

`OlWarning()` writes a string to *stderr* and then returns.

### *OlVaDisplayErrorMsg*

```
#include <Xol/OpenLook.h>
void OlVaDisplayErrorMsg(
 Display *dpy,
 String name,
 String type,
 String class,
 String default_msg,
 ...);
```

`OlVaDisplayErrorMsg()` writes an error message to *stderr* and exits. The error message is looked up in the error database by calling `XtAppGetErrorDatabaseText()` using the *name*, *type*, and *class* arguments.



## Error Functions

If no message is found in the error database, the *default\_msg* string is used. The application context is determined by calling `XtDisplayToApplicationContext()` with the supplied `Display` pointer. If the display pointer is `NULL`, the display created at application startup is used to determine the application context.

### *OlVaDisplayWarningMsg*

```
#include <Xol/OpenLook.h>
void OlVaDisplayWarningMsg(
 Display *dpy,
 String name,
 String type,
 String class,
 String default_msg,
 ...);
```

`OlVaDisplayWarningMsg()` has the same semantics as `OlVaDisplayErrorMsg()`, except that it returns instead of exiting.

### *OlSetErrorHandler*

```
#include <Xol/OpenLook.h>
OlErrorHandler OlSetErrorHandler(
 OlErrorHandler handler);
```

`OlSetErrorHandler()`, `OlSetWarningHandler()`, `OlSetVaDisplayErrorMsgHandler()`, and `OlSetVaDisplayWarningMsgHandler()` allow an application to override the various warning and error handlers. These routines return a pointer to the previous handler. If `NULL` is supplied to any of these routines, the default handler will be used. Application-supplied error handlers should do the same since continuation of an application will result in undefined behavior.

### *OlSetWarningHandler*

```
#include <Xol/OpenLook.h>
OlWarningHandler OlSetWarningHandler(
 OlWarningHandler handler);
```

See `OlSetErrorHandler()` above.

### ***OlSetVaDisplayErrorMsgHandler***

```
#include <Xol/OpenLook.h>
OlVaDisplayErrorMsgHandler OlSetVaDisplayErrorMsgHandler(
 OlVaDisplayErrorMsgHandler handler,
 ...);
```

See `OlSetErrorHandler()` above.

### ***OlSetVaDisplayWarningMsgHandler***

```
#include <Xol/OpenLook.h>
OlVaDisplayWarningMsgHandler OlSetVaDisplayWarningMsgHandler(
 OlVaDisplayWarningMsgHandler handler,
 ...);
```

See `OlSetErrorHandler()` above.

### ***OlErrorHandler***

```
#include <Xol/OpenLook.h>
typedef void (*OlErrorHandler)(
 String msg);
```

### ***OlWarningHandler***

```
#include <Xol/OpenLook.h>
typedef void (*OlWarningHandler)(
 String msg);
```

### ***OlVaDisplayErrorMsgHandler***

```
#include <Xol/OpenLook.h>
typedef void (*OlVaDisplayErrorMsgHandler)(
 Display *dpy,
 String name,
 String type,
 String class,
 String default_msg,
 ...);
```

***OIVaDisplayWarningMsgHandler***

```
#include <Xol/OpenLook.h>
typedef void (*OIVaDisplayWarningMsgHandler)(
 Display *dpy,
 String name,
 String type,
 String class,
 String default_msg,
 ...);
```

*Help Function*  
*Help Function*

The following function is used to register help.

***OlRegisterHelp***

```
#include <Xol/OpenLook.h>
void OlRegisterHelp(
 OlDefine id_type,
 XtPointer id,
 String tag,
 OlDefine source_type,
 XtPointer source);
```

`OlRegisterHelp()` associates help information with either a widget instance or a widget class. The widget ID or widget class pointer is given in *id*, and *id\_type* identifies whether it is a widget or a widget class using one of the values `OL_WIDGET_HELP` or `OL_CLASS_HELP`, respectively. Use `OL_WIDGET_HELP` to register help on gadgets. The other arguments are explained in “Format of Help” on page 147.

The *tag* value is shown in the title of the help window, as follows:

```
app-name:tag Help
```

where *app-name* is the name of the application. The *tag* can be null, in which case only *app-name: Help* is printed.

***Help for Flat Widgets***

To set the same help message for all items in a flat widget container, use the `OlRegisterHelp()` routine with *id\_type* set to `OL_WIDGET_HELP`. To register help for individual items in a flat widget container, use `OlRegisterHelp()` with *id\_type* set to `OL_FLAT_HELP`. Use the following structure to specify the object that gets the help message and pass `OlRegisterHelp()` a pointer to it in the *id* parameter:

```
typedef struct {
 Widget widget;
 Cardinal item_index;
} OlFlatHelpId;
```

## Format of Help

The help message is identified in *source*; *source\_type* identifies the form of the help message as one of the following:

### ***OL\_STRING\_SOURCE***

The *source* is of type `String` and contains text with embedded newlines. `OlRegisterHelp()` does not copy this source; the application is expected to maintain the original as long as it is registered.

### ***OL\_DISK\_SOURCE***

The *source* is also of type `String`, but contains the name of a file that contains the help text. `OlRegisterHelp()` does not copy this filename; the application is expected to maintain the original as long as it is registered. The file content is expected to be text with embedded newlines.

### ***OL\_INDIRECT\_SOURCE***

The *source* is of type `void(*)()` and is a pointer to an application-defined routine to be called by OLIT. This routine is called after HELP has been clicked. The application is expected to define the type of the help source in the routine; after it has returned, the help information will be displayed.

The routine is called as follows:

```
(*source)(id_type, id, src_x, src_y, &source_type, &source);
```

|                                      |                                                                                                                                                                                                                  |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>id_type</i> and <i>id</i>         | The values for the widget class or widget instance that was under the pointer when HELP was pressed. These are the same values previously registered with <code>OlRegisterHelp()</code> .                        |
| <i>src_x</i> and <i>src_y</i>        | The coordinates of the pointer when HELP was pressed. These are relative to the upper-left corner of the window.                                                                                                 |
| <i>source_type</i> and <i>source</i> | Pointers to values the application's routine should set for the help source it wants to display. The only <i>source_type</i> values accepted are <code>OL_STRING_SOURCE</code> and <code>OL_DISK_SOURCE</code> . |

*Help Function*

***OL\_TRANSPARENT\_SOURCE***

The *source* is of type `void(*)()` and is a pointer to an application-defined routine. The routine is called after HELP has been invoked. The application is expected to handle the HELP event completely. This might be used by an application that does not want the standard help window (for example, `xterm(1)` simply generates an escape sequence).

The routine is called as follows:

```
(*source)(id_type, id, src_x, src_y);
```

*id\_type* and *id*     The values for the widget class or widget instance that was under the pointer when HELP was pressed. These are the same values registered with `OlRegisterHelp()`.

*src\_x* and *src\_y*    The coordinates of the pointer when HELP was pressed. These are relative to the upper-left corner of the window.

The help window is automatically popped up for the `OL_STRING_SOURCE`, `OL_DISK_SOURCE`, and `OL_INDIRECT_SOURCE` help sources. (It is popped up after the application routine returns for the `OL_INDIRECT_SOURCE` help source.) The application is responsible for popping up a help window (if needed) for the `OL_TRANSPARENT_SOURCE` help source.

***Handling the Help Key Event***

When the user clicks HELP, if the event occurs within a widget or window registered with the `OlRegisterHelp()` routine, the corresponding help message is automatically displayed (for source types `OL_STRING_SOURCE` and `OL_DISK_SOURCE`) or the application routine is called (for source types `OL_INDIRECT_SOURCE` and `OL_TRANSPARENT_SOURCE`). If the event occurs elsewhere, a default help message is displayed.

If the help key is pressed on a widget, the help routine looks for help registered on that widget of type `OL_WIDGET_HELP`. If no help is found, the help routine searches up the widget tree (i.e., goes to the widget's parents up to a widget which is a subclass of shell) looking for the first widget that has help of type `OL_WIDGET_HELP` registered. If it finds help registered on one of the ancestors of the original widget, the help message for that widget will be used. If help is not found, the help routine looks for help of type `OL_CLASS_HELP` on the original widget. If no help is found, the default message is used.

---

## *Help Function*

The use of `OlRegisterHelp()` is considered across all applications. In other words, even though a regular application does not register help for the root window (the “workspace”), it does not mean that pressing HELP on the root window causes a default message. Another application (typically the workspace manager) may have registered the help.

### *Separate Help per Application*

An application will have, at most, one help message displayed. However, several applications can display their separate help messages simultaneously, in different help windows.

### *Displaying the Help Message*

A help source of type `OL_STRING_SOURCE` and `OL_DISK_SOURCE` is displayed in a help window that is 50 ens wide and 10 lines tall. (An en is  $S/2$  points, where  $S$  is the current point size.)

Lines longer than the help window width are wrapped at the space(s) between words, or at the nearest character boundary if there is no space at which to wrap. Lines are also wrapped at embedded newlines, regardless of their lengths.

Only spaces and newlines are recognized for format control; all other non-printable characters are silently ignored.

Up to ten lines of the message are visible at once. Messages longer than ten lines have a scrollbar control that allows scrolling non-visible lines into view.

### *Static Variables*

The *tag* and *source* values should be statically defined (or allocated and not freed). Using automatic variables here will almost always fail.

## Input Focus Functions

Each of these utility routines works with widgets or gadgets to manipulate input focus.

### *OlCallAcceptFocus*

```
#include <Xol/OpenLook.h>
Boolean OlCallAcceptFocus(
 Widget w,
 Time time);
```

`OlCallAcceptFocus()` sets the focus to a specified widget. If widget `w` currently is capable of accepting input focus, `OlCallAcceptFocus()` assigns focus to `w` and it returns TRUE; otherwise, it returns FALSE. See the `XtCallAcceptFocus()` function in the *Xt Intrinsic Reference Manual* for further details about the `time` argument.

---

**Note** – `OlCallAcceptFocus()` will be declared obsolete in a future version of OLIT. You should use the `XtCallAcceptFocus()` for all new applications.

---

### *OlCanAcceptFocus*

```
#include <Xol/OpenLook.h>
Boolean OlCanAcceptFocus(
 Widget w,
 Time time);
```

`OlCanAcceptFocus()` tests whether a widget can accept focus. If it can accept focus, it returns TRUE; otherwise, it returns FALSE. Acceptance of focus is determined by all of the following being true:

- The widget is not being destroyed.
- The widget is managed.
- The widget is mapped when managed (if it is not a gadget).
- The widget is realized, or for a gadget, the gadget's parent are realized.
- The widget and its ancestors are sensitive.



## Input Focus Functions

- A query for the widget's Window attributes is successful and the widget's window is viewable (i.e., the window and all its ancestor windows are mapped).
- The `XtNmouseless` resource is TRUE or the widget is a shell or text input widget.

### *OlSetInputFocus*

```
#include <Xol/OpenLook.h>
void OlSetInputFocus(
 Widget w,
 int revert_to,
 Time time);
```

`OlSetInputFocus()` sets focus to a widget. Applications should use this routine instead of `XSetInputFocus()`; see the description of `XSetInputFocus()` in the *XLib Reference Manual* for further details about the `revert_to` and `time` arguments. If `XtNmouseless` is FALSE, `OlSetInputFocus()` is ignored unless the widget is a text input or shell widget.

### *OlGetCurrentFocusWidget*

```
#include <Xol/OpenLook.h>
Widget OlGetCurrentFocusWidget(
 Widget w);
```

`OlGetCurrentFocusWidget()` returns the widget that currently has focus in the window group of the specified widget. If no widget in the window group has focus, `OlGetCurrentFocusWidget()` returns NULL.

### *OlHasFocus*

```
#include <Xol/OpenLook.h>
Boolean OlHasFocus(
 Widget w);
```

`OlHasFocus()` returns TRUE if the specified widget has focus. `OlHasFocus()` simply calls `OlGetCurrentFocusWidget()` and compares its return value to the supplied widget.

***OlMoveFocus***

```
#include <Xol/OpenLook.h>
Widget OlMoveFocus(
 Widget w,
 OlVirtualName direction,
 Time time);
```

`OlMoveFocus()` moves the input focus relative to the widget `w`, as indicated by *direction*, and returns the new focus widget. It calls `OlCallAcceptFocus()` to move the input focus. If `OlCallAcceptFocus()` is unable to move input focus, `OlMoveFocus()` returns NULL. It will also return NULL if `XtNmouseless` is set to FALSE and the widget is not a text input widget. When moving input focus between widgets contained within an `Exclusives` or `Nonexclusives` widget, valid values for *direction* are shown in the following list. For the `OL_MULTI` directions below, the value of *m* is the value of the toolkit resource `XtNmultiObjectCount`. See the description of `XSetInputFocus()` in the *XLib Reference Manual* for further details about the *time* argument.

- `OL_IMMEDIATE`     Set focus to the next widget that will accept it, starting with `w`.
- `OL_MOVERIGHT`    Set focus to the widget in the next column (and same row) that will accept it, starting with the first column after `w`'s column. If `w` is located in the extreme right column, focus is set to the widget in the extreme left column of the same row.
- `OL_MOVELEFT`     Set focus to the widget in the previous column (and same row) that will accept it, starting with the first column before `w`'s column. If `w` is located on the extreme left column, focus is set to the widget in the extreme right column of the same row.
- `OL_MOVEUP`        Set focus to the widget in the previous row (and same column) that will accept it, starting with the first row before `w`'s row. If `w` is located in the top row, focus is set to the widget in the bottom row of the same column.
- `OL_MOVEDOWN`     Set focus to the widget in the next row (and same column) that will accept it, starting with the first row after `w`'s row. If `w` is located in the bottom row, focus is set to the widget in the top row of the same column.
- `OL_MULTIRIGHT`    Set focus to the widget in the next column (and same row) that will accept it, starting with the first column *m* columns after `w`'s column. If *m* is greater than the number of objects

*Input Focus Functions*

|              |                                                                                                                                                                                                                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | between <i>w</i> and the extreme right column, focus is set to the widget in the extreme left column of the same row.                                                                                                                                                                                                                 |
| OL_MULTILEFT | Set focus to the widget in the previous column (and same row) that will accept it, starting with the first column <i>m</i> columns before <i>w</i> 's column. If <i>m</i> is greater than the number of objects between <i>w</i> and the extreme left column, focus is set to the widget in the extreme right column of the same row. |
| OL_MULTIUP   | Set focus to the widget in the previous row (and same column) that will accept it, starting with the first row <i>m</i> rows before <i>w</i> 's row. If <i>m</i> is greater than the number of objects between <i>w</i> and the extreme top row, focus is set to the widget in the extreme bottom row of the same column.             |
| OL_MULTIDOWN | Set focus to the widget in the next row (and same column) that will accept it, starting with the first row <i>m</i> rows after <i>w</i> 's row. If <i>m</i> is greater than the number of objects between <i>w</i> and the extreme bottom row, focus is set to the widget in the extreme top row of the same column.                  |

When moving between widgets in a base window or popup window, focus is moved according to the order defined by the *traversal list*. The default traversal order is determined by the order in which widgets are created. Valid values for *direction* are:

|                                         |                                                                                                                                                       |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| OL_IMMEDIATE                            | Set focus to the next object that will accept it, starting with <i>w</i> .                                                                            |
| OL_NEXTFIELD, OL_MOVERIGHT, OL_MOVEDOWN | Set focus to the next object that will accept it, starting with the first object after <i>w</i> .                                                     |
| OL_PREVFIELD, OL_MOVELEFT, OL_MOVEUP    | Set focus to the next object that will accept it, starting with the first object before <i>w</i> . (The list is searched in reverse order.)           |
| OL_MULTIRIGHT, OL_MULTIDOWN             | Set focus to the next object that will accept it, starting with the first <i>m</i> objects after <i>w</i> .                                           |
| OL_MULTILEFT, OL_MULTIUP                | Set focus to the next object that will accept it, starting with the first <i>m</i> objects before <i>w</i> . (The list is searched in reverse order.) |

*Multiple Visual Functions*

***Multiple Visual Functions***

The following functions are used to work with multiple visuals.

A visual is specified by a *depth* (for example, 8 bits) and a *visual class* (for example, `PseudoColor`).

A shell widget or a `DrawArea` widget can have a nondefault visual. Other widgets use the visuals of their nearest shell or `DrawArea` ancestor. An application in which eligible widgets have nondefault visuals is termed a multi-visual application.

You must specify the visual class when you specify the depth, or the depth will be ignored. Specifically, you should use the `XtVaTypedArg` interface with `XtVaCreateManagedWidget()`, rather than `XMatchVisualInfo()`. (The argument list interface implicitly invokes the resource converter, while `XMatchVisualInfo()` does not. Trying to set the depth without also setting the visual class and running the resource converter can create problems.)

For example, in creating a `DrawArea` widget using this interface, you might use something like:

```
drawarea = XtVaCreateManagedWidget("drawarea",
 drawAreaWidgetClass, toplevel,
 XtVaTypedArg, XtNvisual, XtRString,
 VisualClassName, sizeof(VisualClassName),
 XtNlayout, OL_IGNORE,
 XtNheight, Height,
 XtNwidth, Width,
 NULL);
```

Each multiple visual function returns a characteristic of either a widget or gadget.

Multiple visuals are meant to run on machines with hardware colormaps; otherwise, serious flashing results when the mouse pointer moves between applications or widgets with different visuals.

***OIBlackPixel***

```
#include <Xol/OpenLook.h>
Pixel OIBlackPixel(
 Widget w);
```

---

## Multiple Visual Functions

`OlBlackPixel()` returns the black pixel for the colormap associated with the given widget. Use this function instead of the macro `BlackPixel()`, in OLIT applications that use multiple colormaps and/or multiple visuals.

### *OlColormapOfObject*

```
#include <Xol/OpenLook.h>
Colormap OlColormapOfObject(
 Widget object);
```

`OlColormapOfObject()` obtains the colormap associated with the *object*.

### *OlDepthOfObject*

```
#include <Xol/OpenLook.h>
int OlDepthOfObject(
 Widget object);
```

`OlDepthOfObject()` obtains the depth associated with the *object*.

### *OlInternAtom*

```
#include <Xol/RootShell.h>
Atom OlInternAtom(
 Display *dpy,
 String atom_name);
```

`OlInternAtom()` uses the Intrinsic `XtRString-to-XtRAtom` resource converter and the converter cache to store Atoms in the resource cache on a per display basis.

You should use this function to cache Atoms across displays, especially for applications using multiple displays.

For efficient use of the resource converter cache, the string *atom\_name* should be the same physical string for each invocation. For example:

```
OlInternAtom(dpy, "foo");
OlInternAtom(dpy, "foo");
```

results in *two* entries in the resource converter cache, while the following results in only one cache entry:

## Packed Widget Function

```
char *foo = "foo";
OlInternAtom(dpy, foo);
OlInternAtom(dpy, foo);
```

### *OlVisualOfObject*

```
#include <Xol/OpenLook.h>
Visual *OlVisualOfObject(
 Widget object);
```

`OlVisualOfObject()` obtains the visual associated with the *object*.

### *OlWhitePixel*

```
#include <Xol/OpenLook.h>
Pixel OlWhitePixel(
 Widget w);
```

`OlWhitePixel()` returns the white pixel for the colormap associated with the given widget. Use this function instead of the macro `WhitePixel()`, in OLIT applications that use multiple colormaps and/or multiple visuals.

## Packed Widget Function

The following function creates a widget (sub)tree in one call.

### *OlCreatePackedWidgetList*

```
#include <Xol/OpenLook.h>
Widget OlCreatePackedWidgetList(
 OlPackedWidgetList *pw_list,
 Cardinal num_pw);
```

`OlCreatePackedWidgetList()` and its associated `OlPackedWidget` structure allow an application to create a widget tree or subtree in one call.

***pw\_list***     A pointer to an `OlPackedWidget` array. It creates widgets starting from the first element in the array.

***num\_pw***     The number of elements in the array *pw\_list*

`OlCreatePackedWidgetList()` returns the widget ID of the first element in the array *pw\_list*.

## Packed Widget Function

The `OlPackedWidget` structure contains all the information needed to create a new widget. It is defined as:

```
typedef struct {
 Widget widget;
 String name;
 WidgetClass *class_ptr;
 Widget *parent_ptr;
 String descendant;
 ArgList resources;
 Cardinal num_resources;
 Boolean managed;
} OlPackedWidget, *OlPackedWidgetList;
```

|                             |                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>widget</i></b>        | Contains the ID of the newly created widget.                                                                                                                                                                                                                                                                                                                                                        |
| <b><i>name</i></b>          | The name of the widget that will be created.                                                                                                                                                                                                                                                                                                                                                        |
| <b><i>class_ptr</i></b>     | A pointer to the <code>WidgetClass</code> pointer for the new widget. This gives the class of widget to create. It is a pointer to the pointer because typically the pointer itself is an external value that is not suitable for using in an array initialization.                                                                                                                                 |
| <b><i>parent_ptr</i></b>    | A pointer to the widget ID of the intended parent of the new widget <i>or</i> the ID of an indirect widget that “knows who the parent is” (see below). This value may point to a <i>widget</i> member in another <code>PackedWidget</code> item; if the parent is an indirect widget, it must appear earlier in the list.                                                                           |
| <b><i>descendant</i></b>    | The name of a resource available in the widget identified by <i>parent_ptr</i> . The value of this resource is the ID of the real parent for the new widget. If the <i>descendant</i> value is not zero, <i>parent</i> is expected to identify an indirect parent that is interrogated for the ID of the real parent. If this value is zero, <i>parent</i> is expected to identify the real parent. |
| <b><i>resources</i></b>     | The resource array to use when creating the new widget.                                                                                                                                                                                                                                                                                                                                             |
| <b><i>num_resources</i></b> | The number of resources in the array.                                                                                                                                                                                                                                                                                                                                                               |
| <b><i>managed</i></b>       | TRUE if the new widget should be managed when created, FALSE otherwise.                                                                                                                                                                                                                                                                                                                             |

*Pixel Conversion Functions*

***Pixel Conversion Functions***

The following routines convert pixel dimensions to other measurements.

```
#include <Xol/OpenLook.h>
Screen *OlDefaultScreen;
Display *OlDefaultDisplay;
Axis axis;
Screen screen;

OlMMToPixel(axis, millimeters);
Ol_MMToPixel(axis, millimeters);

OlPointToPixel(axis, points);
Ol_PointToPixel(axis, points);

OlScreenMMToPixel(axis, millimeters, screen);
Ol_ScreenMMToPixel(axis, millimeters, screen);

OlScreenPointToPixel(axis, points, screen);
Ol_ScreenPointToPixel(axis, points, screen);

OlPixelToMM(axis, pixels);
Ol_PixelToMM(axis, pixels);

OlPixelToPoint(axis, pixels);
Ol_PixelToPoint(axis, pixels);

OlScreenPixelToPoint(axis, pixels, screen);
Ol_ScreenPixelToPoint(axis, pixels, screen);

OlScreenPixelToMM(axis, pixels, screen);
Ol_ScreenPixelToMM(axis, pixels, screen);
```

All the X-based OPEN LOOK widgets refer to pixels in coordinates and dimensions for compatibility with other X Window System widgets.

This puts the burden on the application programmer to convert between externally useful measures, such as points or millimeters, and pixels as applied to the screen at hand. These routines examine the data structures that describe the physical dimensions and the pixel resolution of a screen and convert among millimeters, points, and pixels for that screen.



***Screen Selection***

The shorter forms of these routines (the ones without the word `Screen` in their names) work for the default screen. This is the screen that is active when the X-Toolkit Intrinsic is started. The longer forms of these routines take a `Screen *` type argument that refers to a particular screen. The macros `OlDefaultScreen` and `OlDefaultDisplay` identify the current screen and display being used by the Intrinsic.

***Use After Toolkit Initialization***

These routines make use of data structures that are initialized when the Toolkit is initialized (see Initialization and Activation Functions on page 92). Therefore, using them before toolkit initialization (for example, as an initial value to a statically defined variable) will result in a run-time error.

***Axis Argument***

The first argument of all the routines is the direction in which the measurement is made. This is necessary because not all screens have equivalent resolution in the horizontal and vertical axes. The *axis* argument can take one of the two values: `OL_HORIZONTAL` or `OL_VERTICAL`. These routines are not directly usable in computing a diagonal measure. (Find the diagonal with the Pythagorean Theorem:  $a^2 + b^2 = c^2$ ).

***Implemented as Macros***

All these routines are implemented as macros, so they can take any reasonable type value for the millimeters, points, and pixels. The macros cast the values into the proper type needed for the conversion. However, only a single type value can be “returned.”

The routines without an underscore in their names produce values of type `int` (the values are rounded to the nearest integer). The routines with an underscore in their names produce values of type `double` (these values have not been rounded, leaving it up to the application to round up, round down, or truncate as needed). Given the small size of the units involved, the integer-returning routines should be sufficient for many applications.

Because these routines are implemented as macros, there are no function addresses available.

*Protocol Function*  
*Protocol Function*

***OlWMPProtocolAction***

`OlWMPProtocolAction()` simulates a response to any window manager's protocol messages.

```
#include <Xol/OpenLook.h>
void OlWMPProtocolAction(
 Widget w,
 OlWMPProtocolVerify *st,
 OlDefine action)
```

The `w` parameter must be a widget that is a subclass of `VendorShell`. Otherwise, no action will be taken.

The `OlWMPProtocolVerify` structure is defined as follows:

```
typedef struct {
 unsigned long msgtype;
 XEvent *xevent;
} OlWMPProtocolVerify;
```

Its `msgtype` field is an integer constant indicating the type of protocol message that invoked the callback; it will be one of the following values:

- OL\_WM\_TAKE\_FOCUS
- OL\_WM\_SAVE\_YOURSELF
- OL\_WM\_DELETE\_WINDOW

The `action` parameter can be:

- OL\_QUIT                   Quit the application immediately.
- OL\_DEFAULTACTION       Perform the action that is appropriate for each subclass of `VendorShell`.
- OL\_DESTROY               Destroy the shell widget.
- OL\_DISMISS               Dismiss or unmap the shell widget.

## Regular Expression Functions

The following functions scan strings using a form of regular expressions. Unlike the regular expressions supported by `ed(1)` or `egrep(1)`, these functions use a regular expression notation consisting of:

Table 5-2 Regular Expression Notation

| Element                     | Meaning                                                                                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>c</code>              | Match the character <code>c</code>                                                                                                                                                                       |
| <code>[&lt;set&gt;]</code>  | Match any character in <code>&lt;set&gt;</code> (where <code>&lt;set&gt;</code> is one or more characters concatenated into a string; range expressions, such as <code>[a-z]</code> , are not supported) |
| <code>[!&lt;set&gt;]</code> | Match any character not in <code>&lt;set&gt;</code>                                                                                                                                                      |
| <code>*</code>              | Match any character(s) (one or more)                                                                                                                                                                     |
| <code>^</code>              | When the circumflex is the first character in the regular expression, the match must start at <i>curp</i>                                                                                                |

### *strexp*

```
#include <Xol/regex.h>
char *strexp(void);
```

The `strexp()` function returns the pointer of the last character in a match found following a `strexp()` or `strrexp()` function call.

### *strexp*

```
#include <Xol/regex.h>
char *strexp(
 char *string,
 char *curp,
 char *expression);
```

The `strexp()` function performs a regular expression forward scan of *string* for *expression* starting at *curp*.

NULL is returned if *expression* cannot be found in *string*; otherwise, a pointer to the first character in the substring that matches *expression* is returned. The `strexp()` function can be used to get the pointer to the last character in the match.

## Regular Expression Functions

### *strrexp*

```
#include <Xol/regexp.h>
char *strrexp(
 char *string,
 char *curp,
 char *expression);
```

The `strrexp()` function performs a regular expression backward scan of *string* for *expression* starting at *curp*.

NULL is returned if *expression* cannot be found in *string*; otherwise, a pointer to the first character in the substring that matches *expression* is returned. The `strrexp()` function can be used to get the pointer to the last character in the match.

### *See Also*

“Buffer Functions” on page 95,  
“TextField Functions” on page 686.

## Text Buffer Functions

A *TextBuffer* is a data structure used by every single-byte *TextEdit* widget to store and manipulate its data. For internationalized *TextBuffers*, see page 176. The functions in this section can be used to manipulate a *TextBuffer*.

### TextLocation Structure

A number of the functions in this section refer to a `TextLocation` structure. It is defined as follows:

```
typedef struct _TextLocation {
 TextLine line;
 TextPosition offset;
 BufferElement *buffer;
} TextLocation;
```

### AllocateTextBuffer

```
#include <Xol/textbuff.h>

TextBuffer *AllocateTextBuffer(
 char *filename,
 TextUpdateFunction f,
 XtPointer d);
```

`AllocateTextBuffer()` allocates a new *TextBuffer* structure, initializes the members of the structure, does one more step described below, and returns a pointer to the newly allocated structure.

It is possible to register one or more text update functions (of type `TextUpdateFunction`) with a *TextBuffer*. As the name suggests, the text update functions are invoked by the toolkit when the *TextBuffer* is updated. In the course of registering a text update function, a possibly NULL client data (of type `XtPointer`) must be provided with the function. The client data is passed to the associated text update function when the function is invoked by the toolkit. See “`ReplaceBlockInTextBuffer`” on page 172 for more details of the `TextUpdateFunction`.

The argument *f* above is a text update function that together with its associated client data *d* is registered with the newly allocated *TextBuffer*, before `AllocateTextBuffer()` returns. The programmer must use

*Text Buffer Functions*

FreeTextBuffer() function to free the TextBuffer. The *filename* argument is used by the SaveTextBuffer() function (see page 174) if it is called with a NULL *filename* argument.

***BackwardScanTextBuffer***

```
#include <Xol/textbuff.h>
ScanResult BackwardScanTextBuffer(
 TextBuffer *text,
 char *exp,
 TextLocation *location);
```

BackwardScanTextBuffer() scans towards the beginning of the buffer for a given *expression* in the TextBuffer starting at *location*. The *exp* string is interpreted as described in “Regular Expression Functions” on page 161. A ScanResult is returned, which indicates:

- SCAN\_NOTFOUND    The scan wrapped without finding a match.
- SCAN\_WRAPPED    A match was found at a location after the start location.
- SCAN\_FOUND       A match was found at a location before the start location.
- SCAN\_INVALID     Either the *location* or the *exp* was invalid.

***CopyTextBufferBlock***

```
#include <Xol/textbuff.h>
int CopyTextBufferBlock(
 TextBuffer *text,
 char *buffer,
 TextPosition start_position,
 TextPosition end_position);
```

CopyTextBufferBlock() copies a text block from the *text* TextBuffer into *buffer*. The block is defined as the characters between *start\_position* and *end\_position* inclusive. It returns the number of bytes copied; if the parameters are invalid, the return value is zero.

---

**Note** – The storage for the copy is allocated by the caller. It is the responsibility of the caller to ensure that enough storage is allocated to copy *end\_position* – *start\_position* + 1 bytes.

---

### *EndCurrentTextBufferWord*

```
#include <Xol/textbuff.h>
TextLocation EndCurrentTextBufferWord(
 TextBuffer *textBuffer,
 TextLocation current);
```

`EndCurrentTextBufferWord()` locates the end of a word in the `TextBuffer` relative to a given *current* location. The function returns the location of the end of the current word. Note: this return value will equal the given current value if the current location is already at the end of a word.

### *FreeTextBuffer*

```
#include <Xol/textbuff.h>
void FreeTextBuffer(
 TextBuffer *text,
 TextUpdateFunction f,
 XtPointer d);
```

`FreeTextBuffer()` deallocates storage associated with a given `TextBuffer`. Note: the storage is not actually freed if the `TextBuffer` is still associated with other update function/data pairs. See “`ReplaceBlockInTextBuffer`” on page 172 for more details of the `TextUpdateFunction`.

### *ForwardScanTextBuffer*

```
#include <Xol/textbuff.h>
ScanResult ForwardScanTextBuffer(TextBuffer *text,
 char *exp,
 TextLocation *location);
```

`ForwardScanTextBuffer()` scans towards the end of the buffer for a given expression in the `TextBuffer` starting at *location*. The *exp* string is interpreted as described in “Regular Expression Functions” on page 161. A `ScanResult` is returned, which indicates:

|                            |                                                            |
|----------------------------|------------------------------------------------------------|
| <code>SCAN_NOTFOUND</code> | The scan wrapped without finding a match.                  |
| <code>SCAN_WRAPPED</code>  | A match was found at a location before the start location. |
| <code>SCAN_FOUND</code>    | A match was found at a location after the start location.  |
| <code>SCAN_INVALID</code>  | Either the location or the expression was invalid.         |

### *GetTextBufferBlock*

```
#include <Xol/textbuff.h>
char *GetTextBufferBlock(
 TextBuffer *text,
 TextLocation start_location,
 TextLocation end_location);
```

`GetTextBufferBlock()` retrieves a text block from the *text* TextBuffer. The block is defined as the characters between *start\_location* and *end\_location* inclusive. It returns a pointer to a string containing the copy. If the parameters are invalid, NULL is returned.

---

**Note** – The storage for the copy is allocated by this routine. It is the responsibility of the caller to free this storage when it becomes dispensable.

---

### *GetTextBufferBuffer*

```
#include <Xol/textbuff.h>
Buffer *GetTextBufferBuffer(
 TextBuffer *text,
 TextLine line);
```

`GetTextBufferBuffer()` retrieves a pointer to the Buffer stored in TextBuffer *text* for *line*. This pointer is volatile; subsequent calls to any TextBuffer routine may make it invalid. If a more permanent copy of this Buffer is required, the `CopyTextBufferBlock()` function (see page 164) can be used to create a private copy of it.

### *GetTextBufferChar*

```
#include <Xol/textbuff.h>
int GetTextBufferChar(
 TextBuffer *text,
 TextLocation location);
```

`GetTextBufferChar()` retrieves a character stored in the *text* TextBuffer at *location*. It returns either the character itself or EOF if location is outside the range of valid locations within the TextBuffer.



### *GetTextBufferLine*

```
#include <Xol/textbuff.h>
char *GetTextBufferLine(
 TextBuffer *text,
 TextLine lineindex);
```

`GetTextBufferLine()` retrieves the contents of string containing the copy of the contents of the line or NULL if the *lineindex* is outside the range of valid lines in *text*.

---

**Note** – The storage for the copy is allocated by this routine. It is the responsibility of the caller to free this storage when it becomes dispensable.

---

### *GetTextBufferLocation*

```
#include <Xol/textbuff.h>
char *GetTextBufferLocation(
 TextBuffer *text,
 TextLine line_number,
 TextLocation *location);
```

`GetTextBufferLocation()` retrieves the contents of the given line within the `TextBuffer`. It returns a pointer to the character string. If the line number is invalid, a NULL pointer is returned. If a non-NULL `TextLocation` pointer is supplied in the argument list, the contents of this structure are modified to reflect the values corresponding to the given line.

### *IncrementTextBufferLocation*

```
#include <Xol/textbuff.h>
TextLocation IncrementTextBufferLocation(
 TextBuffer *text,
 TextLocation location,
 TextLine line,
 TextPosition offset);
```

`IncrementTextBufferLocation()` increments a *location* by either *line* lines and/or *offset* characters. It returns the new *location*. If *line* or *offset* are negative, the function performs a decrement operation. If the starting location or the resulting location is invalid, the starting location is returned without modification; otherwise, the new location is returned.

### *LastTextBufferLocation*

```
#include <Xol/textbuff.h>
TextLocation LastTextBufferLocation(
 TextBuffer *text);
```

LastTextBufferLocation() returns the last valid TextLocation in the TextBuffer associated with *text*.

### *LastTextBufferPosition*

```
#include <Xol/textbuff.h>
TextPosition LastTextBufferPosition(
 TextBuffer *text);
```

LastTextBufferPosition() returns the last valid TextPosition in the TextBuffer associated with *text*.

### *LineOfPosition*

```
#include <Xol/textbuff.h>
int LineOfPosition(
 TextBuffer *text,
 TextPosition position);
```

The LineOfPosition() function returns the line number in which *position* occurs. If *position* is invalid, it returns EOF.

### *LocationOfPosition*

```
#include <Xol/textbuff.h>
TextLocation LocationOfPosition(
 TextBuffer *text,
 TextPosition position);
```

LocationOfPosition() translates a *position* in the *text* TextBuffer to a TextLocation (see page 163). It returns the translated TextLocation. If the *position* is invalid, the *buffer* pointer in the TextLocation struct is set to NULL and the *line* and *offset* members in the TextLocation struct are set the last valid location in the TextBuffer; otherwise, *buffer* is set to a non-NULL (though useless) value.

### *NextLocation*

```
#include <Xol/textbuff.h>
TextLocation NextLocation(
 TextBuffer *textBuffer,
 TextLocation current);
```

`NextLocation()` returns the `TextLocation` that follows the given *current* location in a `TextBuffer`. If the current location points to the end of the `TextBuffer`, this function wraps to the beginning of the `TextBuffer`.

### *NextTextBufferWord*

```
#include <Xol/textbuff.h>
TextLocation NextTextBufferWord(
 TextBuffer *textBuffer,
 TextLocation current);
```

`NextTextBufferWord()` locates the beginning of the next word from a given *current* location in a `TextBuffer`. If the current location is within the last word in the `TextBuffer`, the function wraps to the beginning of the `TextBuffer`.

### *PositionOfLine*

```
#include <Xol/textbuff.h>
TextPosition PositionOfLine(
 TextBuffer *text,
 TextLine lineindex);
```

The `PositionOfLine()` function returns the `TextPosition` corresponding to *lineindex*. If *lineindex* is invalid, it returns EOF.

### *PositionOfLocation*

```
#include <Xol/textbuff.h>
TextPosition PositionOfLocation(
 TextBuffer *text,
 TextLocation location);
```

The `PositionOfLocation()` function returns the `TextPosition` corresponding to *location*. If *location* is invalid, it returns EOF.

### *PreviousLocation*

```
#include <Xol/textbuff.h>
TextLocation PreviousLocation(
 TextBuffer *textBuffer,
 TextLocation current);
```

The `PreviousLocation()` function returns the `TextLocation` (see page 163) that precedes the given *current* location in a `TextBuffer`. If the current location points to the beginning of the `TextBuffer`, this function wraps to the end of the `TextBuffer`.

### *PreviousTextBufferWord*

```
#include <Xol/textbuff.h>
TextLocation PreviousTextBufferWord(
 TextBuffer *textBuffer,
 TextLocation current);
```

`PreviousTextBufferWord()` locates the beginning of a word in a `TextBuffer` relative to a given *current* location. It returns the location of the beginning of the word that precedes the given current location. If the *current* location is within a word, this function returns beginning of the current word.

### *ReadFileIntoTextBuffer*

```
#include <Xol/textbuff.h>
TextBuffer *ReadFileIntoTextBuffer(
 char *filename,
 TextUpdateFunction f,
 XtPointer d);
```

`ReadFileIntoTextBuffer()` allocates a new `TextBuffer` and reads the file denoted by the given *filename* into it. The supplied text update function *f* and the client data *d* are associated with the newly allocated `TextBuffer`. The function returns a pointer to this `TextBuffer`. See “`ReplaceBlockInTextBuffer`” on page 172 for more details of the `TextUpdateFunction`.

### ***ReadStringIntoTextBuffer***

```
#include <Xol/textbuff.h>
TextBuffer *ReadStringIntoTextBuffer(
 char *string,
 TextUpdateFunction f,
 XtPointer d);
```

`ReadStringIntoTextBuffer()` allocates a new `TextBuffer` and copies the given *string* into it. The supplied `TextUpdateFunction` and data pointer are associated with this `TextBuffer`. The function returns a pointer to this `TextBuffer`. See “`ReplaceBlockInTextBuffer`” on page 172 for more details of the `TextUpdateFunction`.

### ***RegisterTextBufferScanFunctions***

```
#include <Xol/textbuff.h>
void RegisterTextBufferScanFunctions(
 char *(*forward)(),
 char *(*backward)());
```

`RegisterTextBufferScanFunctions()` provides the capability to replace the default scan functions used by the `ForwardScanTextBuffer()` and `BackwardScanTextBuffer()` functions. These functions are called as:

```
(*forward)(string, curp, expression);
(*backward)(string, curp, expression);
```

and are responsible for returning either a pointer to the beginning of a match for the expression or `NULL`. Calling `RegisterTextBufferScanFunctions()` with `NULL` function pointers reinstates the default regular expression facility, as described in “`Regular Expression Functions`” on page 161.

### ***RegisterTextBufferWordDefinition***

```
#include <Xol/textbuff.h>
void RegisterTextBufferWordDefinition(
 int (*word_definition)());
```

`RegisterTextBufferWordDefinition()` provides the capability to replace the default word definition function used by the `TextBuffer` functions in this section. This function is called as:

```
(*word_definition)(c);
```

The function is responsible for returning nonzero if the character *c* is considered a character that can occur in a word, and zero otherwise. Calling `RegisterTextBufferWordDefinition()` with NULL reinstates the default word definition, which allows the following set of characters: a-z, A-Z, 0-9\_

### *RegisterTextBufferUpdate*

```
#include <Xol/textbuff.h>

void RegisterTextBufferUpdate(
 TextBuffer *text,
 TextUpdateFunction f,
 XtPointer d);
```

`RegisterTextBufferUpdate()` associates the `TextUpdateFunction` *f* and data pointer *d* with the given `TextBuffer` *text*. This update function will be called whenever an update operation is performed on the `TextBuffer`. See “`ReplaceBlockInTextBuffer`” on page 172 for more details of the `TextUpdateFunction`.

---

**Note** – Calling `RegisterTextBufferUpdate()` increments a reference count mechanism used to determine when to actually free the `TextBuffer`. Calling the function with a NULL value for the function circumvents this mechanism.

---

### *ReplaceBlockInTextBuffer*

```
#include <Xol/textbuff.h>

EditResult ReplaceBlockInTextBuffer(
 TextBuffer *text,
 TextLocation *startloc,
 TextLocation *endloc,
 char *string,
 TextUpdateFunction f,
 XtPointer d);
```

`ReplaceBlockInTextBuffer()` updates the contents of the `TextBuffer` *text*. The characters stored between *startloc* (inclusive) and *endloc* (exclusive) are deleted and the string is inserted after *startloc*. If the edit succeeds and if `TextUpdateFunction` *f* is associated with `TextBuffer` *text*, then *f* is called with the following parameters:

```
(*f)(XtPointer d, TextBuffer *text, EDIT_SUCCESS)
```

---

## Text Buffer Functions

All the other text update functions associated with TextBuffer text are called with the following parameters:

```
(XtPointer d, TextBuffer *text, EDIT_FAILURE)
```

ReplaceBlockInTextBuffer() stores the details of the editing operation it performs in *text->deleted* and *text->insert* TextUndoItem structures. The contents of these structures may be used for implementing an Undo mechanism. The hints provided in *text->deleted.hint* and *text->insert.hint* are an inclusive OR of:

```
#define TEXT_BUFFER_NOP (0)
#define TEXT_BUFFER_DELETE_START_LINE (1L<<0)
#define TEXT_BUFFER_DELETE_START_CHARS (1L<<1)
#define TEXT_BUFFER_DELETE_END_LINE (1L<<2)
#define TEXT_BUFFER_DELETE_END_CHARS (1L<<3)
#define TEXT_BUFFER_DELETE_JOIN_LINE (1L<<4)
#define TEXT_BUFFER_DELETE_SIMPLE (1L<<5)
#define TEXT_BUFFER_INSERT_SPLIT_LINE (1L<<6)
#define TEXT_BUFFER_INSERT_LINE (1L<<7)
#define TEXT_BUFFER_INSERT_CHARS (1L<<8)
```

The meaning of each of these values is described below:

---

|                                |                                                                            |
|--------------------------------|----------------------------------------------------------------------------|
| TEXT_BUFFER_NOP                | No edit operation.                                                         |
| TEXT_BUFFER_DELETE_START_LINE  | The deleted block started at beginning of some line.                       |
| TEXT_BUFFER_DELETE_START_CHARS | The deleted block did not start at the beginning of some line.             |
| TEXT_BUFFER_DELETE_END_LINE    | The end of the deleted block coincided with the end of some line.          |
| TEXT_BUFFER_DELETE_END_CHARS   | Some characters were deleted from the end of some line.                    |
| TEXT_BUFFER_DELETE_JOIN_LINE   | Some characters were deleted and two lines were joined into a single line. |
| TEXT_BUFFER_DELETE_SIMPLE      | The whole of the deleted block was confined to a single line.              |

---

*Text Buffer Functions*

---

|                               |                                                                                            |
|-------------------------------|--------------------------------------------------------------------------------------------|
| TEXT_BUFFER_INSERT_SPLIT_LINE | One line was split into two lines and some characters were inserted at the split location. |
| TEXT_BUFFER_INSERT_LINE       | A line was inserted without splitting an existing line.                                    |
| TEXT_BUFFER_INSERT_CHARS      | Some characters were inserted at the beginning of some existing line.                      |

---

***ReplaceCharInTextBuffer***

```
#include <Xol/textbuff.h>
EditResult ReplaceCharInTextBuffer(
 TextBuffer *text,
 TextLocation *location,
 int c,
 TextUpdateFunction f,
 XtPointer d);
```

ReplaceCharInTextBuffer() replaces the character in the TextBuffer *text* at *location* with the character *c*. Everything described in “ReplaceBlockInTextBuffer” on page 172 about text update functions also applies to this function.

***SaveTextBuffer***

```
#include <Xol/textbuff.h>
SaveResult SaveTextBuffer(
 TextBuffer *text,
 char *filename);
```

SaveTextBuffer() writes the contents of the *text* TextBuffer to the file *filename*. If *filename* is NULL, it uses the *filename* argument that was given to the AllocateTextBuffer() function (see page 163).

SaveTextBuffer() returns a SaveResult, which can be SAVE\_FAILURE or SAVE\_SUCCESS.



### *StartCurrentTextBufferWord*

```
#include <Xol/textbuff.h>
TextLocation StartCurrentTextBufferWord(
 TextBuffer *textBuffer,
 TextLocation current);
```

`StartCurrentTextBufferWord()` locates the beginning of a word in the `TextBuffer` relative to a given *current* location. The function returns the location of the beginning of the current word. Note: this return value will equal the given current value if the current location is the beginning of a word.

### *UnregisterTextBufferUpdate*

```
#include <Xol/textbuff.h>
int UnregisterTextBufferUpdate(
 TextBuffer *text,
 TextUpdateFunction f,
 XtPointer d);
```

The `UnregisterTextBufferUpdate()` function disassociates the `TextUpdateFunction` *f* and data pointer *d* with the given `TextBuffer` *text*. If the function/data pointer pair is not associated with the given `TextBuffer`, zero is returned; otherwise, the association is dissolved and one is returned. See “`ReplaceBlockInTextBuffer`” on page 172 for more details of the `TextUpdateFunction`.

### *TextBuffer Macros*

The macros described in “`Buffer Macros`” on page 95 can also be used with the text buffer functions in this section.

### *See Also*

Buffer Functions on page 95,  
Regular Expression Functions on page 161,  
“Text Buffer Functions for Internationalization” on page 176.

*Text Buffer Functions for Internationalization*

***Text Buffer Functions for Internationalization***

The text buffer functions in this section provide multibyte equivalents to the single-byte OLIT text buffer functions in the previous section.

***OlAllocateTextBuffer***

```
#include <Xol/Oltextbuff.h>
OlTextBufferPtr OlAllocateTextBuffer(
 OlStrRep strrep,
 char *filename,
 TextUpdateFunction update_func,
 XtPointer data);
```

***Arguments***

|                    |                           |
|--------------------|---------------------------|
| <i>strrep</i>      | Specifies the text format |
| <i>filename</i>    | Specifies the filename    |
| <i>update_func</i> | The update function       |
| <i>data</i>        | Client data               |

`OlAllocateTextBuffer()` allocates a new `TextBuffer` structure, initializes the members of the structure, does one more step described below, and returns a pointer to the newly allocated structure.

It is possible to register one or more text update functions (of type `TextUpdateFunction`) with a `TextBuffer`. As the name suggests, the text update functions are invoked by the toolkit when the `TextBuffer` is updated. In the course of registering a text update function, a client data (of type `XtPointer`) must be provided with the function; this client data can be `NULL`. The client data is passed to the associated text update function when the function is invoked by the toolkit (See “`OlReplaceBlockInTextBuffer`” on page 197).

The argument *update\_func* is a text update function that, together with its associated client data *data*, is registered with the newly allocated `TextBuffer`, before `AllocateTextBuffer()` returns. The programmer must use `OlFreeTextBuffer()` function to free the `TextBuffer`.

---

## Text Buffer Functions for Internationalization

The *strrep* argument can have the following values:

| Value         | Meaning                              |
|---------------|--------------------------------------|
| OL_SB_STR_REP | Single-byte character representation |
| OL_WC_STR_REP | Wide character representation        |
| OL_MB_STR_REP | Multibyte character representation   |

### See Also

“OIFreeTextBuffer” on page 180,  
 “OIReadFileIntoTextBuffer” on page 193,  
 “OIReadStringIntoTextBuffer” on page 194.

### OIBackwardScanTextBuffer

```
#include <Xol/OItextbuff.h>

ScanResult OIBackwardScanTextBuffer(
 OITextBufferPtr text,
 OIStr exp,
 TextLocation *location);
```

### Arguments

|                 |                                   |
|-----------------|-----------------------------------|
| <i>text</i>     | The text buffer                   |
| <i>exp</i>      | The expression to scan for        |
| <i>location</i> | The location to start scanning at |

OIBackwardScanTextBuffer() scans towards the beginning of the buffer for a given expression in the OITextBuffer starting at *location*. The ScanResult can have the following values:

|               |                                                           |
|---------------|-----------------------------------------------------------|
| SCAN_NOTFOUND | The scan wrapped without finding a match.                 |
| SCAN_WRAPPED  | A match was found at a location after the start location  |
| SCAN_FOUND    | A match was found at a location before the start location |
| SCAN_INVALID  | Either the location or the expression was invalid         |

### See Also

“OIForwardScanTextBuffer” on page 179.

*Text Buffer Functions for Internationalization*

***OlCopyTextBufferBlock***

```
#include <Xol/Oltextbuff.h>

int OlCopyTextBufferBlock(
 OlTextBufferPtr text,
 OlStr outbuffer,
 int num_bytes,
 TextPosition start_position,
 TextPosition end_position);
```

**Arguments**

|                       |                                  |
|-----------------------|----------------------------------|
| <i>text</i>           | The text buffer                  |
| <i>outbuffer</i>      | The buffer to output the text to |
| <i>num_bytes</i>      | Size of the text block           |
| <i>start_position</i> | Beginning of text block          |
| <i>end_position</i>   | End of text block                |

*OlCopyTextBufferBlock()* retrieves a text block from the *OlTextBuffer*. The block is defined as the characters between *start\_position* and *end\_position* inclusive. If *num\_bytes* is not sufficient, *OlCopyTextBufferBlock()* returns -1; otherwise, it returns actual bytes used.

---

**Note** – The storage for the copy is allocated by the caller. It is the responsibility of the caller to ensure that enough storage is allocated to copy (*end\_position* – *start\_position*) + (bytes to store null character).

---

**See Also**

- “*OlAllocateTextBuffer*” on page 176,
- “*OlRegisterTextBufferUpdate*” on page 197,
- “*OlGetTextBufferCharAtLoc*” on page 182,
- “*OlGetTextBufferLine*” on page 183.

***OlEndCurrentTextBufferWord***

```
#include <Xol/Oltextbuff.h>

TextLocation *OlEndCurrentTextBufferWord(
 OlTextBufferPtr text,
 TextLocation *current);
```

---

## Text Buffer Functions for Internationalization

### Arguments

|                |                            |
|----------------|----------------------------|
| <i>text</i>    | The text buffer            |
| <i>current</i> | Specifies current location |

`OlEndCurrentTextBufferWord()` locates the end of a word in the `OlTextBuffer` relative to a given current location. It returns the location of the end of the current word. The return value will equal the given current value if the current location is already at the end of a word. If the location is not in a word, it returns the end of the “not word” region it is in.

---

**Note** – The location passed to `OlEndCurrentTextBufferWord()` is modified. It contains the end of the current buffer word (or “not word”) at the end of the call.

---

### See Also

“`OlPreviousTextBufferWord`” on page 192,  
“`OlNextTextBufferWord`” on page 189.

### *OlForwardScanTextBuffer*

```
#include <Xol/Oltextbuff.h>

ScanResult OlForwardScanTextBuffer(
 OlTextBufferPtr text,
 OlStr exp,
 TextLocation *location);
```

### Arguments

|                 |                                      |
|-----------------|--------------------------------------|
| <i>text</i>     | The text buffer                      |
| <i>exp</i>      | Specifies the expression to scan for |
| <i>location</i> | The location to start scanning at    |

`OlForwardScanTextBuffer()` scans towards the end of the buffer, for a given expression in the `OlTextBuffer` starting at `location`. A `ScanResult` is returned, which indicates the following:

|                            |                                                            |
|----------------------------|------------------------------------------------------------|
| <code>SCAN_NOTFOUND</code> | The scan wrapped without finding a match.                  |
| <code>SCAN_WRAPPED</code>  | A match was found at a location before the start location. |
| <code>SCAN_FOUND</code>    | A match was found at a location after the start location.  |
| <code>SCAN_INVALID</code>  | Either the location or the expression was invalid          |

*Text Buffer Functions for Internationalization*

**See Also**

“OlBackwardScanTextBuffer” on page 177.

***OlFreeTextBuffer***

```
#include <Xol/Oltextbuff.h>

void OlFreeTextBuffer(
 OlTextBufferPtr text,
 TextUpdateFunction update_func,
 XtPointer data);
```

**Arguments**

|             |                         |
|-------------|-------------------------|
| <i>text</i> | The text buffer to free |
| <i>exp</i>  | The update function     |
| <i>data</i> | Data                    |

OlFreeTextBuffer() deallocates storage associated with a given OlTextBuffer. See “OlReplaceBlockInTextBuffer” on page 197 for more details of the TextUpdateFunction.

---

**Note** - The storage is not actually freed if the OlTextBuffer is still associated with other update function/data pairs.

---

**See Also**

“OlAllocateTextBuffer” on page 176,  
 “OlRegisterTextBufferUpdate” on page 197.

***OlGetTextBufferBlock***

```
#include <Xol/Oltextbuff.h>

OlStr OlGetTextBufferBlock(
 OlTextBufferPtr text,
 TextLocation *start_location,
 TextLocation *end_location);
```

---

## Text Buffer Functions for Internationalization

### Arguments

|                       |                                        |
|-----------------------|----------------------------------------|
| <i>text</i>           | The text buffer to retrieve block from |
| <i>start_location</i> | Start of the text block                |
| <i>end_location</i>   | End of the text block                  |

`OlGetTextBufferBlock()` retrieves a text block from the *text* `TextBuffer`. The block is defined as the characters between *start\_location* and *end\_location* inclusive. It returns a pointer to a string containing the copy. If the parameters are invalid NULL is returned.

---

**Note** – The storage for the copy is allocated by this routine. It is the responsibility of the caller to free this storage when it becomes dispensable.

---

### See Also

“`OlAllocateTextBuffer`” on page 176,  
“`OlRegisterTextBufferUpdate`” on page 197,  
“`OlGetTextBufferCharAtLoc`” on page 182,  
“`OlGetTextBufferLine`” on page 183.

### *OlGetTextBufferBuffer*

```
#include <Xol/Oltextbuff.h>

Buffer *OlGetTextBufferBuffer(
 OlTextBufferPtr text,
 TextLine line);
```

### Arguments

|             |                                    |
|-------------|------------------------------------|
| <i>text</i> | The text buffer                    |
| <i>line</i> | The line to retrieve a pointer for |

`OlGetTextBufferBuffer()` retrieves a pointer to the `Buffer` stored in `OlTextBuffer` *text* for a line. This pointer is volatile; subsequent calls to any `OlTextBuffer` routine may make it invalid. If a more permanent copy of this `Buffer` is required the buffer utility, `CopyBuffer()` can be used to create a private copy of it.

---

## Text Buffer Functions for Internationalization

### **See Also**

“OlGetTextBufferBlock” on page 180,  
“OlAllocateTextBuffer” on page 176,  
“OlRegisterTextBufferUpdate” on page 197.

### ***OlGetTextBufferCharAtLoc***

```
#include <Xol/Oltextbuff.h>

OlStr OlGetTextBufferCharAtLoc(
 OlTextBufferPtr text,
 TextLocation *location);
```

### **Arguments**

|                 |                                          |
|-----------------|------------------------------------------|
| <i>text</i>     | The text buffer                          |
| <i>location</i> | The location at which to get a character |

`OlGetTextBufferCharAtLoc()` retrieves a character stored in the `OlTextBuffer` at *location*. It returns either the pointer to the character itself or NULL if *location* is outside the range of valid locations within the `OlTextBuffer`.

### **See Also**

“OlAllocateTextBuffer” on page 176,  
“OlRegisterTextBufferUpdate” on page 197,  
“OlGetTextBufferBlock” on page 180,  
“OlGetTextBufferLine” on page 183.

### ***OlGetTextBufferFileName***

```
#include <Xol/Oltextbuff.h>

String OlGetTextBufferFileName(
 OlTextBufferPtr text);
```

### **Arguments**

|             |                                              |
|-------------|----------------------------------------------|
| <i>text</i> | The text buffer for which to get a filename. |
|-------------|----------------------------------------------|

`OlGetTextBufferFileName()` returns the file name associated with the buffer. Otherwise, it returns NULL.



***OlGetTextBufferLine***

```
#include <Xol/Oltextbuff.h>

OlStr OlGetTextBufferLine(
 OlTextBufferPtr text,
 TextLine lineindex);
```

**Arguments**

|                  |                                                |
|------------------|------------------------------------------------|
| <i>text</i>      | The text buffer                                |
| <i>lineindex</i> | Index of the line to retrieve the contents for |

`OlGetTextBufferLine()` retrieves the contents of line from the `OlTextBuffer`. It returns a pointer to a string containing the copy of the contents of the line or NULL if the line is outside the range of valid lines in text.

---

**Note** – The storage for the copy is allocated by this routine. It is the responsibility of the caller to free this storage when it becomes dispensable.

---

**See Also**

“`OlAllocateTextBuffer`” on page 176,  
“`OlRegisterTextBufferUpdate`” on page 197,  
“`OlGetTextBufferCharAtLoc`” on page 182,  
“`OlGetTextBufferBlock`” on page 180.

***OlGetTextUndoDeleteItem***

```
#include <Xol/Oltextbuff.h>

OlTextUndoItem OlGetTextUndoDeleteItem(
 OlTextBufferPtr text);
```

**Arguments**

|             |                                                        |
|-------------|--------------------------------------------------------|
| <i>text</i> | The text buffer for which to get the undo delete item. |
|-------------|--------------------------------------------------------|

`OlGetTextUndoDeleteItem()` returns a `OlTextUndoItem` struct containing the value of “deleted” undo item. A copy of the deleted string is provided in the returned struct.











### ***OlNextLocation***

```
#include <Xol/Oltextbuff.h>

TextLocation *OlNextLocation(
 OlTextBufferPtr text,
 TextLocation *current);
```

#### ***Arguments***

|                |                      |
|----------------|----------------------|
| <i>text</i>    | The text buffer      |
| <i>current</i> | The current location |

`OlNextLocation()` returns the pointer to the `TextLocation` that follows the given current location in an `OlTextBuffer`. If the current location points to the end of the `OlTextBuffer`, this function wraps to the beginning of the `OlTextBuffer`.

---

**Note** – The location passed to this function is modified. It contains the next location at the end of the call.

---

#### ***See Also***

“`OlPreviousLocation`” on page 192.

### ***OlNextTextBufferWord***

```
#include <Xol/Oltextbuff.h>

TextLocation *OlNextTextBufferWord(
 OlTextBufferPtr text,
 TextLocation *current);
```

#### ***Arguments***

|                |                      |
|----------------|----------------------|
| <i>text</i>    | The text buffer      |
| <i>current</i> | The current location |

`OlNextTextBufferWord()` locates the beginning of the next word from a given current location in an `OlTextBuffer`. If the current location is within the last word in the `OlTextBuffer`, the function wraps to the beginning of the `OlTextBuffer`.

---

*Text Buffer Functions for Internationalization*

---

**Note** – The location passed to this function is modified. It contains the start of the next buffer word at the end of the call.

---

**See Also**

“OlPreviousTextBufferWord” on page 192,  
 “OlStartCurrentTextBufferWord” on page 201.

***OlNumBytesInTextBufferLine***

```
#include <Xol/Oltextbuff.h>

int OlNumBytesInTextBufferLine(
 OlTextBufferPtr text,
 TextLine line);
```

**Arguments**

*text*                                    The text buffer  
*line*                                    The line to get the number of bytes for

OlNumBytesInTextBufferLine() returns the number of bytes in *line*.

***OlNumCharsInTextBufferLine***

```
#include <Xol/Oltextbuff.h>

int OlNumCharsInTextBufferLine(
 OlTextBufferPtr text,
 TextLine line);
```

**Arguments**

*text*                                    The text buffer  
*line*                                    The line for which to get the number of chars

OlNumCharsInTextBufferLine() returns the number of characters in the specified *line*.



---

## Text Buffer Functions for Internationalization

### *OlNumUnitsInTextBufferLine*

```
#include <Xol/Oltextbuff.h>

int OlNumUnitsInTextBufferLine(
 OlTextBufferPtr text,
 TextLine line);
```

#### **Arguments**

*text*                                    The text buffer  
*line*                                    The line for which to get the number of units

`OlNumUnitsInTextBufferLine()` returns the number of units in the specified *line*.

### *OlPositionOfLine()*

```
#include <Xol/Oltextbuff.h>

TextPosition OlPositionOfLine(
 OlTextBufferPtr text,
 TextLine lineindex);
```

#### **Arguments**

*text*                                    The text buffer  
*lineindex*                                The line index to translate into a text position

`OlPositionOfLine()` translates a *lineindex* in the `OlTextBuffer` to a `TextPosition`. It returns the translated `TextPosition` or EOF if the *lineindex* is invalid.

### *OlPositionOfLocation*

```
#include <Xol/Oltextbuff.h>

TextPosition OlPositionOfLocation(
 OlTextBufferPtr text,
 TextLocation *location);
```

#### **Arguments**

*text*                                    The text buffer  
*location*                                The location to translate to a `TextPosition`

---

## Text Buffer Functions for Internationalization

`OlPositionOfLocation()` translates a location in the `OlTextBuffer` to a `TextPosition`. It returns the translated `TextPosition` or EOF if the *location* is invalid.

### *OlPreviousLocation*

```
#include <Xol/Oltextbuff.h>

TextLocation *OlPreviousLocation(
 OlTextBufferPtr text,
 TextLocation *current);
```

#### **Arguments**

|                |                      |
|----------------|----------------------|
| <i>text</i>    | The text buffer      |
| <i>current</i> | The current location |

`OlPreviousLocation()` function the pointer to the `TextLocation` that precedes the given current location in a `OlTextBuffer`. If the current location points to the beginning of the `OlTextBuffer`, this function wraps to the end of the `OlTextBuffer`.

---

**Note** – The current location is modified. It contains the previous location at the end of the call.

---

#### **See Also**

“`OlNextLocation`” on page 189.

### *OlPreviousTextBufferWord*

```
#include <Xol/Oltextbuff.h>

TextLocation *OlPreviousTextBufferWord(
 OlTextBufferPtr text,
 TextLocation *current);
```

#### **Arguments**

|                |                      |
|----------------|----------------------|
| <i>text</i>    | The text buffer      |
| <i>current</i> | The current location |

---

### *Text Buffer Functions for Internationalization*

`OlPreviousTextBufferWord()` locates the beginning of a word in a `OlTextBuffer` relative to a given current location. It returns the location of the beginning of the word that precedes the given *current* location. If the current location is within a word, this function will skip over the current word. If the current word is the first word in the `OlTextBuffer`, the function wraps to the end of the `OlTextBuffer`.

---

**Note** – The location passed to this function is modified. It contains the start of the previous buffer word at the end of the call.

---

#### **See Also**

“`OlPreviousTextBufferWord`” on page 192

#### ***OlReadFileIntoTextBuffer***

```
#include <Xol/Oltextbuff.h>

OlTextBufferPtr OlReadFileIntoTextBuffer(
 OlStrRep strrep,
 char *filename,
 TextUpdateFunction update_func,
 XtPointer data);
```

#### **Arguments**

|                    |                                                                            |
|--------------------|----------------------------------------------------------------------------|
| <i>strrep</i>      | The string representation (OL_SB_STR_REP, OL_WC_STR_REP, or OL_MB_STR_REP) |
| <i>filename</i>    | The file to be read                                                        |
| <i>update_func</i> | The update function                                                        |
| <i>data</i>        | Data                                                                       |

`OlReadFileIntoTextBuffer()` reads the given file into a newly allocated `OlTextBuffer`. The supplied `TextUpdateFunction` and `data` pointer are associated with this `OlTextBuffer`. See “`OlReplaceBlockInTextBuffer`” on page 197 for more details of the `TextUpdateFunction`.

#### **See Also**

“`OlReadStringIntoTextBuffer`” on page 194.

*Text Buffer Functions for Internationalization*

***OlReadStringIntoTextBuffer***

```
#include <Xol/Oltextbuff.h>
OlTextBufferPtr OlReadStringIntoTextBuffer(
 OlStrRep strrep,
 char *string,
 TextUpdateFunction update_func,
 XtPointer data);
```

**Arguments**

|                    |                                                                            |
|--------------------|----------------------------------------------------------------------------|
| <i>strrep</i>      | The string representation (OL_SB_STR_REP, OL_WC_STR_REP, or OL_MB_STR_REP) |
| <i>string</i>      | The string to be read                                                      |
| <i>update_func</i> | The update function                                                        |
| <i>data</i>        | Data                                                                       |

`OlReadStringIntoTextBuffer()` copies the given string into a newly allocated `OlTextBuffer`. The supplied `TextUpdateFunction` and *data* pointer are associated with this `OlTextBuffer`. See “`OlReplaceBlockInTextBuffer`” on page 197 for more details of the `TextUpdateFunction`.

**See Also**

“`OlReadFileIntoTextBuffer`” on page 193.

***OlRegisterAllTextBufferScanFunctions***

```
#include <Xol/Oltextbuff.h>

void OlRegisterAllTextBufferScanFunctions(
 OlStrRep strrep,
 OlStrScanDefFunc forward_scan_func,
 OlStrScanDefFunc backward_scan_func);
```

**Arguments**

|                           |                                                                            |
|---------------------------|----------------------------------------------------------------------------|
| <i>strrep</i>             | The string representation (OL_SB_STR_REP, OL_WC_STR_REP, or OL_MB_STR_REP) |
| <i>forward_scan_func</i>  | The forward scan function to be used by all <code>OlTextBuffers</code>     |
| <i>backward_scan_func</i> | The backward scan function to be used by all <code>OlTextBuffers</code>    |

## *Text Buffer Functions for Internationalization*

The *forward\_scan\_func* and *backward\_scan\_func* arguments specify `OlStrScanDefFunc()` functions. `OlStrScanDefFunc` is defined as:

```
typedef XtPointer (*OlStrScanDefFunc)(
 OlStr string,
 OlStr curp,
 OlStr expression);
```

`OlRegisterAllTextBufferWordDefinition()` provides the capability to replace the text buffer functions used by all `OlTextBuffers`.

### *OlRegisterAllTextBufferWordDefinition*

```
#include <Xol/Oltextbuff.h>

void OlRegisterAllTextBufferWordDefinition(
 OlStrRep strrep,
 OlStrWordDefFunc word_definition_func);
```

#### **Arguments**

|                             |                                                                     |
|-----------------------------|---------------------------------------------------------------------|
| <i>strrep</i>               | The string representation                                           |
| <i>word_definition_func</i> | The word definition function used by all <code>OlTextBuffers</code> |

The *word\_definition\_func* argument specifies an `OlStrWordDefFunc()`, which is defined as:

```
typedef Boolean (*OlStrWordDefFunc)(OlStr rc);
```

`OlRegisterAllTextBufferWordDefinition()` provides the capability to replace the word definition function used by all `OlTextBuffers`. These functions are responsible for returning `TRUE` if the character that *rc* points to can occur in a word, and `FALSE` otherwise. Calling this function with `NULL` reinstates the default word definition function associated with the text format.

### *OlRegisterPerTextBufferScanFunctions*

```
#include <Xol/Oltextbuff.h>

void OlRegisterPerTextBufferScanFunctions(
 OlTextBufferPtr text,
 OlStrScanDefFunc forward_scan_func,
 OlStrScanDefFunc backward_scan_func);
```

## Text Buffer Functions for Internationalization

### Arguments

|                           |                                                                            |
|---------------------------|----------------------------------------------------------------------------|
| <i>strrep</i>             | The string representation (OL_SB_STR_REP, OL_WC_STR_REP, or OL_MB_STR_REP) |
| <i>forward_scan_func</i>  | The forward scan function used by OlTextBuffers                            |
| <i>backward_scan_func</i> | The backward scan function used by OlTextBuffers                           |

The arguments *forward\_scan\_func* and *backward\_scan\_func* specify OlStrScanDefFunc() functions. OlStrScanDefFunc is defined as:

```
typedef XtPointer (*OlStrScanDefFunc)(
 OlStr string,
 OlStr curp,
 OlStr expression);
```

OlRegisterPerTextBufferScanFunctions() provides the capability to replace the scan functions used by the OlForwardScanTextBuffer() and OlBackwardScanTextBuffer() functions, as applied to the passed OlTextBuffer only.

OlRegisterAllTextBufferScanFunctions() provides the capability to replace the scan functions used by the OlForwardScanTextBuffer() and OlBackwardScanTextBuffer() functions, as applied to all OlTextBuffers. These functions are responsible for returning either a pointer to the beginning of a match for the expression or NULL. Calling this procedure with NULL function pointers reinstates the default regular expression facility associated with the text format.

### OlRegisterPerTextBufferWordDefinition

```
#include <Xol/Oltextbuff.h>

void OlRegisterPerTextBufferWordDefinition(
 OlTextBufferPtr text,
 OlStrWordDefFunc word_definition_func);
```

### Arguments

|                             |                                  |
|-----------------------------|----------------------------------|
| <i>text</i>                 | The text buffer                  |
| <i>word_definition_func</i> | The new word definition function |

The *word\_definition\_func* argument specifies an OlStrWordDefFunc(), which is defined as:

```
typedef Boolean (*OlStrWordDefFunc)(OlStr rc);
```

---

## Text Buffer Functions for Internationalization

`OlRegisterPerTextBufferWordDefinition()` provides the capability to replace the word definition function used by the passed `OlTextBuffer`.

### *OlRegisterTextBufferUpdate*

```
#include <Xol/Oltextbuff.h>

void OlRegisterTextBufferUpdate(
 OlTextBufferPtr text,
 TextUpdateFunction update_func,
 XtPointer data);
```

#### **Arguments**

|                    |                     |
|--------------------|---------------------|
| <i>text</i>        | The text buffer     |
| <i>update_func</i> | The update function |
| <i>data</i>        | Data                |

`OlRegisterTextBufferUpdate()` associates the `TextUpdateFunction` *update\_func* and data pointer *data* with the given `OlTextBuffer` *text*. This update function will be called whenever an update operation is performed on the `OlTextBuffer`. See “`OlReplaceBlockInTextBuffer`” on page 197 for more details of the `TextUpdateFunction`.

---

**Note** – Calling this function increments a reference count mechanism used to determine when to actually free the `OlTextBuffer`. Calling the function with a NULL value for the function circumvents this mechanism.

---

#### **See Also**

“`OlUnregisterTextBufferUpdate`” on page 203,  
“`OlReadStringIntoTextBuffer`” on page 194,  
“`OlReadFileIntoTextBuffer`” on page 193.

### *OlReplaceBlockInTextBuffer*

```
#include <Xol/Oltextbuff.h>

EditResult OlReplaceBlockInTextBuffer(
 OlTextBufferPtr text,
 TextLocation *startloc,
 TextLocation *endloc,
 OlStr string,
```

**Text Buffer Functions for Internationalization**

```
TextUpdateFunction update_func,
XtPointer data);
```

**Arguments**

|                    |                                      |
|--------------------|--------------------------------------|
| <i>text</i>        | The text buffer                      |
| <i>startloc</i>    | The start location                   |
| <i>endloc</i>      | the end location                     |
| <i>string</i>      | The string to replace the block with |
| <i>update_func</i> | The update function                  |
| <i>data</i>        | Data                                 |

`OlReplaceBlockInTextBuffer()` updates the contents of the `TextBuffer` *text*. The characters stored between *startloc* (inclusive) and *endloc* (exclusive) are deleted and the string is inserted after *startloc*. If the edit succeeds and if `TextUpdateFunction` *update\_func* is associated with `TextBuffer` *text*, then *update\_func* is called with the following parameters:

```
(*update_func)(XtPointer d, TextBuffer *text, EDIT_SUCCESS)
```

All the other text update functions associated with `TextBuffer` *text* are called with the following parameters:

```
(XtPointer data, TextBuffer *text, EDIT_FAILURE)
```

`OlReplaceBlockInTextBuffer()` stores the details of the editing operation it performs in *text->deleted* and *text->insert* `OlTextUndoItem` structures. The contents of these structures may be used for implementing an Undo mechanism. The hints provided in *text->deleted.hint* and *text->insert.hint* are an inclusive OR of:

```
#define TEXT_BUFFER_NOP (0)
#define TEXT_BUFFER_DELETE_START_LINE (1L<<0)
#define TEXT_BUFFER_DELETE_START_CHARS (1L<<1)
#define TEXT_BUFFER_DELETE_END_LINE (1L<<2)
#define TEXT_BUFFER_DELETE_END_CHARS (1L<<3)
#define TEXT_BUFFER_DELETE_JOIN_LINE (1L<<4)
#define TEXT_BUFFER_DELETE_SIMPLE (1L<<5)
#define TEXT_BUFFER_INSERT_SPLIT_LINE (1L<<6)
#define TEXT_BUFFER_INSERT_LINE (1L<<7)
#define TEXT_BUFFER_INSERT_CHARS (1L<<8)
```



---

### *Text Buffer Functions for Internationalization*

The meaning of each of these values is described below:

---

|                                |                                                                                            |
|--------------------------------|--------------------------------------------------------------------------------------------|
| TEXT_BUFFER_NOP                | No edit operation.                                                                         |
| TEXT_BUFFER_DELETE_START_LINE  | The deleted block started at beginning of some line.                                       |
| TEXT_BUFFER_DELETE_START_CHARS | The deleted block did not start at the beginning of some line.                             |
| TEXT_BUFFER_DELETE_END_LINE    | The end of the deleted block coincided with the end of some line.                          |
| TEXT_BUFFER_DELETE_END_CHARS   | Some characters were deleted from the end of some line.                                    |
| TEXT_BUFFER_DELETE_JOIN_LINE   | Some characters were deleted and two lines were joined into a single line.                 |
| TEXT_BUFFER_DELETE_SIMPLE      | The whole of the deleted block was confined to a single line.                              |
| TEXT_BUFFER_INSERT_SPLIT_LINE  | One line was split into two lines and some characters were inserted at the split location. |
| TEXT_BUFFER_INSERT_LINE        | A line was inserted without splitting an existing line.                                    |
| TEXT_BUFFER_INSERT_CHARS       | Some characters were inserted at the beginning of some existing line.                      |

---

#### **See Also**

“OlReplaceCharInTextBuffer” on page 199.

#### ***OlReplaceCharInTextBuffer***

```
#include <Xol/Oltextbuff.h>
```

```
EditResult OlReplaceCharInTextBuffer(
 OlTextBufferPtr text,
 TextLocation *location,
 OlStr c,
 TextUpdateFunction update_func,
 XtPointer data);
```

## Text Buffer Functions for Internationalization

### Arguments

|                    |                      |
|--------------------|----------------------|
| <i>text</i>        | The text buffer      |
| <i>location</i>    | The location of text |
| <i>c</i>           | replacement buffer   |
| <i>update_func</i> | The update function  |
| <i>data</i>        | Data                 |

`OlReplaceCharInTextBuffer()` replaces the character in the `OlTextBuffer`.

### See Also

“`OlReplaceBlockInTextBuffer`” on page 197.

### *OlSaveTextBuffer*

```
#include <Xol/Oltextbuff.h>
```

```
SaveResult OlSaveTextBuffer(
 OlTextBufferPtr text,
 char *filename);
```

### Arguments

|                 |                                          |
|-----------------|------------------------------------------|
| <i>text</i>     | The text buffer                          |
| <i>filename</i> | The filename to write the text buffer to |

`OlSaveTextBuffer()` writes the contents of the `OlTextBuffer` to the file *filename*. It returns a `SaveResult`, which can be `SAVE_FAILURE` or `SAVE_SUCCESS`.

### *OlSetTextUndoDeleteItem*

```
#include <Xol/Oltextbuff.h>
```

```
void OlSetTextUndoDeleteItem(
 OlTextBufferPtr text,
 OlTextUndoItem text_undo_deleted);
```

### Arguments

|                          |                                          |
|--------------------------|------------------------------------------|
| <i>text</i>              | The text buffer                          |
| <i>text_undo_deleted</i> | The item for which the delete was undone |

## *Text Buffer Functions for Internationalization*

`OlSetTextUndoDeleteItem()` sets the “deleted” `OlTextUndoItem` of the `OlTextBuffer` to the value of the passed `OlTextUndoItem`. The “deleted” string is copied in. `OlTextUndoItem` is defined as:

```
typedef struct _OlTextUndoItem {
 OlStr string;
 TextLocation start;
 TextLocation end;
 TextUndoHint hint;
} OlTextUndoItem;
```

### *OlSetTextUndoInsertItem*

```
#include <Xol/Oltextbuff.h>

void OlSetTextUndoInsertItem(
 OlTextBufferPtr text,
 OlTextUndoItem text_undo_insert);
```

#### **Arguments**

|                         |                                          |
|-------------------------|------------------------------------------|
| <i>text</i>             | The text buffer                          |
| <i>text_undo_insert</i> | The item for which the insert was undone |

`OlSetTextUndoInsertItem()` sets the “insert” `OlTextUndoItem` of the `OlTextBuffer` to the value of the passed `OlTextUndoItem`. The “insert” string is copied in.

### *OlStartCurrentTextBufferWord*

```
#include <Xol/Oltextbuff.h>

TextLocation *OlStartCurrentTextBufferWord(
 OlTextBufferPtr text,
 TextLocation *current);
```

#### **Arguments**

|                |                 |
|----------------|-----------------|
| <i>text</i>    | The text buffer |
| <i>current</i> | The location    |

`OlStartCurrentTextBufferWord()` locates the beginning of a word in the `OlTextBuffer` relative to a given current location. It returns the location of the beginning of the current word.

*Text Buffer Functions for Internationalization*

---

**Note** – This return value will equal the given current value if the current location is the beginning of a word. If the location is not in a word, it returns the start of the “not word” region it is in. The location passed to this function is modified. It contains the start of the current buffer word (or “not word”) at the end of the call.

---

**See Also**

“OlPreviousTextBufferWord” on page 192,  
 “OlNextTextBufferWord” on page 189.

***OlTextEditOlTextBuffer***

```
#include <Xol/buffutil.h>
#include <Xol/Oltextbuff.h>
#include <Xol/Dynamic.h>
#include <Xol/TextEdit.h>

OlTextBufferPtr OlTextEditOlTextBuffer(
 TextEditWidget ctx);
```

`OlTextEditOlTextBuffer()` retrieves the `OlTextBufferPtr` associated with the `TextEdit` widget `ctx`. This buffer exists only when the value of `XtNtextFormat` for the widget is not `OL_SB_STR_REP`. This pointer can be used to access the facilities provided by the multibyte functions. In case `XtNtextFormat` is `OL_SB_STR_REP`, this function returns a `NULL` pointer.

***OlUnitOffsetOfLocation***

```
#include <Xol/Oltextbuff.h>

UnitPosition OlUnitOffsetOfLocation(
 OlTextBufferPtr text,
 TextLocation *loc);
```

**Arguments**

|             |                 |
|-------------|-----------------|
| <i>text</i> | The text buffer |
| <i>loc</i>  | The location    |

`OlUnitOffsetOfLocation()` returns the font offset corresponding to the `TextLocation` offset passed to it. The units are `char` for single-byte and multi-byte and `wchar_t` for wide character.

***OlUnregisterTextBufferUpdate***

```
#include <Xol/Oltextbuff.h>

int OlUnregisterTextBufferUpdate(
 OlTextBufferPtr text,
 TextUpdateFunction update_func,
 XtPointer data);
```

***Arguments***

|                    |                                     |
|--------------------|-------------------------------------|
| <i>text</i>        | The text buffer                     |
| <i>update_func</i> | The update function to disassociate |
| <i>data</i>        | Data                                |

`OlUnregisterTextBufferUpdate()` disassociates the `TextUpdateFunction` and data pointer `data` with the given `OlTextBuffer` `text`. If the function/data pointer pair is not associated with the given `OlTextBuffer`, zero is returned; otherwise, the association is dissolved and 1 is returned. See “`OlReplaceBlockInTextBuffer`” on page 197 for more details of the `TextUpdateFunction`.

***See Also***

“`OlRegisterTextBufferUpdate`” on page 197,  
“`OlFreeTextBuffer`” on page 180.

## *Text Selection Operations*

### *Text Selection Operations*

The Caption, NumericField, StaticText, TextEdit, TextField, and TextLine widgets use the following operations to copy and move text.

#### *Setting Insert Point*

Clicking SELECT sets the insert point at the boundary between two characters or spaces nearest the pointer. This makes an inactive caret active and highlights the header of the main window (base window or popup window) containing the specific text widget, to show which window has the input focus. Any active selection on the screen is deselected.

#### *Wipethrough Selection*

Pressing and dragging SELECT marks the bounds of a new selection and highlights it, and deselects any other active selection on the screen. While SELECT is pressed, the active or inactive caret that marks the insert point is invisible, but when SELECT is released, the insert point is left at the position of the release. This does not make the insert point (caret) active if it is not already active.

The selection starts with the character where SELECT is pressed and extends to the character where SELECT is released. If the pointer moves outside the widget and the widget can scroll in that direction (i.e., there is a scrollbar for that direction), the widget scrolls additional text into the widget and adds it to the selection. The rate at which text scrolls into the widget is the same rate at which pressing SELECT on the arrows of the Scrollbar scrolls the widget.

#### *Deletion of the New Selection*

If new text is entered from the keyboard or pasted from the CLIPBOARD, it replaces the selection.

### *Adjusted Selection*

Clicking SELECT, moving the pointer, and clicking ADJUST marks the bounds of a selection and highlights it. A subsequent click of ADJUST changes the end bound of the selection. The ADJUST may also follow a wipe-through selection. The selection starts with the character where SELECT was clicked and extends to the character where ADJUST is clicked. The insert point is moved to the position of the ADJUST. As above, deletion of the new selection is pending.

### *Multiclick Selection*

Double-clicking SELECT selects the word nearest the pointer. In case of a tie, the word to the left is selected. Triple-clicking SELECT selects the entire line, and quadruple-clicking selects the entire content. The selection is highlighted and the insert point is left at the position of the multi-click.

- Copying Text – Using COPY copies any selected text to the CLIPBOARD and deselects it.
- Cutting Text – Using CUT moves any selected text to the CLIPBOARD and deletes it from the Input Field.
- Pasting Text – After setting the insert point, using PASTE copies text from the CLIPBOARD as though it were typed in, leaving the insert point at the end of the pasted text. This will replace any text currently selected in the widget. Note that the data on the CLIPBOARD may have come from outside the input field, but it must be text. If the CLIPBOARD is empty, the system beeps.

## Toolkit Resource Functions

### *OlGetApplicationValues*

```
#include <Xol/OpenLook.h>
void OlGetApplicationValues(
 Widget widget,
 ArgList args,
 Cardinal num_args);
```

`OlGetApplicationValues()` retrieves the value of any of the OLIT Toolkit Resources listed in Table 2-1 on page 7. OLIT toolkit resources have an application-wide scope. The *widget* argument is used to derive the screen and display. The *args* argument is a list of name/address pairs that contain the resource names and the addresses into which the resource values are to be stored. The *num\_args* argument specifies the number of name/address pairs in *args*. If the resource name supplied in the *args* list is not recognized by the toolkit, the corresponding supplied address is not accessed by the toolkit. An application should query the value of an OLIT toolkit resource each time it needs it.

### *OlSetApplicationValues*

```
#include <Xol/OpenLook.h>
void OlSetApplicationValues(
 Widget widget,
 ArgList args,
 Cardinal num_args);
```

`OlSetApplicationValues()` sets the OLIT toolkit resources values. The *widget* and *num\_args* arguments are used as in `OlGetApplicationValues()`. The *args* argument is a list of name/value pairs that contain the resource names and the values; as with `XtSetValues()`, if a resource name does not fit into an `XtArgVal`, the corresponding *args* value field contains a pointer to the resource value.

### *See Also*

`XtGetValues()` and `XtSetValues()` in the *Xt Intrinsic Reference Manual*.



## Virtual Event Functions

For all functions discussed here, the registration order determines the search order when doing a lookup.

### *LookupOlInputEvent*

```
#include <Xol/Dynamic.h>

OlInputEvent LookupOlInputEvent(
 Widget w,
 XEvent *event,
 KeySym *keysym,
 char **buffer,
 int *length);
```

`LookupOlInputEvent()` decodes the *event* for widget *w* to an `OlInputEvent`. See the table of OLIT Activation Types (Table 3-1 on page 65) for a list of the `OlInputEvent` values (listed in the “Activation Type” column) this function may return. The event passed should be a `ButtonPress`, `ButtonRelease`, or `KeyPress` event. `LookupOlInputEvent()` attempts to decode this event based on the settings of the OPEN LOOK defined dynamic mouse and keyboard settings.

If the event is a `KeyPress`, the function may return the *keysym*, *buffer*, and/or *length* of the buffer returned from a call to `XLookupString()`. It returns these values if non-NULL values are provided by the caller.

### *OldetermineMouseAction*

```
#include <Xol/Dynamic.h>

ButtonAction OldetermineMouseAction(
 Widget w,
 XEvent *event);
```

`OldetermineMouseAction()` determines the kind of mouse gesture that is being attempted: it will return one of the values `MOUSE_CLICK`, `MOUSE_MULTI_CLICK`, or `MOUSE_MOVE`. This function is normally called immediately upon receipt of a mouse button press event. It uses the current settings for the `XtNmouseDampingFactor` and `XtNmultiClickTimeout` resources to determine the kind of gesture being made.

*Virtual Event Functions*

OldetermineMouseAction() performs an active pointer grab. This grab is released for the CLICK type actions, but not for MOUSE\_MOVE. It is the responsibility of the caller to ungrab the pointer if the action is MOUSE\_MOVE.

**Example**

```
static void ButtonConsumeCB (w, client_data, call_data)
widget w;
XtPointer client_data;
XtPointer call_data;
{
Position x, y;
OlVirtualEvent ve;
ve = (OlVirtualEvent) call_data;
switch (ve->virtual_name) {
case OL_SELECT:
switch(OldetermineMouseAction(widget, event)) {
case MOUSE_MOVE:
OlGrabDragPointer(widget,
OlGetMoveCursor(XtScreen(widget), None);
OlDragAndDrop(widget, &drop_window, &x, &y);
DropOn(widget, drop_window, x, y,);
OlUngrabDragPointer(widget);
break;
case MOUSE_CLICK:
ClickSelect(widget,);
break;
case MOUSE_MULTI_CLICK:
MultiClickSelect(widget,);
break;
}
break;
default:
OlReplayBtnEvent(widget, NULL, event);
break;
}
}
```

### *OlReplayBtnEvent*

```
#include <Xol/Dynamic.h>
void OlReplayBtnEvent(
 Widget w,
 caddr_t client_data,
 XEvent *event);
```

`OlReplayBtnEvent()` replays a button press event to the next window (towards the root) that is interested in button events. This provides a means of propagating events up a window tree.

### *OlClassSearchIEDB*

```
#include <Xol/OpenLook.h>
void OlClassSearchIEDB(
 WidgetClass wc;
 OlVirtualEventTable db);
```

`OlClassSearchIEDB()` registers a given database on a specific widget class. The *db* value was returned from a call to `OlCreateInputEventDB()`. Once a database is registered with a given widget class, the `OlLookupInputEvent()` procedure (if *db\_flag* is `OL_DEFAULT_IE` or `OLTEXT_IE`) will include this database in the search stack if the given widget ID is a subclass of this widget class.

### *Example*

```
/* To create a client application database */
 /* start with a big value to avoid */
 /* the "virtual_name" collision */
#define OL_MY_BASE 1000
#define OL_MY_REDISPLAYKEY OL_MY_BASE+2
#define OL_MY_SAVEPARTKEY OL_MY_BASE+3

#define XtNmyDrawLineBtn "myDrawLineBtn"
#define XtNmyDrawArcBtn "myDrawArcBtn"
#define XtNmyRedisplayKey "myRedisplayKey"
#define XtNmySavePartKey "mySavePartKey"

static OlKeyOrBtnRec OlMyBtnInfo[] = {
 /*name default_value virtual_name */
};
```

## Virtual Event Functions

```

static OlKeyOrBtnRec OlMyKeyInfo[] = {
 /*name default_value virtual_name */
 { XtNmyRedisplayKey, "c<F5>", OL_MY_REDISPLAYKEY },
 { XtNmySavePartKey, "c<F5>", OL_MY_SAVEPARTKEY },
};

static OlVirtualEventTable OlMyDB;

...
OlMyDB = OlCreateInputEventDB(
 w,
 OlMyKeyInfo, XtNumber(OlMyKeyInfo),
 OlMyBtnInfo, XtNumber(OlMyBtnInfo)
);

...
/* assume: all stub widgets are interested in OlMyDB */
OlClassSearchIEDB(stubWidgetClass, OlMyDB);
/* once this step is done, all stub widget instances */
/* will receive the OlMyDB commands after a call to */
/* OlLookupInputEvent(), or in the XtNconsumeEvent */
/* callback's OlVirtualEvent structure supplied with */
/* the call_data field. */

```

### *OlClassSearchTextDB*

```

#include <Xol/OpenLook.h>
void OlClassSearchTextDB(
 WidgetClass wc);

```

`OlClassSearchTextDB()` registers the OPEN LOOK TEXT database on a specific widget class. Once the OPEN LOOK TEXT database is registered with a given widget class, the `OlLookupInputEvent()` procedure (if *db\_flag* is `OL_DEFAULT_IE` or `OLTEXT_IE`) will include this database in the search stack if the given widget ID is a subclass of this widget class.

### **Example**

```

/* assume: all stub widgets are interested in the */
/* OPEN LOOK TEXT database */
OlClassSearchTextDB(stubWidgetClass);
/* once this step is done, all stub widget instances */
/* will receive OPEN LOOK TEXT commands after a */

```

```

/* call to OlLookupInputEvent(), or in the */
/* XtNconsumeEvent callback's OlVirtualEvent */
/* structure supplied with the call_data field. */

```

### *OlCreateInputEventDB*

```

#include <Xol/OpenLook.h>

OlVirtualEventTable OlCreateInputEventDB(
 Widget w,
 OlKeyOrBtnInfo key_info,
 int num_key_info,
 OlKeyBtnInfo btn_info,
 int num_btn_info);

```

`OlCreateInputEventDB()` creates a client specific Key and/or Button database. This function returns a database pointer if the call to this function is successful; otherwise, a NULL pointer is returned. Mapping for a new virtual command can be composed from the mappings of a previously defined virtual command. The returned value from this function is an opaque pointer (`OlVirtualEventTable`). A client application should use this pointer when registering and/or looking up this database.

```
typedef struct _OlVirtualEventInfo *OlVirtualEventTable;
```

The *key\_info* and *btn\_info* parameters are pointers to an `OlKeyOrBtnRec` structure.

```

typedef struct {
 String name;
 String default_value; /* comma-separated string */
 OlVirtualName virtual_name;
} OlKeyOrBtnRec, *OlKeyOrBtnInfo;

```

**Note** – A client application can create a Key-only database by specifying a NULL *btn\_info*. The same applies to a Button-only database. Each virtual command can have two different bindings because the OLIT toolkit allows the alternate key or button sequence. The OLIT toolkit already has a set of predefined OPEN LOOK virtual names. It is important that the *virtual\_name* value of a client application database starts with a big value to avoid a virtual name collision.

**Example**

```

 /* To create a client application database */
 /* start with a big value to avoid */
 /* the "virtual_name" collision */
#define OL_MY_BASE 1000
#define OL_MY_REDISPLAYKEY OL_MY_BASE+2
#define OL_MY_SAVEPARTKEY OL_MY_BASE+3

#define XtNmyDrawLineBtn "myDrawLineBtn"
#define XtNmyDrawArcBtn "myDrawArcBtn"
#define XtNmyRedisplayKey "myRedisplayKey"
#define XtNmySavePartKey "mySavePartKey"

static OlKeyOrBtnRec OlMyBtnInfo[] = {
 /*name default_value virtual_name */
};

static OlKeyOrBtnRec OlMyKeyInfo[] = {
 /*name default_value virtual_name */
 { XtNmyRedisplayKey, "c<F5>", OL_MY_REDISPLAYKEY },
 { XtNmySavePartKey, "c<F5>", OL_MY_SAVEPARTKEY },
};

static OlVirtualEventTable OlMyDB;

...
OlMyDB = OlCreateInputEventDB(
 w,
 OlMyKeyInfo, XtNumber(OlMyKeyInfo),
 OlMyBtnInfo, XtNumber(OlMyBtnInfo)
);
...

```

***OlLookupInputEvent***

```

#include <Xol/OpenLook.h>

void OlLookupInputEvent(
 Widget w,
 XEvent *xevent,
 OlVirtualEvent virtual_event_ret,
 XtPointer db_flag);

```

`OlLookupInputEvent()` translates an X event to an OPEN LOOK virtual event. The X event (*xevent*) could be a `KeyPress`, `ButtonPress`, `ButtonRelease`, `EnterNotify`, `LeaveNotify`, or `MotionNotify` event. The procedure attempts to translate this event based on the setting of the OPEN LOOK-defined dynamic

databases. The *virtual\_event\_ret* parameter is a pointer to an `OlVirtualEventRec` structure in which the OPEN LOOK virtual event is returned:

```
typedef struct {
 Boolean consumed;
 XEvent *xevent;
 Modifiers dont_care;
 OlVirtualName virtual_name;
 KeySym keysym;
 String buffer;
 Cardinal length;
 Cardinal item_index;
} OlVirtualEventRec, *OlVirtualEvent;
```

(This structure is also used by the `XtNconsumeEvent` resource's callbacks.)

See the table of OLIT Activation Types (Table 3-1 on page 65) for a list of the values (listed in the "Activation Type" column) that may be returned in the *virtual\_name* member of the *virtual\_event\_rec*. If the X event is a KeyPress, the *keysym*, *buffer*, and *length* information will be included in *virtual\_event\_ret*; `OlLookupInputEvent()` obtains these values from `XLookupString()`.

The (*w*, *db\_flag*) pair determines the searching database(s). Valid values for the *db\_flag* parameter are `OL_DEFAULT_IE`, `OL_CORE_IE`, and `OL_TEXT_IE`. If the *db\_flag* value is not `OL_DEFAULT_IE`, then only the given database (for example, `OL_TEXT_IE` means: search the OPEN LOOK TEXT database) will be searched; otherwise, a search stack will be built. This stack is based on the widget information (*w*) and the registering order to determine the searching database(s). Once this stack is built, the procedure searches in a LIFO (Last In First Out) manner.

Most OLIT widgets have an `XtNconsumeEvent` callback. When this callback is called, the *call\_data* field is a pointer to an `OlVirtualEventRec` structure that is filled in with the results of calling `OlLookupInputEvent()` with the *db\_flag* set to `OL_DEFAULT_IE`.

### Example

```
OlVirtualEventRec ve;

 /* To look up the OPEN LOOK CORE database */
OlLookupInputEvent(w, xevent, &ve, OL_CORE_IE);
switch (ve.virtual_name) {
 case OL_UNKNOWN_INPUT:
```

*Virtual Event Functions*

---

```

 ...
 case OL_UNKNOWN_KEY_INPUT:
 ...
 case OL_ADJUST:
 printf ("pressed the adjustBtn\n");
 ...
 case OL_ADJUSTKEY:
 printf ("pressed the adjustKey\n");
 ...
}

...
OlVirtualEventRec ve;
 /* To look up the OPEN LOOK TEXT database */
OlLookupInputEvent(w, xevent, &ve, OLTEXT_IE);
switch (ve.virtual_name) {
 ...
 case OL_DOCEND:
 printf ("pressed the docEndKey\n");
 ...
 case OL_LINEEND:
 printf ("pressed the lineEndKey\n");
 ...
}

...
OlVirtualEventRec ve;
 /* To look up all possible databases */
 /* assume: "w" is a textfield widget */
OlLookupInputEvent(w, xevent, &ve, OL_DEFAULT_IE);
switch (ve.virtual_name) {
 ...
 case OL_ADJUST:
 printf ("pressed the adjustBtn\n");
 ...
 case OL_ADJUSTKEY:
 printf ("pressed the adjustKey\n");
 ...
 case OL_DOCEND:
 printf ("pressed the docEndKey\n");
 ...
 case OL_LINEEND:
 printf ("pressed the lineEndKey\n");
 ...
}

```



**OlWidgetSearchIEDB**

```
#include <Xol/OpenLook.h>
void OlWidgetSearchIEDB(
 Widget w,
 OlVirtualEventTable db);
```

`OlWidgetSearchIEDB()` registers a given database on a specific widget instance. The `db` value was returned from a call to `OlCreateInputEventDB()`. Once a database is registered with a given widget instance, the `OlLookupInputEvent()` procedure (if `db_flag` is `OL_DEFAULT_IE` or `OL_TEXT_IE`) will include this database in the search stack if the given widget ID is this widget instance.

**Example**

```
/* To create a client application database */
/* start with a big value to avoid */
/* the "virtual_name" collision */
#define OL_MY_BASE 1000
#define OL_MY_REDISPLAYKEY OL_MY_BASE+2
#define OL_MY_SAVEPARTKEY OL_MY_BASE+3

#define XtNmyDrawLineBtn "myDrawLineBtn"
#define XtNmyDrawArcBtn "myDrawArcBtn"
#define XtNmyRedisplayKey "myRedisplayKey"
#define XtNmySavePartKey "mySavePartKey"

static OlKeyOrBtnRec OlMyBtnInfo[] = {
 /*name default_value virtual_name */
};

static OlKeyOrBtnRec OlMyKeyInfo[] = {
 /*name default_value virtual_name */
 { XtNmyRedisplayKey, "c<F5>", OL_MY_REDISPLAYKEY },
 { XtNmySavePartKey, "c<F5>", OL_MY_SAVEPARTKEY },
};

static OlVirtualEventTable OlMyDB;
...
OlMyDB = OlCreateInputEventDB(
 w,
 OlMyKeyInfo, XtNumber(OlMyKeyInfo),
 OlMyBtnInfo, XtNumber(OlMyBtnInfo)
);
...

```

**Virtual Event Functions**

```

 /* Assume: "w" is a stub widget that is interested in */
 /* OlMyDB */
OlWidgetSearchIEDB(w, OlMyDB);
 /* Once this step is done, this widget instance will */
 /* receive OlMyDB commands after a call to */
 /* OlLookupInputEvent(), or in the XtNconsumeEvent */
 /* callback's OlVirtualEvent structure supplied with */
 /* the call_data field. */

```

***OlWidgetSearchTextDB***

```

#include <Xol/OpenLook.h>
void OlWidgetSearchTextDB(
 OlVirtualEventTable w);

```

`OlWidgetSearchTextDB()` is used to register the OPEN LOOK TEXT database on a given widget instance.

Once the OPEN LOOK TEXT database is registered with a given widget instance, the `OlLookupInputEvent()` procedure (if *db\_flag* is `OL_DEFAULT_IE` or `OL_TEXT_IE`) will include this database in the search stack if the given widget ID is this widget instance.

***Example***

```

 /* assume: "w" is a stub widget that is interested in */
 /* the OPEN LOOK TEXT database */
OlWidgetSearchTextDB(w);
 /* Once this step is done, this widget instance will */
 /* receive OPEN LOOK TEXT commands after a call */
 /* to OlLookupInputEvent(), or in the XtNconsumeEvent */
 /* callbacks OlVirtualEvent structure supplied with */
 /* the call_data field. */

```

***See Also***

Chapter 3, "Activation Types."

## *Widget Reference (A – C)*

---



Chapters 6 to 10 describe the widgets in the OLIT widget set and functions that augment the specific widgets. (For functions that are not specific to a widget, see Chapter 5, “Toolkit Functions.”)

### *AbbrevMenuButton Widget*

#### *Class*

*Class Name:*     AbbrevMenuButton  
*Class Pointer:*  abbrevMenuButtonWidgetClass

#### *Ancestry*

Core-Primitive-AbbrevMenuButton

#### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/AbbrevMenu.h>
```

#### *Description*

The AbbrevMenuButton widget is used to create a popup menu that also provides current selection viewing to the user. When the user invokes the

## AbbrevMenuButton Widget

MENU command on the AbbrevMenuButton, a menu pops up. Once the user makes a selection off the menu, the selected item should be displayed next to the AbbrevMenuButton. The AbbrevMenuButton also provides the features of the MenuButton widget (menu default selection, menu previewing, menu selection).

### Components

The AbbrevMenuButton consists of a square button containing an arrow (menumark) with a popup menu attached. An application should create and identify an additional component, the Current Selection Widget, which is used to display previewing and also to display the current selection off the menu. Each AbbrevMenuButton also has the components of the MenuShell widget.

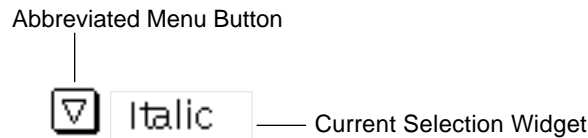


Figure 6-1 AbbrevMenuButton Widget

### Subwidget

The AbbrevMenuButton widget automatically creates and attaches a MenuShell widget. An application can add menu items to this menu by obtaining the value of the XtNmenuPane resource and adding children to this widget.

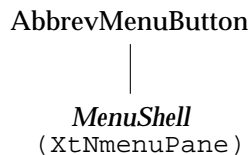


Figure 6-2 AbbrevMenuButton Subwidget

### Current Selection Widget

The Current Selection Widget is created by the application. This widget should be a StaticText, TextField, or TextLine widget. Typically, the Current Selection Widget and the AbbrevMenuButton widget are placed together in a composite

## *AbbrevMenuButton Widget*

widget that manages their side-by-side placement. The `AbbrevMenuButton` widget uses the `Current Selection Widget` only for previewing the default item in the menu. The application is responsible for using it to display the current selection, and if the `Current Selection Widget` is an editable field, for adding the new item to the menu as appropriate.

### *Coloration*

For 3D, `AbbrevMenuButton` coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.” `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNbackground` and `XtNforeground` are used to render the `AbbrevMenuButton` as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, “Controls.”

If the toolkit resource `XtNmouseless` is set to `TRUE` and the toolkit resource `XtNinputFocusFeedback` is set to `OL_INPUT_FOCUS_COLOR`, then the background of the `AbbrevMenuButton` will be drawn with the value of `XtNinputFocusColor` when the widget receives input focus. However, if `XtNinputFocusColor` is the same as `XtNbackground`, then the widget inverts `XtNforeground` and `XtNbackground`. Once the input focus leaves the widget, the original coloration is restored.

### *Keyboard Traversal*

The default value of the `XtNtraversalOn` resource is `TRUE`. The `AbbrevMenuButton` responds to the following navigation keys:

- **SELECTKEY:** The response depends on the value of the toolkit resource `XtNselectDoesPreview`. If `XtNselectDoesPreview` is `TRUE`, this key activates the default item in the Menu. If `XtNselectDoesPreview` is `FALSE`, this key pops up the Menu and transfers the keyboard focus to the default item in the Menu.
- **MENUKEY:** Pops up the Menu and transfers the keyboard focus to the default item in the Menu.
- **MOVEDOWN:** Pops up the Menu and transfers the keyboard focus to the default item in the Menu.

*AbbrevMenuButton Widget*

- NEXTFIELD and MOVERIGHT move to the next traversable widget in the shell.
- PREVFIELD, MOVEUP, and MOVELEFT move to the previous traversable widget in the shell.
- NEXTWINDOW moves to the next window in the application.
- PREVWINDOW moves to the previous window in the application.
- NEXTAPP moves to the first window in the next application.
- PREVAPP moves to the first window in the previous application.
- The response of the Menu associated with the AbbrevMenuButton is specified in the MenuShell widget’s description. See page 414.

***Keyboard Mnemonic Display***

The AbbrevMenuButton does not display the mnemonic accelerator. If the AbbrevMenuButton is the child of a Caption widget, the Caption widget can be used to display the mnemonic.

***Keyboard Accelerator Display***

The AbbrevMenuButton does not display the keyboard accelerator. If the AbbrevMenuButton is a child of a Caption widget, the Caption widget can be used to display the accelerator as part of the label.

*Resources*

*Table 6-1* AbbrevMenuButton Core Resources

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent’s)          | SGI           |
| XtNdepth             | int              | (parent’s)          | GI            |

*AbbrevMenuButton Widget**Table 6-1* AbbrevMenuButton Core Resources (Continued)

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNdestroyCallback   | XtCallbackList | NULL           | SGIO          |
| XtNheight            | Dimension      | (calculated)   | SGI           |
| XtNmappedWhenManaged | Boolean        | TRUE           | SGI           |
| XtNscreen            | Screen *       | (parent's)     | G             |
| XtNsensitive         | Boolean        | TRUE           | GIO           |
| XtNtranslations      | XtTranslations | NULL           | SGI           |
| XtNwidth             | Dimension      | (calculated)   | SGI           |
| XtNx                 | Position       | 0              | SGI           |
| XtNy                 | Position       | 0              | SGI           |

*Table 6-2* AbbrevMenuButton Primitive Resources

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|--------------------|----------------|---------------------|---------------|
| XtNaccelerator     | String         | NULL                | SGI           |
| XtNacceleratorText | String         | NULL                | SGI           |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor | Pixel          | Red                 | SGID          |
| XtNmnemonic        | unsigned char  | '\0'                | SGI           |
| XtNreferenceName   | String         | NULL                | GI            |
| XtNreferenceWidget | Widget         | NULL                | GI            |
| XtNscale           | int            | 12                  | SGI           |
| XtNtextFormat      | OlStrRep       | OL_SB_STR_REP       | GI            |
| XtNtraversalOn     | Boolean        | TRUE                | SGI           |
| XtNuserData        | XtPointer      | NULL                | SGI           |

*Table 6-3* AbbrevMenuButton Resources

| <b>Name</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtNmenuPane      | Widget      | (special)      | G             |
| XtNpreviewWidget | Widget      | NULL           | SGI           |

*AbbrevMenuButton Widget*

The following table lists the *AbbrevMenuButton* resources that are propagated to the *MenuShell* subwidget.

*Table 6-4 AbbrevMenuButton Subwidget Resources<sup>1</sup>*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b>  |
|-------------------|-------------|----------------|----------------|
| XtNcenter         | Boolean     | TRUE           | I              |
| XtNhPad           | Dimension   | 4              | I              |
| XtNhSpace         | Dimension   | 4              | I              |
| XtNlayoutType     | OlDefine    | OL_FIXEDROWS   | I <sup>2</sup> |
| XtNmeasure        | int         | 1              | I <sup>2</sup> |
| XtNpushpin        | OlDefine    | OL_NONE        | I              |
| XtNpushpinDefault | Boolean     | FALSE          | I              |
| XtNsameSize       | OlDefine    | OL_COLUMNS     | I              |
| XtNshellTitle     | OlStr       | (widget name)  | SGI            |
| XtNvPad           | Dimension   | 4              | I              |
| XtNvSpace         | Dimension   | 4              | I              |

1. These subwidget resources are described in the sections “ControlArea Widget” on page 249 and “MenuShell Widget” on page 414.

2. These resources can only be set programmatically via *XtCreateWidget*, *XtVaCreateWidget*, etc. Application resource file settings will not apply to these resources.

***XtNmenuPane***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCMenuPane  | Widget      | (special)      | G             |

Synopsis: The widget where menu items can be added.

Values: ID of the menupane widget contained in the *AbbrevMenuButton*’s *MenuShell*.

The value of this resource is available once the *AbbrevMenuButton* widget has been created.

***XtNpreviewWidget***

| <b>Class</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtCPreviewWidget | Widget      | NULL           | SGI           |

Synopsis: The Current Selection Widget that the *AbbrevMenuButton* can use for previewing the Default Item.



*AbbrevMenuButton Widget*

Values: ID of an existing widget; this should be a StaticText, TextField, or TextLine widget.

When the user presses SELECT over the AbbrevMenuButton widget, it uses the location and size of the Current Selection Widget to display the label of the Default Item. The preview is constrained to be within the height and width of the Current Selection Widget. If the Current Selection Widget is not defined or is not mapped, previewing does not take place.

*Activation Types*

The following table lists the activation types used by the AbbrevMenuButton.

Table 6-5 AbbrevMenuButton Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MENU          | MENU          | XtNmenuBtn          |
| OL_MENUKEY       | MENU          | XtNmenuKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_SELECT        | SELECT        | XtNselectBtn        |
| OL_SELECTKEY     | SELECT        | XtNselectKey        |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

***OL\_MENU***

The OL\_MENU activation type can be used to pop up the menu in two different modes: press-drag-release and click-move-click. These modes are described in the *OPEN LOOK GUI Functional Specification* section “Using Menu Buttons” in Chapter 15. The position of the menu depends on the space available on the

### *AbbrevMenuButton Widget*

screen and is described in the *OPEN LOOK GUI Functional Specification* section “Menu Placement” in Chapter 15.

#### ***OL\_MENUKEY***

The *OL\_MENUKEY* activation type can be used to pop up the menu according to the *OPEN LOOK Mouseless Specification* section 4.2.

#### ***OL\_SELECT***

The activation of the *AbbrevMenuButton* widget with the *SELECT* button depends on the value of the toolkit resources *XtNselectDoesPreview*. When the resource *XtNselectDoesPreview* is *FALSE*, this activation type will behave exactly as the *OL\_MENU* activation type described previously. When *XtNselectDoesPreview* is *TRUE*, the *SELECT* action can be used as a shortcut to display and activate the menu default as described in the *OPEN LOOK GUI Functional Specification* section “Button Controls” in Chapter 4.

#### ***OL\_SELECTKEY***

When the *AbbrevMenuButton* has keyboard focus, the *OL\_SELECTKEY* activation type can be used to pop up the menu according to the *OPEN LOOK Mouseless Specification* section 4.2.

### *See Also*

“*ControlArea Widget*” on page 249,  
“*MenuButton Widget*” on page 403,  
“*MenuShell Widget*” on page 414.

## *BulletinBoard Widget*

### *Class*

*Class Name:* BulletinBoard  
*Class Pointer:* bulletinBoardWidgetClass

### *Ancestry*

Core-Composite-Constraint-Manager-BulletinBoard

### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/BulletinBo.h>
```

### *Description*

The BulletinBoard widget is a composite widget that enforces no ordering on its children. It is up to the application to specify the x- and y-coordinates of each child inserted. Otherwise, it will be placed in the upper left corner of the BulletinBoard widget. The BulletinBoard can be mapped with no children. It displays an empty space which can be surrounded by a border.

### *Keyboard Traversal*

The BulletinBoard widget is a Composite widget and cannot be accessed via keyboard traversal. Input focus moves between the Primitive children of this widget.

*BulletinBoard Widget*

**Coloration**

The following diagram illustrates the resources used for BulletinBoard coloration.

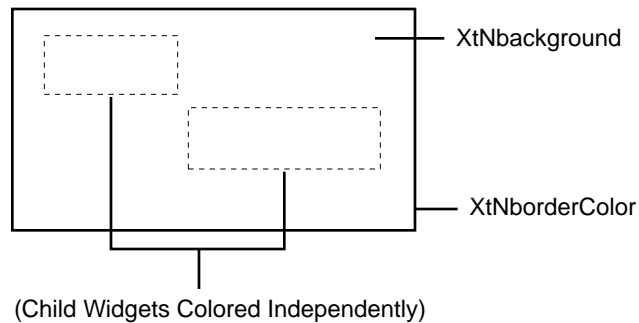


Figure 6-3 BulletinBoard Coloration

**Resources**

Table 6-6 BulletinBoard Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | 0                   | SGI    |

*BulletinBoard Widget**Table 6-6* BulletinBoard Core Resources (Continued)

| <b>Name</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------|-------------|----------------|---------------|
| XtNx        | Position    | 0              | SGI           |
| XtNy        | Position    | 0              | SGI           |

*Table 6-7* BulletinBoard Composite Resources

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 6-8* BulletinBoard Manager Resources

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor   | Pixel          | Red            | n/a           |
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 6-9* BulletinBoard Resources

| <b>Name</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------|-------------|----------------|---------------|
| XtNlayout   | OlDefine    | OL_MINIMIZE    | SGI           |

***XtNlayout***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCLayout    | OlDefine    | OL_MINIMIZE    | SGI           |

Synopsis: The layout policy the BulletinBoard widget is to follow.

**BulletinBoard Widget**

- Values:
- OL\_MINIMIZE/"minimize" - The BulletinBoard widget will always be just large enough to contain all its children, regardless of any provided width and height values. Thus, the BulletinBoard widget will grow and shrink depending on the size needs of its children.
  - OL\_IGNORE/"ignore" - The BulletinBoard widget will honor its own width and height; it will not grow or shrink in response to the addition, deletion, or alteration of its children.
  - OL\_MAXIMIZE/"maximize" - The BulletinBoard widget will grow as required due to new or altered children, but will not shrink.

**Activation Types**

The following table lists the activation types used by the BulletinBoard.

*Table 6-10* BulletinBoard Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

The BulletinBoard widget has no activation types besides the ones in "Common Activation Types" on page 68.

## Caption Widget

### Class

**Class Name:** Caption  
**Class Pointer:** captionWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-Caption

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/Caption.h>
```

### Description

The Caption composite widget provides a convenient way to label an arbitrary widget.

### Components

The Caption widget has two parts: the label and the child widget.

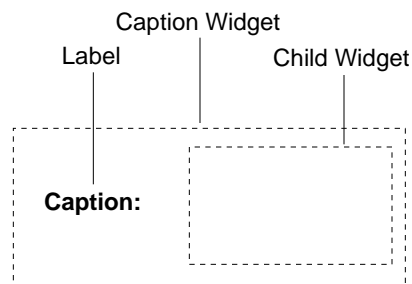


Figure 6-4 Caption Widget

### Layout Control

The application can specify that the label goes above, below, to the left, or to the right of the child, and how far away the label is to be placed.

### Child Constraints

The Caption composite allows only one child; attempts to add more than one are refused with a warning. If the Caption widget is mapped without a child widget, or if the child widget is not managed, only the label is shown.

### Coloration

The following diagram illustrates the resources used for Caption coloration.

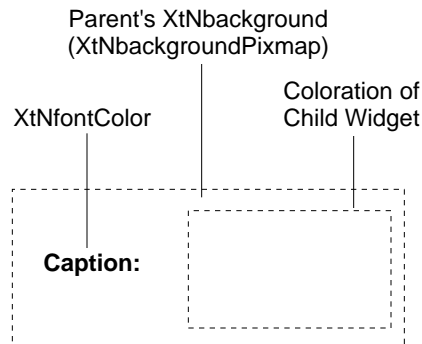


Figure 6-5 Caption Coloration

### Keyboard Traversal

The Caption is a special Manager widget that can be used to display the mnemonic for its single child. However, the label used as a caption to the child is not accessible via keyboard traversal.

The action of a mnemonic on a Caption widget is used for traversal as well as performing a SELECT on the Caption widget's child widget.



**Keyboard Mnemonic Display**

The Caption widget displays its mnemonic as part of its label. If the mnemonic character is in the label, then that character is marked according to the value of the toolkit resource `XtNshowMnemonics`. If the mnemonic character is not in the label, it is displayed to the right of the label in parentheses and marked according to the value of `XtNshowMnemonics`.

If truncation is necessary, the mnemonic displayed in parentheses is truncated as a unit.

**Keyboard Accelerator Display**

The Caption widget displays the keyboard accelerator for its child as part of its label. The string in the `XtNacceleratorText` resource is displayed to the right of the label (or mnemonic), right justified, and separated by at least one space.

If truncation is necessary, the accelerator is truncated as a unit. The accelerator is truncated before the mnemonic or the label.

**Resources***Table 6-11* Caption Core Resources

| Name                              | Type                        | Default                          | Access |
|-----------------------------------|-----------------------------|----------------------------------|--------|
| <code>XtNaccelerators</code>      | AcceleratorTable            | NULL                             | SGI    |
| <code>XtNancestorSensitive</code> | Boolean                     | TRUE                             | G      |
| <code>XtNbackground</code>        | Pixel                       | <code>XtDefaultBackground</code> | SGID   |
| <code>XtNbackgroundPixmap</code>  | Pixmap                      | <code>XtUnspecifiedPixmap</code> | SGI    |
| <code>XtNborderColor</code>       | Pixel                       | <code>XtDefaultForeground</code> | SGID   |
| <code>XtNborderPixmap</code>      | Pixmap                      | <code>XtUnspecifiedPixmap</code> | SGI    |
| <code>XtNborderWidth</code>       | Dimension                   | 1                                | SGI    |
| <code>XtNcolormap</code>          | Colormap                    | (parent's)                       | SGI    |
| <code>XtNdepth</code>             | int                         | (parent's)                       | GI     |
| <code>XtNdestroyCallback</code>   | <code>XtCallbackList</code> | NULL                             | SGIO   |
| <code>XtNheight</code>            | Dimension                   | 0                                | SGI    |
| <code>XtNmappedWhenManaged</code> | Boolean                     | TRUE                             | SGI    |
| <code>XtNscreen</code>            | Screen *                    | (parent's)                       | G      |
| <code>XtNsensitive</code>         | Boolean                     | TRUE                             | GIO    |

*Caption Widget*

*Table 6-11* Caption Core Resources (Continued)

| <b>Name</b>     | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|-----------------|----------------|----------------|---------------|
| XtNtranslations | XtTranslations | NULL           | SGI           |
| XtNwidth        | Dimension      | 0              | SGI           |
| XtNx            | Position       | 0              | SGI           |
| XtNy            | Position       | 0              | SGI           |

*Table 6-12* Caption Composite Resources

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 6-13* Caption Manager Resources

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor   | Pixel          | Red            | SGID          |
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 6-14* Caption Resources

| <b>Name</b>  | <b>Type</b>   | <b>Default</b>      | <b>Access</b> |
|--------------|---------------|---------------------|---------------|
| XtNalignment | OlDefine      | OL_CENTER           | SGI           |
| XtNfont      | OlFont        | OlDefaultBoldFont   | SGID          |
| XtNfontColor | Pixel         | XtDefaultForeground | SGID          |
| XtNlabel     | OlStr         | (instance name)     | SGI           |
| XtNmnemonic  | unsigned char | '\0'                | SGI           |
| XtNposition  | OlDefine      | OL_LEFT             | SGI           |

Table 6-14 Caption Resources (Continued)

| Name             | Type      | Default       | Access |
|------------------|-----------|---------------|--------|
| XtNrecomputeSize | Boolean   | TRUE          | SGI    |
| XtNspace         | Dimension | 4             | SGI    |
| XtNtextFormat    | OlStrRep  | OL_SB_STR_REP | GI     |

**XtNalignment**

| Class        | Type     | Default   | Access |
|--------------|----------|-----------|--------|
| XtCAlignment | OlDefine | OL_CENTER | SGI    |

Synopsis: The alignment of the label relative to the child widget.

Values: OL\_BOTTOM/"bottom" - Align the bottom edge of the label with the bottom edge of the child widget.  
 OL\_CENTER/"center" - Align the center of the label with the center of the child widget.  
 OL\_LEFT/"left" - Align the left edge of the label with the left edge of the child widget.  
 OL\_RIGHT/"right" - Align the right edge of the label with the right edge of the child widget.  
 OL\_TOP/"top" - Align the top edge of the label with the top edge of the child widget.

The XtNalignment and XtNposition resources interact in the following way. If XtNposition is OL\_LEFT or OL\_RIGHT, then the alignment can be OL\_TOP, OL\_CENTER, or OL\_BOTTOM. If XtNposition is OL\_TOP or OL\_BOTTOM, then the alignment can be OL\_LEFT, OL\_CENTER, or OL\_RIGHT.

**XtNfont**

| Class   | Type   | Default           | Access |
|---------|--------|-------------------|--------|
| XtCFont | OlFont | OlDefaultBoldFont | SGID   |

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfont" on page 26.

**XtNfontColor**

| Class        | Type  | Default             | Access |
|--------------|-------|---------------------|--------|
| XtCFontColor | Pixel | XtDefaultForeground | SGID   |

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfontColor" on page 27.

*Caption Widget*

***XtNlabel***

| <b>Class</b> | <b>Type</b> | <b>Default</b>  | <b>Access</b> |
|--------------|-------------|-----------------|---------------|
| XtCLabel     | OlStr       | (instance name) | SGI           |

Synopsis: The label text.

Values: Any OlStr value valid in the current locale. NULL is the same as the empty string.

The label is displayed as given; no punctuation (such as a colon) is added. Control characters (other than spaces) are ignored without warning. For example, embedded newlines do not cause line breaks.

***XtNmnemonic***

| <b>Class</b> | <b>Type</b>   | <b>Default</b> | <b>Access</b> |
|--------------|---------------|----------------|---------------|
| XtCMnemonic  | unsigned char | '\0'           | SGI           |

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNmnemonic” on page 28.

***XtNposition***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCPosition  | OlDefine    | OL_LEFT        | SGI           |

Synopsis: The placement of the label in relation to the child widget.

Values: OL\_BOTTOM/"bottom" - The label is below the child widget.  
 OL\_LEFT/"left" - The label is to the left of the child widget.  
 OL\_RIGHT/"right" - The label is to the right of the child widget.  
 OL\_TOP/"top" - The label is above the child widget.

***XtNrecomputeSize***

| <b>Class</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtCRecomputeSize | Boolean     | TRUE           | SGI           |

Synopsis: The widget resize policy.

Values: TRUE/"true" - The widget resizes itself to accommodate changes in its children’s sizes due to changes in resources such as fonts or labels.  
 FALSE/"false" - The widget does not resize itself.

***XtNspace***

| Class    | Type      | Default | Access |
|----------|-----------|---------|--------|
| XtCSPACE | Dimension | 4       | SGI    |

Synopsis: The separation of the label from the child widget, in pixels.

Values:  $0 \leq XtNspace$

The separation of the label and child widget is shown in the following figure.

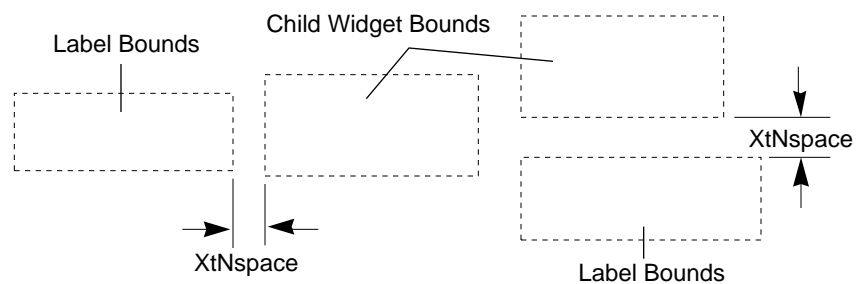


Figure 6-6 Label and Child Widget Spacing

***XtNtextFormat***

| Class         | Type     | Default       | Access |
|---------------|----------|---------------|--------|
| XtCTextFormat | OlStrRep | OL_SB_STR_REP | GI     |

The Caption widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNtextFormat” on page 29.

***Activation Types***

The following table lists the activation types used by the Caption.

Table 6-15 Caption Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |

*Table 6-15* Caption Activation Types (Continued)

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_SELECT              | SELECT           | XtNselectBtn         |
| OL_SELECTKEY           | SELECT           | XtNselectKey         |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

***OL\_SELECT/  
OL\_SELECTKEY***

The Caption widget does not respond to any user gestures, but a client can activate it with `OLActivateWidget()` and an activation type of `OL_SELECT` or `OL_SELECTKEY`. When so activated, the Caption widget will move focus to its child widget and then activate the child with the `OL_SELECT` activation type.

## CheckBox Widget

### Class

**Class Name:** CheckBox  
**Class Pointer:** checkBoxWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-CheckBox

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/CheckBox.h>
```

### Description

The CheckBox widget is similar in function to the RectButton widget. Several CheckBoxes are typically used together to provide the user with a set of options that can be toggled on or off.

### Components

The CheckBox widget consists of a label next to a Check Box; the Check Box will have a Check Mark, if selected.

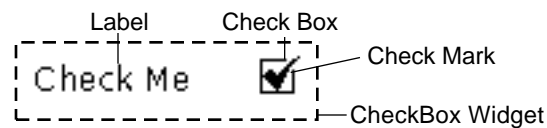


Figure 6-7 **CheckBox Widget**

*CheckBox Widget*

The following figure shows several buttons, in unselected and selected, as well as normal and dim states.



Figure 6-8 Check Boxes in Various States

**Typical Use**

Check Boxes may be used alone, but are usually used in the Nonexclusives composite widget, where they are used to implement a several-of-many selection. Making the CheckBox widget a child of a different composite widget will not produce an error, but proper behavior is not guaranteed.

**Operations**

A CheckBox widget has two states: “set” and “not set.” When set, the Check Mark is visible. Toggling this state alternates a resource (`XtNset`) between logical TRUE and FALSE and starts an action associated with the check box. Clicking SELECT on a check box toggles the state associated with it. Pressing SELECT, or moving the pointer into the check box while SELECT is pressed, adds or removes the Check Mark to reflect the state the check box would be in if SELECT was released. Releasing SELECT toggles the state. Moving the pointer off the check box before releasing SELECT restores the original Check Box, but does not toggle the state. Clicking or pressing MENU does not do anything in the CheckBox widget; the event is passed up to an ancestor widget.

**Bounds on SELECT**

Only the Check Box and Check Mark respond to SELECT, as shown in the following figure.



Figure 6-9 **CheckBox Widget**



### Coloration

For both 3D and 2D, the background of the CheckBox widget is drawn in the parent's `XtNbackground` resource. The label is drawn using `XtNfontColor`. The checkmark is drawn using `XtNforeground`.

For 3D, the check box component coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." The parent's `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNforeground` is used to render the outline of the check box component as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

If the toolkit resource `XtNmouseless` is set to TRUE and the toolkit resource `XtNinputFocusFeedback` is set to `OL_INPUT_FOCUS_COLOR`, then the background of the check box component will be drawn with the value of `XtNinputFocusColor` when the widget receives input focus. However, if `XtNinputFocusColor` is the same as the parent's `XtNbackground`, then the widget inverts `XtNforeground` and the parent's `XtNbackground` inside the check box component. Once the input focus leaves the widget, the original coloration is restored.

### Keyboard Traversal

The default value of the `XtNtraversalOn` resource is TRUE. The CheckBox widget responds to the following keyboard navigation keys:

- NEXTFIELD moves to the next traversable widget in the window
- PREVIOUS moves to the previous traversable widget in the window
- MOVEUP moves to the CheckBox above the current widget in the Nonexclusives composite
- MOVEDOWN moves to the CheckBox below the current widget in the Nonexclusives composite
- MOVELEFT moves to the CheckBox to the left of the current widget in the Nonexclusives composite
- MOVERIGHT moves to the CheckBox to the right of the current widget in the Nonexclusives composite
- NEXTWINDOW moves to the next window in the application
- PREVIOUSWINDOW moves to the previous window in the application
- NEXTAPP moves to the first window in the next application
- PREVIOUSAPP moves to the first window in the previous application

*CheckBox Widget*

***Keyboard Mnemonic Display***

The CheckBox widget displays its mnemonic as part of its label. If the mnemonic character is in the label, then that character is marked according to the value of the toolkit resource `XtNshowMnemonics`. If the mnemonic character is not in the label, it is displayed to the right of the label in parentheses and marked according to the value of `XtNshowMnemonics`.

If truncation is necessary, the mnemonic displayed in parentheses is truncated as a unit.

***Keyboard Accelerator Display***





The CheckBox widget displays the keyboard accelerator as part of its label. The string in the `XtNacceleratorText` resource is displayed to the right of the label (or mnemonic) separated by at least one space. The accelerator text is right justified.

If truncation is necessary, the accelerator is truncated as a unit. The accelerator is truncated before the mnemonic or the label.

***CheckBox Appearance***

The `XtNdim` and `XtNset` resources can be set independently, as shown in the following state table.

*Table 6-16* CheckBox Appearance with Set/Default/Dim

| XtNset | XtNdim | Check Box Appearance                                                                |
|--------|--------|-------------------------------------------------------------------------------------|
| TRUE   | TRUE   |  |
| TRUE   | FALSE  |  |
| FALSE  | TRUE   |  |
| FALSE  | FALSE  |  |

**Label Resource Interactions**

The `XtNwidth`, `XtNheight`, `XtNrecomputeSize`, and `XtNlabelJustify` resources interact to produce a truncated, clipped, centered, left-justified, or right-justified Label and Check Box as shown in the table below.

Table 6-17 CheckBox Label and Check Box Appearance

| XtNwidth           | XtNrecomputeSize  | XtNlabelJustify | Result                                              |
|--------------------|-------------------|-----------------|-----------------------------------------------------|
| any value          | TRUE              | any             | Just Fits <input checked="" type="checkbox"/>       |
| > needed for label | FALSE             | OL_LEFT         | Left Justified <input checked="" type="checkbox"/>  |
| > needed for label | FALSE             | OL_RIGHT        | Right Justified <input checked="" type="checkbox"/> |
| < needed for label | FALSE             | any             | Truncated <input type="checkbox"/>                  |
| XtNheight          | XtNrecomputerSize | XtNlabelJustify | Result                                              |
| any value          | TRUE              | any             | Just Fits <input checked="" type="checkbox"/>       |
| > needed for label | FALSE             | any             | Centered <input checked="" type="checkbox"/>        |
| < needed for label | FALSE             | any             | Clipped <input checked="" type="checkbox"/>         |

When the label is left-justified, right-justified, or centered the extra space is filled with the background color of the CheckBox widget's parent, as determined by the `XtNbackground` and `XtNbackgroundPixmap` resources of the parent. See also the `XtNlabelTile` resource for how it affects the appearance of a label.

**Resources**

Table 6-18 CheckBox Core Resources

| Name                              | Type             | Default                          | Access |
|-----------------------------------|------------------|----------------------------------|--------|
| <code>XtNaccelerators</code>      | AcceleratorTable | NULL                             | SGI    |
| <code>XtNancestorSensitive</code> | Boolean          | TRUE                             | G      |
| <code>XtNbackground</code>        | Pixel            | <code>XtDefaultBackground</code> | SGID   |
| <code>XtNbackgroundPixmap</code>  | Pixmap           | <code>XtUnspecifiedPixmap</code> | SGI    |

CheckBox Widget

Table 6-18 CheckBox Core Resources (Continued)

| Name                 | Type           | Default             | Access |
|----------------------|----------------|---------------------|--------|
| XtNborderColor       | Pixel          | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap         | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension      | 1                   | SGI    |
| XtNcolormap          | Colormap       | (parent's)          | SGI    |
| XtNdepth             | int            | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList | NULL                | SGIO   |
| XtNheight            | Dimension      | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean        | TRUE                | SGI    |
| XtNscreen            | Screen *       | (parent's)          | G      |
| XtNsensitive         | Boolean        | TRUE                | GIO    |
| XtNtranslations      | XtTranslations | NULL                | SGI    |
| XtNwidth             | Dimension      | 0                   | SGI    |
| XtNx                 | Position       | 0                   | SGI    |
| XtNy                 | Position       | 0                   | SGI    |

Table 6-19 CheckBox Composite Resources

| Name              | Type        | Default | Access |
|-------------------|-------------|---------|--------|
| XtNchildren       | WidgetList  | NULL    | G      |
| XtNinsertPosition | XtOrderProc | NULL    | SGI    |
| XtNnumChildren    | Cardinal    | 0       | G      |

Table 6-20 CheckBox Manager Resources

| Name                 | Type           | Default | Access |
|----------------------|----------------|---------|--------|
| XtNconsumeEvent      | XtCallbackList | NULL    | SGIO   |
| XtNinputFocusColor   | Pixel          | Red     | SGID   |
| XtNreferenceName     | String         | NULL    | GI     |
| XtNreferenceWidget   | Widget         | NULL    | GI     |
| XtNtraversalOn       | Boolean        | TRUE    | SGI    |
| XtNunrealizeCallback | XtCallbackList | NULL    | SGIO   |
| XtNuserData          | XtPointer      | NULL    | SGI    |

Table 6-21 CheckBox Resources

| Name               | Type           | Default             | Access |
|--------------------|----------------|---------------------|--------|
| XtNaccelerator     | String         | NULL                | SGI    |
| XtNacceleratorText | String         | NULL                | SGI    |
| XtNdim             | Boolean        | FALSE               | SGI    |
| XtNfont            | OlFont         | XtDefaultFont       | SGID   |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID   |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID   |
| XtNlabel           | OlStr          | (instance name)     | SGI    |
| XtNlabelImage      | XImage *       | NULL                | SGI    |
| XtNlabelJustify    | OlDefine       | OL_LEFT             | SGI    |
| XtNlabelTile       | Boolean        | FALSE               | SGI    |
| XtNlabelType       | OlDefine       | OL_STRING           | SGI    |
| XtNmnemonic        | unsigned char  | '\0'                | SGI    |
| XtNposition        | OlDefine       | OL_LEFT             | SGI    |
| XtNrecomputeSize   | Boolean        | TRUE                | SGI    |
| XtNscale           | int            | 12                  | SGI    |
| XtNselect          | XtCallbackList | NULL                | SGIO   |
| XtNset             | Boolean        | FALSE               | SGI    |
| XtNtextFormat      | OlStrRep       | OL_SB_STR_REP       | GI     |
| XtNunselect        | XtCallbackList | NULL                | SGIO   |

***XtNaccelerator***

| Class          | Type   | Default | Access |
|----------------|--------|---------|--------|
| XtCAccelerator | String | NULL    | SGI    |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNaccelerator” on page 25.

***XtNacceleratorText***

| Class              | Type   | Default | Access |
|--------------------|--------|---------|--------|
| XtCAcceleratorText | String | NULL    | SGI    |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNacceleratorText” on page 25.

*CheckBox Widget*

***XtNdim***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCDim       | Boolean     | FALSE          | SGID          |

Synopsis: The visual appearance of the CheckBox in reflecting the state of associated objects.

Values: TRUE/"true" - The check box border is dimmed to show that the check box represents the state of one or more of several objects that, as a group, are in different states.  
 FALSE/"false" - The border does not show the state of underlying objects.

***XtNfont***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCFont      | OlFont      | XtDefaultFont  | SGID          |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfont" on page 26.

***XtNfontColor***

| <b>Class</b> | <b>Type</b> | <b>Default</b>      | <b>Access</b> |
|--------------|-------------|---------------------|---------------|
| XtCFontColor | Pixel       | XtDefaultForeground | SGID          |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfontColor" on page 27.

***XtNforeground***

| <b>Class</b>  | <b>Type</b> | <b>Default</b>      | <b>Access</b> |
|---------------|-------------|---------------------|---------------|
| XtCForeground | Pixel       | XtDefaultForeground | SGID          |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNforeground" on page 27.

***XtNlabel***

| <b>Class</b> | <b>Type</b> | <b>Default</b>  | <b>Access</b> |
|--------------|-------------|-----------------|---------------|
| XtCLabel     | OlStr       | (instance name) | SGI           |

Synopsis: The text for the label.

Values: Any OlStr value valid in the current locale.

*CheckBox Widget*

The default value is the name of the widget as specified in the `XtCreateWidget()` routine. This resource will be ignored if the `XtNlabelType` resource has the value `OL_IMAGE`.

***XtNlabelImage***

| Class         | Type     | Default | Access |
|---------------|----------|---------|--------|
| XtCLabelImage | XImage * | NULL    | SGI    |

Synopsis: The image for the label of the CheckBox widget.

This resource will be ignored unless the `XtNlabelType` resource has the value `OL_IMAGE`. If the image is smaller than the space available for it next to the check box, it will be right- or left-justified horizontally, depending on the value of the `XtNlabelJustify` resource. If the image is larger than the space available for it, it will be clipped so that it does not display outside the space. See, however, `XtNlabelTile` for alternative behavior.

***XtNlabelJustify***

| Class           | Type     | Default | Access |
|-----------------|----------|---------|--------|
| XtCLabelJustify | OlDefine | OL_LEFT | SGI    |

Synopsis: The justification of the label, if the `XtNwidth` resource gives more space than needed.

Values: `OL_LEFT/"left"` - Left-justify the label.  
`OL_RIGHT/"right"` - Right-justify the label.

The label will be justified within the space available next to the Check Box.

***XtNlabelTile***

| Class        | Type    | Default | Access |
|--------------|---------|---------|--------|
| XtCLabelTile | Boolean | FALSE   | SGI    |

Synopsis: The tiling of the background of the CheckBox rectangle.

Values: `TRUE/"true"` - The label area is tiled with the image to fill the background if the image is smaller than the background of the subobject.  
`FALSE/"false"` - The label area is placed as described by the `XtNlabelImage` resource.

This resource augments the `XtNlabelImage` resource to allow tiling of the background. The `XtNlabelTile` resource is ignored for text labels.

*CheckBox Widget*

***XtNlabelType***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCLabelType | OlDefine    | OL_STRING      | SGI           |

Synopsis: The form that the label takes.

Values: OL\_STRING/"string" - The label is a text string.  
 OL\_IMAGE/"image" - The label is an image.

***XtNmnemonic***

| <b>Class</b> | <b>Type</b>   | <b>Default</b> | <b>Access</b> |
|--------------|---------------|----------------|---------------|
| XtCMnemonic  | unsigned char | '\0'           | SGI           |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNmnemonic" on page 28.

***XtNposition***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCPosition  | OlDefine    | OL_LEFT        | SGI           |

Synopsis: On which side of the CheckBox the label will be placed.

Values: OL\_LEFT/"left" - Place the label on the left.  
 OL\_RIGHT/"right" - Place the label on the right.

***XtNrecomputeSize***

| <b>Class</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtCRecomputeSize | Boolean     | TRUE           | SGI           |

Synopsis: Whether the CheckBox widget should calculate its size.

Values: TRUE/"true" - The CheckBox widget will do normal size calculations that may cause its geometry to change, and automatically set the XtNheight and XtNwidth resources.  
 FALSE/"false" - The CheckBox widget will leave its size unchanged; this may cause truncation of the visible image being shown by the CheckBox widget if the fixed size is too small, or may cause padding if the fixed size is too large. The location of the padding is determined by the XtNlabelJustify resource.



***XtNscale***

| Class    | Type | Default | Access |
|----------|------|---------|--------|
| XtCScale | int  | 12      | SGI    |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNscale” on page 29.

***XtNselect***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the user toggles the widget into “set” mode, making `XtNset` be TRUE. Simply setting `XtNset` to TRUE with a call to `XtSetValues()` does not issue the `XtNselect` callbacks.

***XtNset***

| Class  | Type    | Default | Access |
|--------|---------|---------|--------|
| XtCSet | Boolean | FALSE   | SGI    |

Synopsis: The current state of the check box.

Values: TRUE/“true” - The check mark will be present.  
FALSE/“false” - The check mark will not be present.

***XtNtextFormat***

| Class         | Type     | Default       | Access |
|---------------|----------|---------------|--------|
| XtCTextFormat | OIStrRep | OL_SB_STR_REP | GI     |

The CheckBox widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNtextFormat” on page 29.

***XtNunselect***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the user toggles the widget into “unset” mode, making `XtNset` be FALSE. Simply setting `XtNset` to FALSE with a call to `XtSetValues()` does not issue the `XtNunselect` callbacks.

CheckBox Widget

*Activation Types*

The following table lists the activation types used by the CheckBox.

*Table 6-22* CheckBox Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_SELECT        | SELECT        | XtNselectBtn        |
| OL_SELECTKEY     | SELECT        | XtNselectKey        |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

***OL\_SELECT/  
OL\_SELECTKEY***

The activation of a CheckBox is described in the *OPEN LOOK GUI Functional Specification* section “Check Boxes” in Chapter 4. When the CheckBox is activated with either OL\_SELECT or OL\_SELECTKEY, the state of the XtNset resource will be reversed. When the XtNset resource goes to FALSE, the XtNunselect callback will be called; when the XtNset resource goes to TRUE, the XtNselect callback will be called.

*See Also*

“RectButton Widget” on page 489.

## ControlArea Widget

### Class

*Class Name:* ControlArea  
*Class Pointer:* controlAreaWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-ControlArea

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/ControlAre.h>
```

### Description

The ControlArea is a composite widget that organizes its child controls in rows and columns.

### Components

The ControlArea widget has zero or more child widgets and an optional border. The ControlArea can also provide changebars alongside a child. Changebars are provided by the *OPEN LOOK GUI Functional Specification* as a means of feedback that a change has occurred in the corresponding object.

### Layout Control

The application can choose one of four simple layout schemes:

- Fixed number of columns in the control pane
- Fixed number of rows
- Fixed overall width of the control area
- Fixed overall height

The application can also specify the inter-control spacing and the size of the margin around the children.

*ControlArea Widget*

The children in each row align at the top of the row. The distance between the top of one row and the next is the height of the tallest child in the row plus the application-specified inter-row spacing.

*Coloration*

The following diagram illustrates the resources used for ControlArea coloration.

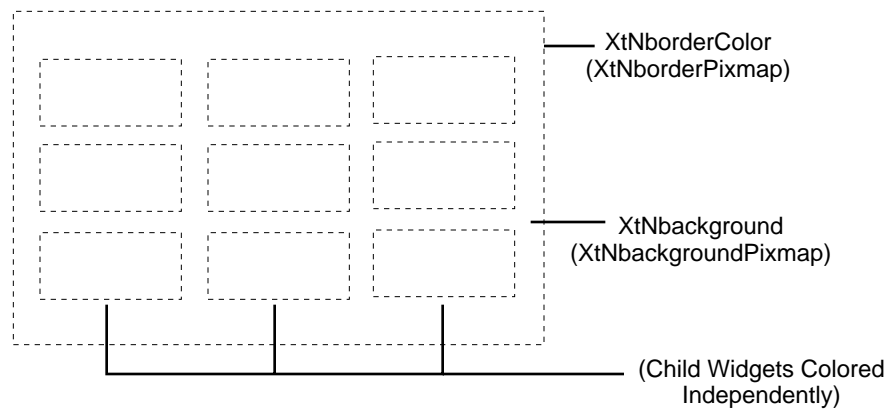


Figure 6-10 ControlArea Coloration

*Resources*

Table 6-23 ControlArea Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |

*ControlArea Widget**Table 6-23 ControlArea Core Resources (Continued)*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNheight            | Dimension      | 0              | SGI           |
| XtNmappedWhenManaged | Boolean        | TRUE           | SGI           |
| XtNscreen            | Screen *       | (parent's)     | G             |
| XtNsensitive         | Boolean        | TRUE           | GIO           |
| XtNtranslations      | XtTranslations | NULL           | SGI           |
| XtNwidth             | Dimension      | 0              | SGI           |
| XtNx                 | Position       | 0              | SGI           |
| XtNy                 | Position       | 0              | SGI           |

*Table 6-24 ControlArea Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 6-25 ControlArea Manager Resources*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor   | Pixel          | Red            | SGID          |
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 6-26 ControlArea Resources*

| <b>Name</b>        | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------------|-------------|----------------|---------------|
| XtNalignCaptions   | Boolean     | FALSE          | SGI           |
| XtNallowChangeBars | Boolean     | FALSE          | SGI           |
| XtNcenter          | Boolean     | FALSE          | SGI           |

*ControlArea Widget*

*Table 6-26 ControlArea Resources (Continued)*

| <b>Name</b>   | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtNhPad       | Dimension   | 4              | SGI           |
| XtNhSpace     | Dimension   | 4              | SGI           |
| XtNlayoutType | OlDefine    | OL_FIXEDROWS   | SGI           |
| XtNmeasure    | int         | 1              | SGI           |
| XtNsameSize   | OlDefine    | OL_COLUMNS     | SGI           |
| XtNvPad       | Dimension   | 4              | SGI           |
| XtNvSpace     | Dimension   | 4              | SGI           |

*Table 6-27 ControlArea Subwidget Resources*

| <b>Name</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtNchangeBar | OlDefine    | OL_NONE        | SGI           |

***XtNallowChangeBars***

| <b>Class</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------------|-------------|----------------|---------------|
| XtCAllowChangeBars | Boolean     | FALSE          | SGI           |

Synopsis: Whether the ControlArea displays changebars.  
 Values: TRUE/"true" - The ControlArea displays changebars.  
 FALSE/"false" - The ControlArea does not display changebars.

The ControlArea supports changebars only if the XtNlayoutType resource value is OL\_FIXEDCOLS or OL\_FIXEDROWS.

***XtNalignCaptions***

| <b>Class</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtCAlignCaptions | Boolean     | FALSE          | SGI           |

Synopsis: The alignment of Caption widgets in the ControlArea.  
 Values: TRUE/"true" - The ControlArea aligns all Caption widgets in each column so that their captions are right- or left-justified, depending on the position of the Caption's label (see Figure 6-11 on page 253). This may affect the width calculation for a column.  
 FALSE/"false" - The ControlArea aligns all Caption widgets the same as other widgets—by their overall width.

## ControlArea Widget

`XtNalignCaptions` takes precedence over the `XtNcenter` resource, but only for Caption widgets.

The effective width for the Caption widgets in a column becomes the sum of the width of the widest caption, plus the largest caption/child widget separation and child widget width. This alignment is only for groups of Caption widgets with all their captions on the left or the right.

Mixed orientation, or captions above or below, cannot be aligned well.

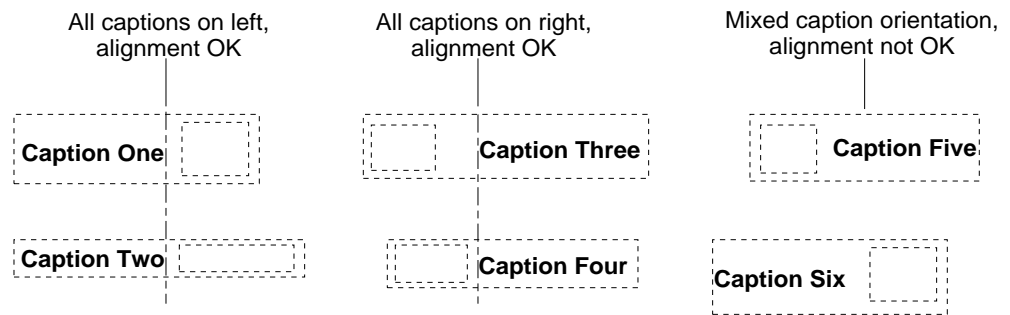


Figure 6-11 Aligning Captions

### ***XtNchangeBar***

| Class        | Type     | Default | Access |
|--------------|----------|---------|--------|
| XtCChangeBar | OlDefine | OL_NONE | SGI    |

Synopsis: The type of change bar to be displayed. This resource should be set for the child that requires the changebar.

Values: `OL_NONE`/"none" - Do not display any change bar  
`OL_NORMAL`/"normal" - Display a solid vertical line.  
`OL_DIM`/"dim" - Display a dimmed vertical line.

### ***XtNcenter***

| Class     | Type    | Default | Access |
|-----------|---------|---------|--------|
| XtCCenter | Boolean | FALSE   | SGI    |

Synopsis: The orientation of each widget within a ControlArea column.

Values: `TRUE`/"true" - Center each widget within each column.  
`FALSE`/"false" - Left-justify each widget within each column, except where the `XtNalignCaptions` resource applies.

ControlArea Widget

**XtNhPad/  
XtNvPad**

| Class   | Type      | Default | Access |
|---------|-----------|---------|--------|
| XtCHPad | Dimension | 4       | SGI    |
| XtCVPad | Dimension | 4       | SGI    |

Synopsis: The amount of padding, in pixels, to leave around the edges of the control area.

Values:  $0 \leq XtNhPad$   
 $0 \leq XtNvPad$

The action of these resources is illustrated by the following figure.

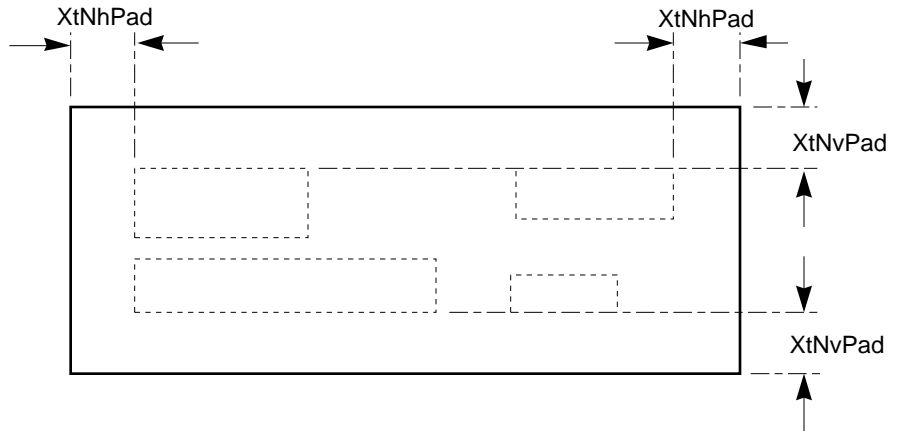


Figure 6-12 ControlArea Padding Around Controls

**XtNhSpace/  
XtNvSpace**

| Class     | Type      | Default | Access |
|-----------|-----------|---------|--------|
| XtCHSpace | Dimension | 4       | SGI    |
| XtCVSpace | Dimension | 4       | SGI    |

Synopsis: The amount of space, in pixels, to leave between controls.

Values:  $0 \leq XtNhSpace$   
 $0 \leq XtNvSpace$



If the controls are of different sizes in a row or column, the spacing applies to the widest or tallest dimension of all the controls.

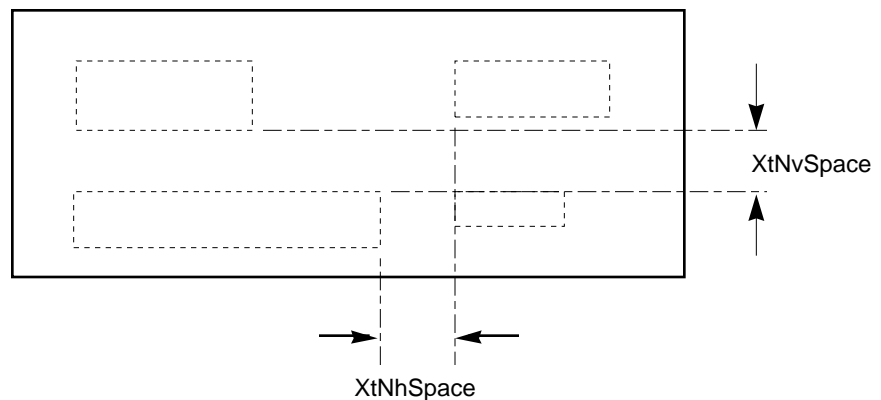


Figure 6-13 Spacing Between Controls

### ***XtNlayoutType***

| Class         | Type     | Default      | Access |
|---------------|----------|--------------|--------|
| XtCLayoutType | OlDefine | OL_FIXEDROWS | SGI    |

Synopsis: The layout of the child widgets by the ControlArea.

Values:

- OL\_FIXEDROWS/"fixedrows" - The layout will have a fixed number of rows and enough columns to hold all the controls;
- OL\_FIXEDCOLS/"fixedcols" - The layout will have a fixed number of columns and enough rows to hold all the controls;
- OL\_FIXEDWIDTH/"fixedwidth" - The layout will be of a fixed width, but tall enough to hold all the controls;
- OL\_FIXEDHEIGHT/"fixedheight" - The layout will be of a fixed height, but wide enough to hold all the controls.

The XtNmeasure resource gives the number of rows or columns or the fixed height or width, in pixels.

The choices are to specify the number of rows or columns, or to specify the overall height or width of the layout area. Only one of these dimensions can be specified directly; the other is determined by the number of controls added.

For instance, if the application specifies that the control area should have four columns, the number of rows will be the number of controls divided by four.

ControlArea Widget

**XtNmeasure**

| Class      | Type | Default | Access |
|------------|------|---------|--------|
| XtCMeasure | int  | 1       | SGI    |

Synopsis: The number of rows or columns in the layout of the child widgets, or the fixed width or height of the control area.

Values: 0 < XtNmeasure

If XtNlayoutType is OL\_FIXEDROWS or OL\_FIXEDCOLS, then the default is 1. If XtNlayoutType is OL\_FIXEDWIDTH or OL\_FIXEDHEIGHT, then the default is the width or height of the widest or tallest widget, depending on XtNlayoutType.

When XtNlayoutType is OL\_FIXEDWIDTH or OL\_FIXEDHEIGHT, the measure includes the padding on both edges and the inter-control spacing, as suggested by the following figure.

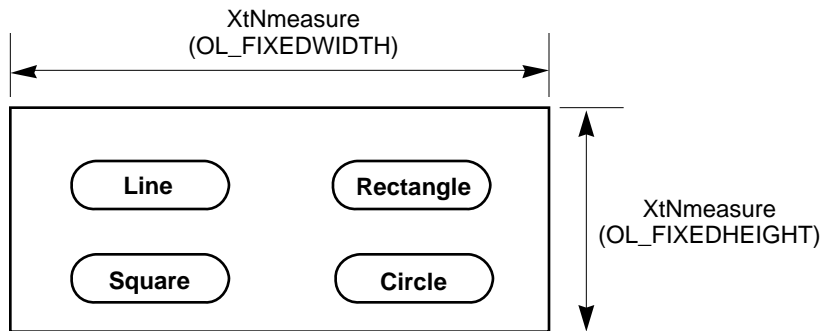


Figure 6-14 XtNmeasure

**XtNsameSize**

| Class       | Type     | Default    | Access |
|-------------|----------|------------|--------|
| XtCSameSize | OlDefine | OL_COLUMNS | SGI    |

Synopsis: The controls within the ControlArea widget that are forced to be the same width (if any).

Values: OL\_NONE/"none" - The controls are placed in fixed-width columns, but the size of each control is left alone. The width of each column is the width of the widest control in the column.  
 OL\_COLUMNS/"columns" - Controls in each column are made the same width as the widest of them. The width of each column is

---

*ControlArea Widget*

thus the width of the widest control in the column.

OL\_ALL/"all" - All controls are made the same width, the width of the widest control in the ControlArea widget.

---

**Note** - The ControlArea widget does not resize its children when it gets resized. Use the Form or RubberTile widget if the children need to be resized.

---

## Activation Types

The following table lists the activation types used by the ControlArea.

*Table 6-28* ControlArea Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

The ControlArea widget has no activation types besides the ones in “Common Activation Types” on page 68.

## See Also

“BulletinBoard Widget” on page 225,  
“Caption Widget” on page 229,  
“Form Widget” on page 385,  
“RubberTile Widget” on page 502.

## ≡ 6

---

### *ControlArea Widget*

### *DrawArea Widget*

#### *Class*

*Class Name:* DrawArea  
*Class Pointer:* drawAreaWidgetClass

#### *Ancestry*

Core-Composite-Constraint-Manager-BulletinBoard-DrawArea

#### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/DrawArea.h>
```

#### *Description*

The DrawArea widget provides a window on which an application can render images using Xlib calls. The DrawArea widget uses callbacks to notify the application that the image needs to be redrawn. To get the window associated with the DrawArea widget, the application can use the `XtWindow()` function; see the *Xt Intrinsic Programming Manual* for details on this function.

## DrawArea Widget

The DrawArea widget is a composite widget that enforces no ordering on its children. It is up to the application to specify the x- and y-coordinates of each child inserted; otherwise, a child will be placed in the upper left corner of the DrawArea widget.

The DrawArea widget can be mapped with no children. It displays an empty space, possibly surrounded by a border.

### Multiple Visuals Support

The DrawArea widget can be created with a nondefault depth, visual, and colormap. This can be done by setting the `XtNdepth`, `XtNvisual`, or `XtNcolormap` resources of the widget.

If `XtNdepth` is not specified, a DrawArea widget will use the depth of its parent. If `XtNvisual` is not specified, a DrawArea widget will use the visual of its parent. If `XtNcolormap` is not specified, a DrawArea widget will use its visual resource to find (share or create) the colormap of the widget.

### Coloration

The following diagram illustrates the resources used for DrawArea coloration.

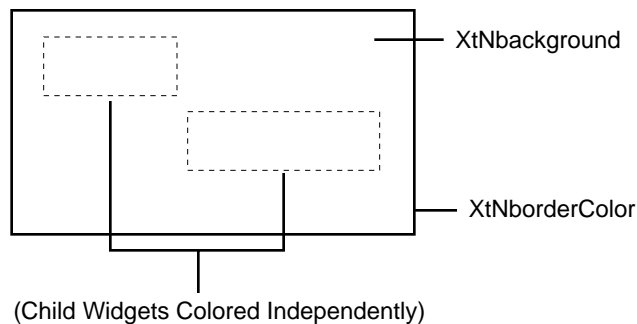


Figure 7-1 DrawArea Coloration

### Keyboard Traversal

The DrawArea widget is a composite widget and cannot be accessed via keyboard traversal. Input focus moves between the children of this widget.

## Resources

Table 7-1 DrawArea Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | 0                   | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

Table 7-2 DrawArea Composite Resources

| Name              | Type        | Default | Access |
|-------------------|-------------|---------|--------|
| XtNchildren       | WidgetList  | NULL    | G      |
| XtNinsertPosition | XtOrderProc | NULL    | SGI    |
| XtNnumChildren    | Cardinal    | 0       | G      |

Table 7-3 DrawArea Manager Resources

| Name               | Type           | Default | Access |
|--------------------|----------------|---------|--------|
| XtNconsumeEvent    | XtCallbackList | NULL    | SGIO   |
| XtNinputFocusColor | Pixel          | Red     | (n/a)  |

## DrawArea Widget

Table 7-3 DrawArea Manager Resources (Continued)

| Name                 | Type           | Default | Access |
|----------------------|----------------|---------|--------|
| XtNreferenceName     | String         | NULL    | GI     |
| XtNreferenceWidget   | Widget         | NULL    | GI     |
| XtNtraversalOn       | Boolean        | TRUE    | SGI    |
| XtNunrealizeCallback | XtCallbackList | NULL    | SGIO   |
| XtNuserData          | XtPointer      | NULL    | SGI    |

Table 7-4 DrawArea BulletinBoard Resources<sup>1</sup>

| Name      | Type     | Default     | Access |
|-----------|----------|-------------|--------|
| XtNlayout | OlDefine | OL_MINIMIZE | SGI    |

1. This resource is described in "BulletinBoard Widget" on page 225.

Table 7-5 DrawArea Resources

| Name                      | Type           | Default             | Access |
|---------------------------|----------------|---------------------|--------|
| XtNexposeCallback         | XtCallbackList | NULL                | SGIO   |
| XtNforeground             | Pixel          | XtDefaultForeground | SGID   |
| XtNgraphicsExposeCallback | XtCallbackList | NULL                | SGIO   |
| XtNresizeCallback         | XtCallbackList | NULL                | SGIO   |
| XtNvisual                 | Visual *       | (parent's)          | GI     |

### XtNexposeCallback

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

**Synopsis:** The callback list invoked whenever the DrawArea widget gets an expose event.

The *call\_data* parameter is a pointer to an `OlDrawAreaCallbackStruct` structure:

```
typedef struct {
 int reason;
 XEvent *event;
 Position x;
```



```

 Position y;
 Dimension width;
 Dimension height;
 Region region;
} OldrawAreaCallbackStruct;

```

**reason** OL\_REASON\_EXPOSE, indicating the validity of the remaining fields of the OldrawAreaCallbackStruct structure.

**event** A pointer to the XEvent that triggered the callback.

**x, y** The upper left corner of the rectangle enclosing the exposed region.

**width, height** The dimensions of the rectangle enclosing the exposed region.

**region** A union of all the areas obscured.

### ***XtNforeground***

| Class         | Type  | Default             | Access |
|---------------|-------|---------------------|--------|
| XtCforeground | Pixel | XtDefaultForeground | SGID   |

**Synopsis:** The foreground color used by objects to draw widget contents.

**Values:** Any Pixel value valid for the current display, or any name from the \$OPENWINHOME/lib/rgb.txt file. (Pixel values are used in C programs, rgb.txt values in X resource files. See “Resource Files” on page 7.)

### ***XtNgraphicsExposeCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

**Synopsis:** The callback list invoked whenever the DrawArea widget gets a graphics expose event. Graphics expose events are generated when a XCopyArea or XCopyPlane fails to copy the entire source area because part of the source area was obscured.

The *call\_data* parameter is a pointer to an OldrawAreaCallbackStruct structure, as shown in “XtNexposeCallback” on page 262. The members are:

**reason** OL\_REASON\_EXPOSE, indicating the validity of the remaining fields of the OldrawAreaCallbackStruct structure.

**event** A pointer to the XEvent that triggered the callback.

**x, y** The upper left corner of the rectangle enclosing the obscured region.

## DrawArea Widget

*width, height* The dimensions of the rectangle enclosing the obscured region.  
*region* A union of all the areas obscured.

### **XtNresizeCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked whenever the DrawArea widget gets a resize event.

The *call\_data* parameter is a pointer to an `OlDrawAreaCallbackStruct` structure, as shown in “XtNexposeCallback” on page 262. The members are:

*reason* OL\_REASON\_RESIZE, indicating the validity of the remaining fields of the `OlDrawAreaCallbackStruct` structure.  
*event* Invalid field.  
*x, y* The upper left corner of the rectangle enclosing the obscured region.  
*width, height* The dimensions of the DrawArea.  
*region* Invalid field.

### **XtNvisual**

| Class     | Type     | Default    | Access |
|-----------|----------|------------|--------|
| XtCVisual | Visual * | (parent's) | GI     |

Synopsis: The visual used to create the widget's window.

Values: A pointer to any visual structure supported by the current display and compatible with the widget's depth and colormap.

Only Shell and DrawArea Widgets have a visual resource. All other widgets are created using their parent's visual. If `XtNvisual` is not specified, Shell and DrawArea widgets inherit their parent's visual.

The recommended method of setting a Shell or DrawArea widget's visual resource is to use the Intrinsic typed args interface. A string containing the desired Visual Class Name should be passed to the String-to-Visual resource converter.

To get the visual associated with any widget or gadget, use the function `OlVisualOfObject()`.

## *Activation Types*

The following table lists the activation types used by the DrawArea.

*Table 7-6* DrawArea Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The DrawArea widget has no activation types besides the ones in “Common Activation Types” on page 68.

## *See Also*

“BulletinBoard Widget” on page 225.

## *DropTarget Widget*

### *DropTarget Widget*

#### *Class*

*Class Name:* DropTarget  
*Class Pointer:* dropTargetWidgetClass

#### *Ancestry*

Core-Primitive-Pixmap-DropTarget

#### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/DropTarget.h>
```

#### *Description*

The DropTarget widget provides an alternative to the Drag and Drop utility functions. It is intended for those cases where an application does not have an obvious drop site.

The DropTarget associates one drop site with a single drop rectangle, providing visual indication of the progress of the Drag and Drop operation.

This differs from the Drag and Drop utility function library in that the utilities allow multiple drop rectangles to be associated with one drop site, and the utilities provide no visual indication.

See “Drag and Drop Functions” on page 109 for more information on terminology.

#### *Coloration*

DropTarget coloration is defined by the “User Interface Specification for Drag and Drop in the OPEN LOOK GUI.” `XtNforeground` is used to render the “full” pixmap inside the DropTarget.

For 3D, `XtNbackground` is used for BG1 (the interior region of the DropTarget). The BG3 (shadow) and Highlight colors are derived by the toolkit

*DropTarget Widget*

from BG1 and are used to draw the three-dimensional outline around the DropTarget.

For 2D, XtNforeground is used to render the two-dimensional outline of the DropTarget.



Figure 7-2 Drop Target Coloration

## Resources

Table 7-7 DropTarget Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 0                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | 0                   | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

*DropTarget Widget*

*Table 7-8 DropTarget Primitive Resources*

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|--------------------|----------------|---------------------|---------------|
| XtNaccelerator     | String         | NULL                | SGI           |
| XtNacceleratorText | String         | NULL                | SGI           |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor | Pixel          | Red                 | SGID          |
| XtNmnemonic        | unsigned char  | '\0'                | SGI           |
| XtNreferenceName   | String         | NULL                | GI            |
| XtNreferenceWidget | Widget         | NULL                | GI            |
| XtNscale           | int            | 12                  | SGI           |
| XtNtextFormat      | OlStrRep       | OL_SB_STR_REP       | GI            |
| XtNtraversalOn     | Boolean        | FALSE               | SGI           |
| XtNuserData        | XtPointer      | NULL                | SGI           |

*Table 7-9 DropTarget Pixmap Resources*

| <b>Name</b>      | <b>Type</b> | <b>Default</b>      | <b>Access</b> |
|------------------|-------------|---------------------|---------------|
| XtNpixmap        | Pixmap      | XtUnspecifiedPixmap | SGI           |
| XtNrecomputeSize | Boolean     | FALSE               | SGI           |

*Table 7-10 DropTarget Resources*

| <b>Name</b>           | <b>Type</b>           | <b>Default</b>       | <b>Access</b> |
|-----------------------|-----------------------|----------------------|---------------|
| XtNbusyPixmap         | Pixmap                | XtUnspecifiedPixmap  | SGI           |
| XtNdndAcceptCursor    | Cursor                | NULL                 | SGI           |
| XtNdndAnimateCallback | XtCallbackList        | NULL                 | SGI           |
| XtNdndCopyCursor      | Cursor                | openlook             | SGI           |
| XtNdndMoveCursor      | Cursor                | openlook             | SGI           |
| XtNdndPreviewHints    | OldnDSitePreviewHints | OldnDSitePreviewNone | SGI           |
| XtNdndPreviewCallback | XtCallbackList        | NULL                 | SGI           |
| XtNdndRejectCursor    | Cursor                | NULL                 | SGI           |

Table 7-10 DropTarget Resources (Continued)

| Name                    | Type           | Default | Access |
|-------------------------|----------------|---------|--------|
| XtNdndTriggerCallback   | XtCallbackList | NULL    | SGI    |
| XtNfull                 | Boolean        | FALSE   | SGI    |
| XtNownSelectionCallback | XtCallbackList | NULL    | SGI    |
| XtNselectionAtom        | Atom           | NULL    | SGI    |

### OldDropTargetCallbackStruct

The following structure is used by all of the callbacks for the DropTarget widget.

```
typedef struct {
 int reason;
 Widget widget;
 Window window;
 Position root_x;
 Position root_y;
 Atom selection;
 Time time;
 OldDnDDropSiteID dropsiteid;
 OldDnDTriggerOperation operation;
 Boolean send_done;
 Boolean forwarded;
 int eventcode;
 Boolean sensitivity;
} OldDropTargetCallbackStruct;
```

**reason** Supplies information necessary for the callback to make use of the remaining fields in the structure. This field can be one of the following values:

```
OL_REASON_NONE
OL_REASON_EXPOSE
OL_REASON_GRAPHICS_EXPOSE
OL_REASON_RESIZE
OL_REASON_DND_PREVIEW
OL_REASON_DND_TRIGGER
OL_REASON_DND_OWNSELECTION
OL_REASON_DND_ANIMATE
```

**widget** The owning widget of the drop site

**window** The owning window of the drop site

**root\_x** The root-relative x-coordinate at which the drop occurred

## DropTarget Widget

|                     |                                                                                                                                                                                                                                                                                                                                      |                     |                                 |                     |                                    |              |                                         |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|---------------------------------|---------------------|------------------------------------|--------------|-----------------------------------------|
| <i>root_y</i>       | The root-relative y-coordinate at which the drop occurred                                                                                                                                                                                                                                                                            |                     |                                 |                     |                                    |              |                                         |
| <i>selection</i>    | The Selection Atom for the selection transfer of the drop information                                                                                                                                                                                                                                                                |                     |                                 |                     |                                    |              |                                         |
| <i>time</i>         | The timestamp of the pertinent event; for example, the trigger or preview                                                                                                                                                                                                                                                            |                     |                                 |                     |                                    |              |                                         |
| <i>dropsiteid</i>   | The ID of the drop site on which the drop occurred                                                                                                                                                                                                                                                                                   |                     |                                 |                     |                                    |              |                                         |
| <i>operation</i>    | One of the following values:<br><table> <tr> <td>OldDnDTriggerCopyOp</td> <td>The object is copied</td> </tr> <tr> <td>OldDnDTriggerMoveOp</td> <td>The object is moved</td> </tr> </table>                                                                                                                                          | OldDnDTriggerCopyOp | The object is copied            | OldDnDTriggerMoveOp | The object is moved                |              |                                         |
| OldDnDTriggerCopyOp | The object is copied                                                                                                                                                                                                                                                                                                                 |                     |                                 |                     |                                    |              |                                         |
| OldDnDTriggerMoveOp | The object is moved                                                                                                                                                                                                                                                                                                                  |                     |                                 |                     |                                    |              |                                         |
| <i>send_done</i>    | If TRUE, the selection holder expects to be notified at the end of the selection transaction that it has been completed and that no further transactions associated with this drop will occur. This notification is achieved by calling <code>OldDnDDragNDropDone()</code> when the selection transaction is completed successfully. |                     |                                 |                     |                                    |              |                                         |
| <i>forwarded</i>    | The drop has been forwarded to this target by another drop site.                                                                                                                                                                                                                                                                     |                     |                                 |                     |                                    |              |                                         |
| <i>eventcode</i>    | One of the following values:<br><table> <tr> <td>LeaveNotify</td> <td>The cursor has left a drop site</td> </tr> <tr> <td>EnterNotify</td> <td>The cursor has entered a drop site</td> </tr> <tr> <td>MotionNotify</td> <td>The cursor is moving within a drop site</td> </tr> </table>                                              | LeaveNotify         | The cursor has left a drop site | EnterNotify         | The cursor has entered a drop site | MotionNotify | The cursor is moving within a drop site |
| LeaveNotify         | The cursor has left a drop site                                                                                                                                                                                                                                                                                                      |                     |                                 |                     |                                    |              |                                         |
| EnterNotify         | The cursor has entered a drop site                                                                                                                                                                                                                                                                                                   |                     |                                 |                     |                                    |              |                                         |
| MotionNotify        | The cursor is moving within a drop site                                                                                                                                                                                                                                                                                              |                     |                                 |                     |                                    |              |                                         |
| <i>sensitivity</i>  | Indicates whether the drop site the cursor has entered is interested in receiving drops.                                                                                                                                                                                                                                             |                     |                                 |                     |                                    |              |                                         |

Since `OldDropTargetCallbackStruct` is shared among all the callbacks of the `DropTarget` widget, not all the fields of `OldDropTargetCallbackStruct` are valid for all the callbacks of the `DropTarget` widget. For example, the *selection* field has no relevance for `XtNpreviewCallback`. The following table enumerates the validity of various fields of `OldDropTargetCallbackStruct` for all the callbacks of the `DropTarget` widget. The rows of the table contain fields of `OldDropTargetCallbackStruct`. The columns of the table contain the callbacks of the `DropTarget` widget. A '+' in a cell indicates the field in that row is relevant for the callback in that column, a '-' indicates that the field in that row should not be accessed by the callback in that column.

Table 7-11 OldDropTargetCallbackStruct Field Validity

| Field         | Preview | Trigger | Ownselection | Animate |
|---------------|---------|---------|--------------|---------|
| <i>reason</i> | +       | +       | +            | +       |
| <i>widget</i> | +       | +       | +            | +       |



Table 7-11 OldDropTargetCallbackStruct Field Validity (Continued)

| Field              | Preview | Trigger | Ownselection | Animate |
|--------------------|---------|---------|--------------|---------|
| <i>window</i>      | +       | +       | –            | –       |
| <i>root_x</i>      | +       | +       | –            | –       |
| <i>root_y</i>      | +       | +       | –            | –       |
| <i>selection</i>   | –       | +       | –            | –       |
| <i>time</i>        | +       | +       | +            | +       |
| <i>dropsiteid</i>  | +       | +       | –            | –       |
| <i>operation</i>   | –       | +       | –            | –       |
| <i>send_done</i>   | –       | +       | –            | –       |
| <i>forwarded</i>   | +       | +       | –            | –       |
| <i>eventcode</i>   | +       | –       | –            | +       |
| <i>sensitivity</i> | –       | –       | –            | +       |

***XtNbusyPixmap***

| Class         | Type   | Default             | Access |
|---------------|--------|---------------------|--------|
| XtCBusyPixmap | Pixmap | XtUnspecifiedPixmap | SGI    |

Synopsis: The image displayed when the DropTarget is busy.

***XtNdndAcceptCursor***

| Class     | Type   | Default | Access |
|-----------|--------|---------|--------|
| XtCCursor | Cursor | NULL    | SGI    |

Synopsis: The cursor used to indicate visually that the *destination* drop site will accept a drop.

Values: Any valid cursor, but the value is OPEN LOOK specified.

This resource is used by *source* DropTarget widgets only. It defines the cursor displayed when the mouse pointer moves over a drop site (during a drag operation) that is willing to accept a drop.

In a Drag and Drop operation, it is the responsibility of the *source* to change the cursor feedback as the mouse pointer moves over the various drop sites on the display (see *XtNdndAnimateCallback*).

DropTarget Widget

**XtNdnAnimateCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGI    |

Synopsis: The callback list used by a *source* DropTarget to indicate the readiness of the *destination*.

This callback list is used by the *source* DropTarget widget. A callback procedure registered on this callback list is invoked for animating the drag cursor as the mouse pointer moves over the various drop sites on a display. The callback should make sure that the feedback of the drag cursor reflects the willingness of the drop site under the mouse pointer to accept a drop. (See XtNdnAcceptCursor and XtNdnRejectCursor.)

**XtNdnCopyCursor**

| Class     | Type   | Default  | Access |
|-----------|--------|----------|--------|
| XtCCursor | Cursor | openlook | SGI    |

Synopsis: The cursor used to indicate visually that the drop operation is a copy.

Values: Any valid cursor, but the value is OPEN LOOK specified.

This resource is used by *source* DropTarget widgets to indicate whether the operation is a copy (or a move).

**XtNdnMoveCursor**

| Class     | Type   | Default  | Access |
|-----------|--------|----------|--------|
| XtCCursor | Cursor | openlook | SGI    |

Synopsis: The cursor used to indicate visually that the drop operation is a move.

Values: Any valid cursor, but the value is OPEN LOOK specified.

This resource is used by *source* DropTarget widgets to indicate whether the operation is a copy (or a move).

**XtNdnPreviewHints**

| Class              | Type                  | Default              | Access |
|--------------------|-----------------------|----------------------|--------|
| XtCDndPreviewHints | OIDnDSitePreviewHints | OIDnDSitePreviewNone | SGI    |

Synopsis: When a DropTarget preview callback is activated.

Values: One of the following values:

---

*DropTarget Widget*

```

OldnDSTargetPreviewNone
OldnDSTargetPreviewEnterLeave
OldnDSTargetPreviewMotion
OldnDSTargetPreviewBoth
OldnDSTargetPreviewDefaultSite
OldnDSTargetPreviewForwarded
OldnDSTargetPreviewInsensitive

```

This resource is used by *destination* DropTarget widgets only.

See “Drag and Drop Functions” for more details, in particular the section “OldnDSTargetPreviewHints” on page 117.

***XtNdnPreviewCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGI    |

Synopsis: The callback list used by the *destination* DropTarget to indicate the readiness of the destination.

This callback list is used by a *destination* DropTarget widget. A callback procedure registered on this callback list is invoked for changing the feedback of the *destination* widget (usually by changing the background pixmap) as a mouse pointer moves over it during a drag operation. The callback should make sure that the feedback of the *destination* widget reflects the willingness of the *destination* to accept a drop (see XtNbusyPixmap).

***XtNdnRejectCursor***

| Class     | Type   | Default | Access |
|-----------|--------|---------|--------|
| XtCCursor | Cursor | NULL    | SGI    |

Synopsis: The cursor used to indicate visually that the *source* DropTarget will reject a drop.

Values: Any valid cursor, but the value is OPEN LOOK specified.

This resource is used by *source* DropTarget widgets only. It defines the cursor displayed when the mouse pointer moves over a drop site (during a drag operation) that is not willing to accept a drop. In a Drag and Drop operation it is the responsibility of the *source* to change the cursor feedback as the mouse pointer moves over the various drop sites on the display (see XtNdnAnimateCallback).

DropTarget Widget

**XtNdndTriggerCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGI    |

Synopsis: The callback list invoked by the *destination* DropTarget to perform the data transfer.

This callback list is used by a *destination* DropTarget widget only. A callback procedure registered on this callback list is invoked after a drop has occurred on the *destination* widget. The callback procedure is responsible for initiating the data transfer associated with the drop.

A callback procedure on the XtNdndTriggerCallback list is invoked after all the callback procedures on the XtNownSelectionCallback callback list of the *source* DropTarget widget have been invoked (see XtNownSelectionCallback).

**XtNfull**

| Class   | Type    | Default | Access |
|---------|---------|---------|--------|
| XtCFull | Boolean | FALSE   | SGI    |

Synopsis: The eligibility of this DropTarget to be *source*.

Values: TRUE/"true" - This DropTarget is "full" of data and can be the *source* of a Drag and Drop operation. The pixmap specified by the XtNpixmap resource is displayed.  
 FALSE/"false" - The DropTarget is "empty" and cannot be a *source*. No pixmap is displayed.

**XtNownSelectionCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGI    |

Synopsis: The callback list used for the Selection Atom to send data.

This callback list is used by a *source* DropTarget widget. A callback procedure registered on this callback list is invoked after a drop has occurred on some *destination*. The callback procedure must perform the following actions:

1. Allocate an X11 selection atom using OldNdAllocTransientAtom() (see page 122).

*DropTarget Widget*

2. Set its `XtNselectionAtom` resource to the ID of the atom allocated in step 1. The ID of the selection atom is passed along to the *destination* by the toolkit. This in turn allows the *destination* to initiate the data transfer operation associated with the drop.
3. Obtain ownership of the X11 selection atom allocated in step 1 by calling `OldDnDownSelection()` (see page 126).

***XtNrecomputeSize***

| Class                         | Type    | Default | Access |
|-------------------------------|---------|---------|--------|
| <code>XtCRecomputeSize</code> | Boolean | FALSE   | SGI    |

Synopsis: The size of the DropTarget widget.

Values: `TRUE`/`"true"` - The DropTarget has the height and width of the pixmap specified by `XtNpixmap`. In this case the `XtNheight` and `XtNwidth` resources are ignored.  
`FALSE`/`"false"` - The DropTarget has the default height and width specified by OPEN LOOK.

***XtNselectionAtom***

| Class                         | Type | Default | Access |
|-------------------------------|------|---------|--------|
| <code>XtCSelectionAtom</code> | Atom | NULL    | SGI    |

Synopsis: The Selection Atom used to carry the data object.

This resource is used by *source* DropTarget widgets only. The atom is obtained by the `XtNownSelectionCallback`, which also sets this resource.

*DropTarget Widget*

*Activation Types*

The following table lists the activation types used by the DropTarget.

*Table 7-12 DropTarget Activation Types*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The DropTarget widget has no activation types besides the ones in “Common Activation Types” on page 68.

*See Also*

“Drag and Drop Functions” on page 109.

## Exclusives Widget

### Class

**Class Name:** Exclusives  
**Class Pointer:** exclusivesWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-Exclusives

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/Exclusives.h>
```

### Description

The Exclusives widget provides a simple way to build a one-of-many button selection object. It provides layout management for a set of rectangular buttons.

### Grid Layout and Button Labels

Sample exclusive button widgets are shown in the following figure.

|        |            |            |
|--------|------------|------------|
| Apple  | Strawberry | Pear       |
| Orange | Plum       | Watermelon |

Figure 7-3 Exclusive Buttons Example

The Exclusives widget lays out the rectangular buttons in a grid in the order they were added as child widgets by the application. The number of rows or columns in this grid can be controlled by the application.

If the grid has more than one row, the Exclusives widget forces the buttons in each column to be the same size as the widest in the column.

## Exclusives Widget

---

If the grid has a single row, each button will be only as wide as necessary to display the label.

### *Selection Control*

- When `XtNnoneSet` is `FALSE`, exactly one button in an Exclusives widget must be set (its `XtNset` resource is set to `TRUE`). An error is generated if an Exclusives is configured with two or more rectangular buttons set or with no button set. The Exclusives widget maintains this condition by ensuring that when a button is set by the user clicking `SELECT` over it, the button that was set is cleared and its `XtNunselect` callbacks are invoked. However, clicking `SELECT` over a button that was already set does nothing.
- When `XtNnoneSet` is `TRUE`, at most one button in an Exclusives widget can be set. An error is generated if an Exclusives is configured with two or more rectangular buttons set, but not if configured with no button set. The Exclusives widget maintains this condition by ensuring that when a button is set by the user clicking `SELECT` over it, any button that was previously set is cleared. Also, clicking `SELECT` over a button that was already set will unset it. Clearing a button in either case invokes its `XtNunselect` callbacks.

### *Menu Use*

The Exclusives widget can be added as a single child to a menu pane to implement a one-of-many menu choice.

### *Child Constraint*

The Exclusives widget constrains its child widgets to be of the class `rectButtonWidgetClass`.

### *Coloration*

When the Exclusives widget manages a number of children that is not a multiple of `XtNmeasure`, the empty space is colored with the value of the `XtNbackground` or `XtNbackgroundPixmap` resource.



### Keyboard Traversal

The Exclusives widget manages the traversal between a set of RectButton widgets. When the user traverses to an Exclusives widget, the first RectButton in the set will receive input focus.

The MOVEUP, MOVEDOWN, MOVERIGHT, and MOVELEFT keys move the input focus between the RectButtons. To traverse out of the Exclusives widget, the following keys can be used:

- NEXTFIELD moves to the next traversable widget in the window
- PREVFIELD moves to the previous traversable widget in the window
- NEXTWINDOW moves to the next window in the application
- PREVWINDOW moves to the previous window in the application
- NEXTAPP moves to the first window in the next application
- PREVAPP moves to the first window in the previous application
- SELECTKEY acts as if the SELECT button had been clicked on the RectButton with input focus
- MENUKEY acts as if the MENU button had been clicked on the RectButton with input focus

### Resources

Table 7-13 Exclusives Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |

*Exclusives Widget*

*Table 7-13 Exclusives Core Resources (Continued)*

| <b>Name</b>     | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|-----------------|----------------|----------------|---------------|
| XtNsensitive    | Boolean        | TRUE           | GIO           |
| XtNtranslations | XtTranslations | NULL           | SGI           |
| XtNwidth        | Dimension      | 0              | SGI           |
| XtNx            | Position       | 0              | SGI           |
| XtNy            | Position       | 0              | SGI           |

*Table 7-14 Exclusives Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 7-15 Exclusives Manager Resources*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor   | Pixel          | Red            | SGID          |
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 7-16 Exclusives Resources*

| <b>Name</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtNlayoutType    | OlDefine    | OL_FIXEDROWS   | SGI           |
| XtNmeasure       | int         | 1              | SGI           |
| XtNnoneSet       | Boolean     | FALSE          | SGI           |
| XtNrecomputeSize | Boolean     | TRUE           | SGI           |

***XtNlayoutType***

| <b>Class</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtCLayoutType | OIDefine    | OL_FIXEDROWS   | SGI           |

Synopsis: The type of layout of child widgets.

Values: OL\_FIXEDROWS/"fixedrows" - The layout should have a fixed number of rows  
 OL\_FIXEDCOLS/"fixedcols" - The layout should have a fixed number of columns.

The choices are to specify the number of rows or the number of columns. Only one of these dimensions can be specified directly; the other is determined by the number of child widgets added, and will always be enough to show all the child widgets.

***XtNmeasure***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCMeasure   | int         | 1              | SGI           |

Synopsis: The number of rows or columns in the layout of child widgets.

Values: 0 < XtNmeasure

If there are not enough child widgets to fill a row or column, the remaining space is left blank.

***XtNnoneSet***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCNoneSet   | Boolean     | FALSE          | SGI           |

Synopsis: Whether the buttons controlled by the Exclusives composite can be toggled into an unset mode directly.

Values: TRUE/"true" - (None Set.) At most one button in an Exclusives widget can be set. An error will be generated if an Exclusives is configured with two or more rectangular buttons set, but not if configured with no button set. The Exclusives widget will maintain this condition by ensuring that when a button is set by the user clicking SELECT over it, any button that was previously set will be cleared. Also, clicking SELECT over a button that was already set will unset it. Clearing a button in either case will invoke its XtNunselect callbacks.  
 FALSE/"false" - (One Set.) Exactly one button in an Exclusives widget is required to be set (its XtNset resource is set to TRUE).

## Exclusives Widget

An error will be generated if an Exclusives is configured with two or more rectangular buttons set or with no button set. The Exclusives widget will maintain this condition by ensuring that when a button is set by the user clicking SELECT over it, the button that was set is cleared and its `XtNunselect` callbacks are invoked. However, clicking SELECT over a button that was already set will do nothing.

### *XtNrecomputeSize*

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCRecomputeSize | Boolean | TRUE    | SGI    |

Synopsis: Whether the widget resizes itself.

Values: TRUE/"true" - The widget resizes itself to accommodate changes in its children's sizes due to changes in resources such as fonts or labels.  
 FALSE/"false" - The widget does not resize itself.

## Activation Types

The following table lists the activation types used by the Exclusives.

Table 7-17 Exclusives Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

The Exclusives widget has no activation types besides the ones in "Common Activation Types" on page 68.

*See Also*

“RectButton Widget” on page 489,  
“FlatExclusives Widget” on page 337,  
“Nonexclusives Widget” on page 428.

---

## FileChooser Widget

### FileChooser Widget

#### Class

**Class Name:** FileChooser  
**Class Pointer:** fileChooserWidgetClass

#### Ancestry

Core-Composite-Constraint-Manager-RubberTile-FileChooser

#### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/FileCh.h>
```

#### Description

The FileChooser widget allows the user to traverse directories (folders), view the files (documents) and subdirectories (folders) in them, and select files (documents).

Although an application may use a file chooser panel embedded in a multi-category property sheet, typically it would use a popup window file chooser. The OLIT FileChooserShell widget class implements such a popup window file chooser as defined in the *OPEN LOOK GUI Functional Specification* (see “FileChooserShell Widget” on page 311 for more details).

The look and feel of this widget has four variants, depending on the operation it is intended to perform: Open, Save, Save As, or Include. These variants are implemented as a single modal widget class (see “XtNoperation” on page 291).

---

**Note** – Only a single instance of a particular file chooser variant is allowed per application. It is the responsibility of the application to uphold this UI policy.

---

The API specification that follows provides for a default behavior that complies with the *OPEN LOOK GUI Functional Specification*. However, it also offers features beyond, and the flexibility to depart from, the prescriptions of the *OPEN LOOK GUI Functional Specification*.

**Note** – The *OPEN LOOK GUI Functional Specification* calls for no more than a single set of file chooser instances (Open, Save, Save As) to be created and made available to the user. It is the responsibility of the application to enforce this policy.

### Coloration

See the Coloration section of the FileChooserShell widget on page 312.

### Known Deficiencies

When the root directory (/) of the file system is browsed using the FileChooser, the first item in the scrolling list does not function properly. At that time this item is preselected (instead of the usual “Go Up One Folder” item, which is not applicable in this situation), but attempting to open the folder or file that it represents will not work. A workaround is for the user to first select another item in the list and then select the first active item.

FileChooser does not support wide character string formats. The application should use the multibyte interface as a workaround.

## Resources

Table 7-18 FileChooser Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | (parent's)          | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 0                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | (calculated)        | SGI    |

*FileChooser Widget*

*Table 7-18 FileChooser Core Resources (Continued)*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNmappedWhenManaged | Boolean        | TRUE           | SGI           |
| XtNscreen            | Screen *       | (parent's)     | G             |
| XtNsensitive         | Boolean        | TRUE           | GIO           |
| XtNtranslations      | XtTranslations | NULL           | SGI           |
| XtNwidth             | Dimension      | (calculated)   | SGI           |
| XtNx                 | Position       | 0              | SGI           |
| XtNy                 | Position       | 0              | SGI           |

*Table 7-19 FileChooser Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 7-20 FileChooser Manager Resources*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b>            | <b>Access</b> |
|----------------------|----------------|---------------------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL                      | SGIO          |
| XtNinputFocusColor   | Pixel          | (calculated; see page 27) | SGID          |
| XtNpostSelect        | XtCallbackList | NULL                      | SGIO          |
| XtNpreSelect         | XtCallbackList | NULL                      | SGIO          |
| XtNreferenceName     | String         | NULL                      | GI            |
| XtNreferenceWidget   | Widget         | NULL                      | GI            |
| XtNtraversalOn       | Boolean        | TRUE                      | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL                      | SGIO          |
| XtNuserData          | XtPointer      | NULL                      | SGI           |

*Table 7-21 FileChooser RubberTile Resources*

| <b>Name</b>    | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|----------------|-------------|----------------|---------------|
| XtNorientation | OlDefine    | OL_VERTICAL    | SGI           |



Table 7-22 FileChooser Resources

| Name                          | Type             | Default                             | Access |
|-------------------------------|------------------|-------------------------------------|--------|
| XtNapplicationFolders         | OIFolderList     | NULL                                | SGI    |
| XtNapplicationFoldersMaxCount | Cardinal         | 5                                   | GI     |
| XtNcancelAccelerator          | String           | NULL                                | SGI    |
| XtNcancelButtonWidget         | Widget           | (calculated)                        | G      |
| XtNcancelCallback             | XtCallbackList   | NULL                                | SGIO   |
| XtNcancelLabel                | OIStr            | “Cancel”                            | SGI    |
| XtNcancelMnemonic             | OIMnemonic       | ‘\0’                                | SGI    |
| XtNcommandButtonWidget        | Widget           | (calculated)                        | G      |
| XtNcomparisonFunc             | OIComparisonFunc | (string case-insensitive ascending) | SGI    |
| XtNcurrentFolder              | String           | (see <sup>1</sup> )                 | SGI    |
| XtNcurrentFolderLabelString   | OIStr            | “Current Folder:”                   | SGI    |
| XtNcurrentFolderLabelWidget   | Widget           | (calculated)                        | G      |
| XtNcurrentFolderWidget        | Widget           | (calculated)                        | G      |
| XtNdefaultDocumentName        | OIStr            | “Untitled1”                         | SGI    |
| XtNdefaultDocumentSuffix      | OIStr            | “.1”                                | SGI    |
| XtNdocumentListWidget         | Widget           | (calculated)                        | G      |
| XtNdocumentNameLabelWidget    | Widget           | (calculated)                        | G      |
| XtNdocumentNameTypeInWidget   | Widget           | (calculated)                        | G      |
| XtNexpandTilde                | Boolean          | TRUE                                | SGI    |
| XtNextensionClass             | WidgetClass      | (formWidgetClass)                   | GI     |
| XtNextensionName              | String           | NULL                                | GI     |
| XtNextensionWidget            | Widget           | NULL                                | G      |
| XtNfolderOpenedCallback       | XtCallbackList   | NULL                                | SGIO   |
| XtNfolderPromptString         | OIStr            | “Select a folder and click %.”      | SGI    |
| XtNfollowSymlinks             | Boolean          | TRUE                                | SGI    |
| XtNfont                       | OIFont           | XtDefaultFont                       | SGI    |
| XtNfontColor                  | Pixel            | XtDefaultForeground                 | SGID   |
| XtNforeground                 | Pixel            | XtDefaultForeground                 | SGID   |
| XtNgotoButtonWidget           | Widget           | (calculated)                        | G      |
| XtNgotoHomeAccelerator        | String           | NULL                                | SGI    |
| XtNgotoHomeButtonWidget       | Widget           | (calculated)                        | G      |

**FileChooser Widget**

*Table 7-22 FileChooser Resources (Continued)*

| <b>Name</b>                | <b>Type</b>    | <b>Default</b>                                     | <b>Access</b> |
|----------------------------|----------------|----------------------------------------------------|---------------|
| XtNgotoHomeLabel           | OIStr          | “Home”                                             | SGI           |
| XtNgotoHomeMnemonic        | OIMnemonic     | ‘\0’                                               | SGI           |
| XtNgotoLabel               | OIStr          | “Go To”                                            | SGI           |
| XtNgotoMenuWidget          | Widget         | (calculated)                                       | G             |
| XtNgotoPromptString        | OIStr          | “Type in the path to the folder and press Return.” | SGI           |
| XtNgotoPromptWidget        | Widget         | (calculated)                                       | G             |
| XtNgotoTypeInWidget        | Widget         | (calculated)                                       | G             |
| XtNgoUpOneFolderLabel      | OIStr          | “...Go up one folder...”                           | SGI           |
| XtNhideDotFiles            | Boolean        | TRUE                                               | SGI           |
| XtNhistoryFoldersMaxCount  | Cardinal       | 15                                                 | GI            |
| XtNhistoryFoldersMinCount  | Cardinal       | 3                                                  | GI            |
| XtNhomeFolder              | String         | (user’s UNIX home directory)                       | SGI           |
| XtNincludeAccelerator      | String         | NULL                                               | SGI           |
| XtNincludeLabel            | OIStr          | “Include”                                          | SGI           |
| XtNincludeMnemonic         | OIMnemonic     | ‘\0’                                               | SGI           |
| XtNinputDocumentCallback   | XtCallbackList | NULL                                               | SGIO          |
| XtNinputFocusColor         | Pixel          | (see page 27)                                      | SGI           |
| XtNlastDocumentName        | String         | NULL                                               | SGI           |
| XtNlistChoiceCallback      | XtCallbackList | NULL                                               | SGIO          |
| XtNlistPromptWidget        | Widget         | (calculated)                                       | G             |
| XtNlistVisibleItemCount    | Cardinal       | 10                                                 | GI            |
| XtNlistVisibleItemMinCount | Cardinal       | 3                                                  | GI            |
| XtNnoTypeInAcceleration    | Boolean        | FALSE                                              | SGI           |
| XtNopenAccelerator         | String         | NULL                                               | SGI           |
| XtNopenButtonWidget        | Widget         | (calculated)                                       | G             |
| XtNopenFolderAccelerator   | String         | NULL                                               | SGI           |
| XtNopenFolderCallback      | XtCallbackList | NULL                                               | SGIO          |
| XtNopenFolderLabel         | OIStr          | “Open Folder”                                      | SGI           |
| XtNopenFolderMnemonic      | OIMnemonic     | ‘\0’                                               | SGI           |
| XtNopenLabel               | OIStr          | “Open”                                             | SGI           |
| XtNopenMnemonic            | OIMnemonic     | ‘\0’                                               | SGI           |
| XtNopenPromptString        | OIStr          | “Select a document or folder and click %.”         | SGI           |

Table 7-22 FileChooser Resources (Continued)

| Name                        | Type           | Default             | Access |
|-----------------------------|----------------|---------------------|--------|
| XtNoperation                | OIDefine       | OL_OPEN             | GI     |
| XtNoutputDocumentCallback   | XtCallbackList | NULL                | SGIO   |
| XtNsaveAccelerator          | String         | NULL                | SGI    |
| XtNsaveAsAccelerator        | String         | NULL                | SGI    |
| XtNsaveAsLabel              | OIStr          | “Save As”           | SGI    |
| XtNsaveAsMnemonic           | OIMnemonic     | ‘\0’                | SGI    |
| XtNsaveLabel                | OIStr          | “Save”              | SGI    |
| XtNsaveMnemonic             | OIMnemonic     | ‘\0’                | SGI    |
| XtNscale                    | int            | (see <sup>2</sup> ) | SGI    |
| XtNshowGlyphs               | Boolean        | TRUE                | SGI    |
| XtNshowInactive             | Boolean        | TRUE                | SGI    |
| XtNsubstituteShellVariables | Boolean        | TRUE                | SGI    |
| XtNtextFormat               | OIStrRep       | OIDefaultTextFormat | GI     |
| XtNuserFolders              | OIFolderList   | NULL                | SGI    |
| XtNuserFoldersMaxCount      | Cardinal       | 5                   | GI     |

1. The application’s current working directory at the time the resource is initialized or set.

2. The default scale for this widget’s screen, if available; otherwise, 12 points.

The widget resources in Table 7-22 fall into three categories:

- Base Resources
  - Internationalization
  - State parameters
  - Standard callbacks (and associated callback structures)
  - Goto control
- Customization Resources
  - Sorting
  - Path name processing
  - User feedback
  - User customization: accelerators and mnemonics
- Extensibility Resources (resources provided for application programmers wishing to extend the FileChooser)
  - Extension container
  - Component access
  - Extensibility callbacks
  - Control of labels

*FileChooser Widget*

*Base Resources*

***XtNcurrentFolder***

| <b>Class</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtCFolderName | String      | (see text)     | SGI           |

Synopsis: The directory name to display, or that is being displayed, as a result of the user’s navigation with the file chooser.

Values: Relative and absolute fully specified pathname (i.e., no wild cards); “.” and “..” are accepted.

The default is to display the application’s current working directory at the time the resource is initialized or set, which for many applications is the directory from which the application was invoked.

Every time this resource is set, the file list view is updated and the XtNopenFolderCallback is called.

***XtNfollowSymlinks***

| <b>Class</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtCFollowSymlinks | Boolean     | TRUE           | SGI           |

Synopsis: The traversal of symbolic links, rather than that of actual folders, when a “Go up one folder” command is executed.

Values: TRUE/“true” - Enable traversing the parent of the symbolic link.  
 FALSE/“false” - Enable traversing the parent of the actual folder.

***XtNfont***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCFont      | OlFont      | XtDefaultFont  | SGI           |

The FileChooser widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNfont” on page 26.

***XtNfontColor***

| <b>Class</b> | <b>Type</b> | <b>Default</b>      | <b>Access</b> |
|--------------|-------------|---------------------|---------------|
| XtCFontColor | Pixel       | XtDefaultForeground | SGID          |

The FileChooser widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNfontColor” on page 27.

**XtNforeground**

| Class         | Type  | Default             | Access |
|---------------|-------|---------------------|--------|
| XtCForeground | Pixel | XtDefaultForeground | SGI    |

The FileChooser widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNforeground” on page 27.

**XtNlastDocumentName**

| Class           | Type   | Default | Access |
|-----------------|--------|---------|--------|
| XtCDocumentName | String | NULL    | SGI    |

Synopsis: The root of the base file name to be used in the “Save As” type-in field.

**XtNlistVisibleItemCount**

| Class               | Type     | Default | Access |
|---------------------|----------|---------|--------|
| XtCVisibleItemCount | Cardinal | 10      | GI     |

Synopsis: The preferred number of files and folders to show in the list.

Values: 0 < XtNlistVisibleItemCount

The list will always show at least XtNlistVisibleItemMinCount entries.

**XtNlistVisibleItemMinCount**

| Class                  | Type     | Default | Access |
|------------------------|----------|---------|--------|
| XtCVisibleItemMinCount | Cardinal | 3       | GI     |

Synopsis: The minimum value for XtNlistVisibleItemCount.

**XtNoperation**

| Class        | Type     | Default | Access |
|--------------|----------|---------|--------|
| XtCOperation | OlDefine | OL_OPEN | GI     |

Synopsis: The operation to be performed.

Values: OL\_OPEN/“open”  
 OL\_SAVE/“save”  
 OL\_SAVE\_AS/“save\_as”  
 OL\_INCLUDE/“include”

The four values correspond to the standard operations provided by the *OPEN LOOK GUI Functional Specification*.

*FileChooser Widget*

***XtNscale***

| Class    | Type | Default    | Access |
|----------|------|------------|--------|
| XtCScale | int  | (see text) | SGI    |

The FileChooser widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNscale” on page 29.

***XtNshowGlyphs***

| Class             | Type    | Default | Access |
|-------------------|---------|---------|--------|
| XtCBooleanDefault | Boolean | TRUE    | SGI    |

Synopsis: The display of glyphs in the folder content list.

Values: TRUE/"true" - Glyphs are displayed in the folder content list.  
 FALSE/"false" - Glyphs are not displayed in the folder content list.

***XtNtextFormat***

| Class         | Type     | Default             | Access |
|---------------|----------|---------------------|--------|
| XtCTextFormat | OlStrRep | OlDefaultTextFormat | GI     |

The FileChooser widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNtextFormat” on page 29.

The values of the all messages are stored in string catalogs for localization purposes, including:

- All status, including error, messages
- All state messages

***Standard Callbacks***

The generic *call\_data* structure for all FileChooser callbacks is:

```
typedef struct {
 /* OLIT standard fields */
 int reason;

 /* FileChooser standard fields */
 XtVersionType version;
 XtPointer extension;
 OlDefine operation;
 String current_folder;
} OlFileChGenericCallbackStruct;
```

Fields in the generic *call\_data* structure are:

|                       |                                                                                                                                                                                                 |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i>         | Can be any one of:<br>OL_REASON_OPEN_FOLDER<br>OL_REASON_OPEN_DOCUMENT<br>OL_REASON_OUTPUT_DOCUMENT<br>OL_REASON_FILTER<br>OL_REASON_LIST_CHOICE<br>OL_REASON_FOLDER_OPENED<br>OL_REASON_CANCEL |
| <i>version</i>        | Version of the FileChooser                                                                                                                                                                      |
| <i>extension</i>      | Reserved for future use                                                                                                                                                                         |
| <i>operation</i>      | Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE                                                                                                                                   |
| <i>current_folder</i> | The folder that the FileChooser is in currently (absolute path name)                                                                                                                            |

### ***XtNinputDocumentCallback***

| <b>Class</b> | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|--------------|----------------|----------------|---------------|
| XtCCallback  | XtCallbackList | NULL           | SGIO          |

Synopsis: The input document callback.

In an Open or Include operation, this callback list is invoked when

- A document item in the list has been selected, and the Open or Include button is depressed, or
- A document item in the list is double-clicked, or
- `XtNnoTypeInAcceleration` is set to false and a pathname expression that resolves to a unique document pathname is entered into the Goto type-in (by following it with an `OL_ENTER` action).

*FileChooser Widget*

The *call\_data* structure is:

```
typedef struct {
 /* OLIT standard fields */
 int reason;

 /* FileChooser standard fields */
 XtVersionType version;
 XtPointer extension;
 OlDefine operation;
 String current_folder;

 /* Callback-dependent fields */
 String request_document_folder;
 String request_document;
 OlFNavNode request_document_node;
} OlFileChDocumentCallbackStruct;
```

Fields in the *call\_data* structure are:

|                                |                                                                                              |
|--------------------------------|----------------------------------------------------------------------------------------------|
| <i>reason</i>                  | OL_REASON_INPUT_DOCUMENT or<br>OL_REASON_OUTPUT_DOCUMENT                                     |
| <i>version</i>                 | Version of the FileChooser                                                                   |
| <i>extension</i>               | Reserved for future use                                                                      |
| <i>operation</i>               | OL_OPEN or OL_INCLUDE                                                                        |
| <i>current_folder</i>          | The folder that the FileChooser is in currently<br>(absolute pathname)                       |
| <i>request_document_folder</i> | The folder in which the FileChooser has been<br>requested for a document (absolute pathname) |
| <i>request_document</i>        | The base document name being requested                                                       |
| <i>request_document_node</i>   | A struct of type OlFNavNode; see below.                                                      |

The OlFNavNode struct type describes the folder or document currently selected in the folder content list. Fields in the structure are:

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <i>name</i>        | The name of the file (a folder or document)                  |
| <i>is_folder</i>   | TRUE if the file is a folder                                 |
| <i>operational</i> | Not used                                                     |
| <i>filtered</i>    | Not used                                                     |
| <i>active</i>      | TRUE is the file is selected                                 |
| <i>glyph</i>       | The glyph used in the content list                           |
| <i>sbufp</i>       | A pointer to the struct stat buffer, if it has been obtained |



**XtNopenFolderCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The open folder callback.

For all operations, this callback is invoked when:

- A folder item in the list has been selected, and the Open or Open Folder button is depressed, or
- A folder item in the list is double-clicked, or
- `XtNnoTypeInAcceleration` is set to FALSE and a pathname expression that resolves to a unique folder pathname is entered into the Goto type-in (either by following it with an `OL_ENTER` action or by depressing the Goto button), or
- An item is selected in the Goto Menu, or
- The `XtNcurrentFolder` resource is set

In a Save or Save As operation, this callback is invoked when:

- `XtNnoTypeInAcceleration` is set to FALSE and a pathname expression that resolves to a unique folder pathname is entered into the Save or Save As type-in (by following it with an `OL_ENTER` action).

In all cases, the client may override the requested directory change by manipulating the *request\_folder* field of the callback structure. If the latter is set to NULL, the current directory remains unchanged. A successful change causes the value of `XtNcurrentFolder` to be updated.

The *call\_data* structure is:

```
typedef struct {
 /* OLIT standard fields */
 int reason;

 /* FileChooser standard fields */
 XtVersionType version;
 XtPointer extension;
 OlDefine operation;
 String current_folder;

 /* Callback-dependent fields */
 String request_folder;
 OlFNavNode request_folder_node;
} OlFileChFolderCallbackStruct;
```

*FileChooser Widget*

Fields in the *call\_data* structure are:

|                            |                                                                                                                                     |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i>              | OL_REASON_OPEN_FOLDER                                                                                                               |
| <i>version</i>             | Version of the FileChooser                                                                                                          |
| <i>extension</i>           | Reserved for future use                                                                                                             |
| <i>operation</i>           | Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE                                                                       |
| <i>current_folder</i>      | The folder that the FileChooser is in currently (absolute pathname)                                                                 |
| <i>request_folder</i>      | The folder that the FileChooser has been requested to move to. This field is NULL if the request is for the <i>current_folder</i> . |
| <i>request_folder_node</i> | A struct of type <code>OlFNavNode</code> ; see “XtNinputDocumentCallback” on page 293                                               |

***XtNoutputDocumentCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The output document callback.

In a Save or Save As operation, this callback list is invoked when:

- A name expression is entered into the Save or Save As type-in (by following it with an OL\_ENTER action).

The *call\_data* structure is:

```
typedef struct {
 /* OLIT standard fields */
 int reason;

 /* FileChooser standard fields */
 XtVersionType version;
 XtPointer extension;
 OlDefine operation;
 String current_folder;

 /* Callback-dependent fields */
 String request_document_folder;
 String request_document;
 OlFNavNode request_document_node;
} OlFileChDocumentCallbackStruct;
```

Fields in the *call\_data* structure are:

|                                |                                                                                            |
|--------------------------------|--------------------------------------------------------------------------------------------|
| <i>reason</i>                  | OL_REASON_OUTPUT_DOCUMENT                                                                  |
| <i>version</i>                 | Version of the FileChooser                                                                 |
| <i>extension</i>               | Reserved for future use                                                                    |
| <i>operation</i>               | OL_SAVE or OL_SAVE_AS                                                                      |
| <i>current_folder</i>          | The folder that the FileChooser is in currently (absolute pathname)                        |
| <i>request_document_folder</i> | The folder in which the FileChooser has been requested for a document (absolute pathname). |
| <i>request_document</i>        | The base document name being requested                                                     |
| <i>request_document_node</i>   | A struct of type <code>OlFNavNode</code> ; see “XtNinputDocumentCallback” on page 293      |

## File Filtering

### XtNfilterProc

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SIG    |

**Synopsis:** The callback called for each filtered (i.e., matching `XtNfilterString`) file in the list. Allows the application to specify custom glyphs for filtered files.

The *call\_data* structure is:

```
typedef struct {
 /* OLIT standard fields */
 int reason;

 /* FileChooser standard fields */
 XtVersionType version;
 XtPointer extension;
 XtPointer user_data;
 OlDefine operation;
 String current_folder;
} OlFiChFilterCallbackStruct;
```

Fields in the *call\_data* structure are:

|                |                            |
|----------------|----------------------------|
| <i>reason</i>  | OL_REASON_FILTER           |
| <i>version</i> | Version of the FileChooser |

*FileChooser Widget*

|                       |                                                                     |
|-----------------------|---------------------------------------------------------------------|
| <i>extension</i>      | Reserved for future use                                             |
| <i>operation</i>      | Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE       |
| <i>current_folder</i> | The folder that the FileChooser is in currently (absolute pathname) |

***XtNfilterString***

| Class           | Type   | Default | Access |
|-----------------|--------|---------|--------|
| XtCFilterString | String | NULL    | SGI    |

Synopsis: An *ed(1)*-like regular expression string used to filter document names from the list (except if they are directory names).

Those files that are filtered will be displayed as inactive (“grayed-out”) if the *XtNshowInactive* resource is set to TRUE, and will not be displayed at all if that resource is set to FALSE.

The expression “.” causes the *XtNfilterProc* to be called for all files in the list. No callbacks are performed if the expression is empty (“”) or NULL.

***XtNhideDotFiles***

| Class             | Type    | Default | Access |
|-------------------|---------|---------|--------|
| XtCBooleanDefault | Boolean | TRUE    | SGI    |

Synopsis: The display dot-files in the list.

Values: TRUE/“true” - Display names with leading dots.  
 FALSE/“false” - Do not display names with leading dots.

***XtNshowInactive***

| Class             | Type    | Default | Access |
|-------------------|---------|---------|--------|
| XtCBooleanDefault | Boolean | TRUE    | SGI    |

Synopsis: The display of filtered-out entries in the list.

Values: TRUE/“true” - Display filtered-out entries in the list as inactive (grayed-out).  
 FALSE/“false” - Do not display filtered-out entries.

## GoTo Control

### ***XtNapplicationFolders***

| Class      | Type         | Default | Access |
|------------|--------------|---------|--------|
| XtCFolders | OlFolderList | NULL    | SGI    |

Synopsis: The list of application specific folder names to be added to the GoTo menu.

Items in the list exceeding the value of `XtNapplicationFoldersMaxCount` are ignored. The syntax of the Intrinsic Translation Manager is used. For example:

```
"~/folder1 \n
~/folder2 \n
/folder/subfolder "
```

The datatype `OlFolderList` is defined as:

```
typedef String *OlFolderList;
```

and is expected to be terminated by a NULL string.

---

**Note** - This resource should be in sync with its equivalent in the OpenWindows WorkSpace resources, when available.

---

### ***XtNapplicationFoldersMaxCount***

| Class             | Type     | Default | Access |
|-------------------|----------|---------|--------|
| XtCFolderMaxCount | Cardinal | 5       | GI     |

Synopsis: The maximum allowed number of application folders. The actual count of these folders may not exceed this value and is determined by the value of the `XtNapplicationFolders` list.

### ***XtNhistoryFoldersMaxCount/ XtNhistoryFoldersMinCount***

| Class             | Type     | Default | Access |
|-------------------|----------|---------|--------|
| XtCFolderMaxCount | Cardinal | 15      | GI     |
| XtCFolderMinCount | Cardinal | 3       | GI     |

Synopsis: The maximum (minimum) allowed number of history folders. The actual count of these folders may not exceed (be less than) this

*FileChooser Widget*

value and is determined by the value of the `XtNhistoryFolders` list.

If the value of `XtNhistoryFoldersMaxCount` is less than that of `XtNhistoryFoldersMinCount`, the former is reset to the latter.

---

**Note** – Setting both of these resources to zero disables the folder history mechanism.

---

***XtNhomeFolder***

| Class         | Type   | Default                 | Access |
|---------------|--------|-------------------------|--------|
| XtCHomeFolder | String | (HOME directory of uid) | SGI    |

Synopsis: The pathname of a folder to be used as the HOME folder in the “Go To:” menu.

***XtNuserFolders***

| Class      | Type         | Default | Access |
|------------|--------------|---------|--------|
| XtCFolders | OlFolderList | NULL    | G      |

Synopsis: The list of user-specific folder names to be added to the GoTo menu.

Items in the list exceeding the value of `XtNuserFoldersMaxCount` are ignored. The syntax of the Intrinsic Translation Manager is used. For example:

```
~/folder1 \n
~/folder2 \n
/folder/subfolder "
```

The datatype `OlFolderList` is defined as:

```
typedef String *OlFolderList;
```

and is expected to be terminated by a NULL string.

---

**Note** – This resource should be in sync with its equivalent in the OpenWindows Workspace resources, when available.

---

***XtNuserFoldersMaxCount***

| Class             | Type     | Default | Access |
|-------------------|----------|---------|--------|
| XtCFolderMaxCount | Cardinal | 5       | GI     |

Synopsis: The maximum allowed number of user folders.

The actual count of these folders may not exceed this value and is determined by the value of the `XtNuserFolders` list.

***Customization Resources******Sorting******XtNcomparisonFunc***

| Class             | Type             | Default                             | Access |
|-------------------|------------------|-------------------------------------|--------|
| XtCComparisonFunc | OlComparisonFunc | (string case-insensitive ascending) | SGI    |

Synopsis: The string comparison function used to sort the directory entries in the list display.

`OlStrComparisonFunc()` compares its arguments and returns an integer greater than, equal to, or less than zero if *left\_string* is lexicographically greater than, equal to, or less than *right\_string*. It is defined as:

```
typedef int (*OlStrComparisonFunc)(
 const OlStr left_string,
 const OlStr right_string);
```

`OlComparisonFunc()` compares its arguments and returns an integer greater than, equal to, or less than 0, if *left\_key* is greater than, equal to, or less than *right\_key*, where the meaning or order is application-specified, as is the choice of key field within the unspecified structures of the arguments. It is defined as:

```
typedef int (*OlComparisonFunc)(
 const XtPointer left_key,
 const XtPointer right_key);
```

*FileChooser Widget*

The following sorting styles are provided by the toolkit and may be specified with these symbolic constants:

*Table 7-23 FileChooser Sorting Styles*

| <b>Sort Style</b>           | <b>OlComparisonFunc()</b>      |
|-----------------------------|--------------------------------|
| case-sensitive ascending    | OL_SORT_STR_CASE_ASCENDING     |
| case-sensitive descending   | OL_SORT_STR_CASE_DESCENDING    |
| case-insensitive ascending  | OL_SORT_STR_NO_CASE_ASCENDING  |
| case-insensitive descending | OL_SORT_STR_NO_CASE_DESCENDING |

By default, sorting of the file list is performed according to the collation sequence specified in the locale's LC\_COLLATE value.

*Pathname Processing*

***XtNexpandTilde***

| <b>Class</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtCBooleanDefault | Boolean     | TRUE           | SGI           |

Synopsis: The treatment of the tilde (~) character.

Values: TRUE/"true" - The tilde is expanded according to the `cs(1)` or `ksh(1)` rules.  
 FALSE/"false" - The tilde is treated as a literal ~ character.

***XtNsubstituteShellVariables***

| <b>Class</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtCBooleanDefault | Boolean     | TRUE           | SGI           |

Synopsis: The expansion of shell variables (such as `$OPENWINHOME`).

Values: TRUE/"true" - Shell variables are recognized and expanded to their string values.  
 FALSE/"false" - Shell variables are not expanded.



## Accelerators

***XtNcancelAccelerator /  
XtNgotoHomeAccelerator /  
XtNincludeAccelerator /  
XtNopenAccelerator /  
XtNopenFolderAccelerator /  
XtNsaveAccelerator /  
XtNsaveAsAccelerator***

| Class          | Type   | Default | Access |
|----------------|--------|---------|--------|
| XtCAccelerator | String | NULL    | SGI    |

Synopsis: The accelerators for the Cancel button, the Home item in the Go To: menu, and the Include, Open, Open Folder, Save, and Save As buttons, respectively.

See “XtNaccelerator” on page 25 for more information on accelerators.

---

**Note** – In OLIT, accelerators are restricted to 7-bit single-byte characters.

---

## Mnemonics

***XtNcancelMnemonic /  
XtNgotoHomeMnemonic /  
XtNincludeMnemonic /  
XtNopenFolderMnemonic /  
XtNopenMnemonic /  
XtNsaveAsMnemonic /  
XtNsaveMnemonic***

| Class       | Type       | Default | Access |
|-------------|------------|---------|--------|
| XtCMnemonic | OlMnemonic | “\0”    | SGI    |

Synopsis: The mnemonics for the Cancel button, the Home item in the Go To: menu, and the Include, Open, Open Folder, Save As, and Save buttons, respectively.

---

**Note** – In OLIT, mnemonics are restricted to 7-bit single-byte characters.

---

*Extensibility Resources*

*Extension Container*

***XtNextensionName***

| Class            | Type   | Default | Access |
|------------------|--------|---------|--------|
| XtCExtensionName | String | NULL    | GI     |

Synopsis: The application-specified instance name of the extension container widget instance used to parent additional controls provided by the application.

In an Open operation, this container widget extends between the file scrolling list and the command buttons at the bottom of the file chooser panel. In a Save or Save As operation, this container widget extends between the Save type-in field and the command buttons at the bottom of the file chooser panel

The container is created by the FileChooser widget on behalf of the application, if, and only if, this instance name is not NULL. This extension container is not intended to resize.

This instance name may be used for resource settings in a defaults file.

***XtNextensionClass***

| Class             | Type        | Default           | Access |
|-------------------|-------------|-------------------|--------|
| XtCExtensionClass | WidgetClass | (formWidgetClass) | G      |

Synopsis: The class name requested by the application for the extension container widget.

***XtNextensionWidget***

| Class              | Type   | Default | Access |
|--------------------|--------|---------|--------|
| XtCExtensionWidget | Widget | NULL    | G      |

Synopsis: The toolkit-provided widget ID of the extension container instance, if present.

*Component Access*

These resources are used to access a component within a FileChooser. They may be used by the client to unmanage unwanted components. An application

*FileChooser Widget*

should not assume that the returned widget ID will be of any particular class. See also `XtNextensionWidget`.

***XtNcancelButtonWidget /  
XtNcommandButtonWidget /  
XtNcurrentFolderLabelWidget /  
XtNcurrentFolderWidget /  
XtNdocumentListWidget /  
XtNdocumentNameLabelWidget /  
XtNdocumentNameTypeInWidget /  
XtNopenButtonWidget /  
XtNgotoButtonWidget /  
XtNgotoHomeButtonWidget /  
XtNgotoMenuWidget /  
XtNgotoPromptWidget /  
XtNgotoTypeInWidget /  
XtNlistPromptWidget***

| Class                           | Type   | Default      | Access |
|---------------------------------|--------|--------------|--------|
| <code>XtCComponentWidget</code> | Widget | (calculated) | G      |

Synopsis: The IDs for various widgets, as follows:

|                                          |                                                                                                                                                           |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>XtNcancelButtonWidget</code>       | The Cancel button widget                                                                                                                                  |
| <code>XtNcommandButtonWidget</code>      | The Open, Save, Save As, or Include button widget                                                                                                         |
| <code>XtNcurrentFolderLabelWidget</code> | The Current Folder: label widget                                                                                                                          |
| <code>XtNcurrentFolderWidget</code>      | The widget displaying the pathname of the current folder                                                                                                  |
| <code>XtNdocumentListWidget</code>       | The folder content scrolling list widget                                                                                                                  |
| <code>XtNdocumentNameLabelWidget</code>  | The label widget of the document name type-in field (in the Save and Save As operations)                                                                  |
| <code>XtNdocumentNameTypeInWidget</code> | The initial input focus of this widget in an Open operation; an OL_DEFAULT action for the FileChooser resolves to the verification callback of this field |
| <code>XtNopenButtonWidget</code>         | The Open button widget                                                                                                                                    |
| <code>XtNgotoButtonWidget</code>         | The Go To: button widget                                                                                                                                  |
| <code>XtNgotoHomeButtonWidget</code>     | The Home button widget in the Go To: menu                                                                                                                 |
| <code>XtNgotoMenuWidget</code>           | The Go To: menu widget                                                                                                                                    |

*FileChooser Widget*

|                     |                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNgotoPromptWidget | The prompt widget for Go To: function                                                                                                                               |
| XtNgotoTypeInWidget | The initial input focus of this widget in a Save or Save As operation; an OL_DEFAULT action for the FileChooser resolves to the verification callback of this field |
| XtNlistPromptWidget | The widget prompting for the folder content list                                                                                                                    |

***Extensibility Callbacks***

These resources are provided to augment the standard internal callbacks for the benefit of applications that need to extend the standard file chooser behavior.

***XtNcancelCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: In all operations, a callback list invoked when the Cancel button is depressed.

The *call\_data* structure is a *OlFileChGenericCallbackStruct*, as shown in “Standard Callbacks” on page 292. Fields in the generic *call\_data* structure are:

|                       |                                                                     |
|-----------------------|---------------------------------------------------------------------|
| <i>reason</i>         | OL_REASON_CANCEL                                                    |
| <i>version</i>        | Version of the FileChooser                                          |
| <i>extension</i>      | Reserved for future use                                             |
| <i>operation</i>      | Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE       |
| <i>current_folder</i> | The folder that the FileChooser is in currently (absolute pathname) |

***XtNfolderOpenedCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback invoked after a directory change has been performed.

The *call\_data* structure is a *OlFileChGenericCallbackStruct*, as shown in “Standard Callbacks” on page 292. Fields in the generic *call\_data* structure are:

|               |                         |
|---------------|-------------------------|
| <i>reason</i> | OL_REASON_FOLDER_OPENED |
|---------------|-------------------------|

|                       |                                                                     |
|-----------------------|---------------------------------------------------------------------|
| <i>version</i>        | Version of the FileChooser                                          |
| <i>extension</i>      | Reserved for future use                                             |
| <i>operation</i>      | Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE       |
| <i>current_folder</i> | The folder that the FileChooser is in currently (absolute pathname) |

**XtNlistChoiceCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback invoked in addition to the standard callback when an entry is selected in the list.

The *call\_data* structure is:

```
typedef struct {
 /* OLIT standard fields */
 int reason;

 /* FileChooser standard fields */
 XtVersionType version;
 XtPointer extension;
 OlDefine operation;
 String current_folder;

 /* Callback-dependent fields */
 String chosen_item;
 OlFNavNode chosen_item_node;
} OlFileChListChoiceCallbackStruct;
```

Fields in the *call\_data* structure are:

|                         |                                                                         |
|-------------------------|-------------------------------------------------------------------------|
| <i>reason</i>           | OL_REASON_LIST_CHOICE                                                   |
| <i>version</i>          | Version of the FileChooser                                              |
| <i>extension</i>        | Reserved for future use                                                 |
| <i>operation</i>        | Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE           |
| <i>current_folder</i>   | The folder that the FileChooser is in currently (absolute pathname)     |
| <i>chosen_item</i>      | Label of the item selected in the list                                  |
| <i>chosen_item_node</i> | A struct of type OlFNavNode; see “XtNinputDocumentCallback” on page 293 |

**Labels**

The default values for the following resources pertaining to labels are stored in string catalogs for localization purposes. These resources allow applications to override the default labels prescribed in the functional specification for the four standard operations (Open, Save, Save As, and Include). These resources will be modified to accomplish other operations not directly supported, for example: Print, Browse, Compile, etc.

***XtNgotoPromptString /  
XtNopenPromptString /  
XtNfolderPromptString***

| Class           | Type  | Default                                            | Access |
|-----------------|-------|----------------------------------------------------|--------|
| XtCPromptString | OlStr | "Type in the path to the folder and press Return." | SGI    |
| XtCPromptString | OlStr | "Select a document or folder and click %."         | SGI    |
| XtCPromptString | OlStr | "Select a folder and click %."                     | SGI    |

Synopsis: The prompt string used in the "Go To:" function, the OL\_OPEN operation, and the OL\_SAVE/OL\_SAVE\_AS operations, respectively.

Values: Any OlStr value valid in the current locale.

'%' in the string is replaced with the label of the appropriate button. For example, in an OL\_OPEN operation the *default* label of the save button is "Open", and is the string used in deriving the list prompt. But, if the label had been customized to "Browse", the latter string would be used. Hence, in the examples above the list prompt would be

"Select a document or folder and click Open."

"Select a document or folder and click Browse."

***XtNgotoLabel /***  
***XtNgotoHomeLabel /***  
***XtNgoUpOneFolderLabel /***  
***XtNopenFolderLabel /***  
***XtNcancelLabel /***  
***XtNcurrentFolderLabelString /***  
***XtNopenLabel /***  
***XtNsaveLabel /***  
***XtNsaveAsLabel /***  
***XtNincludeLabel***

| Class    | Type  | Default               | Access |
|----------|-------|-----------------------|--------|
| XtCLabel | OlStr | “Go To”               | SGI    |
| XtCLabel | OlStr | “Home”                | SGI    |
| XtCLabel | OlStr | ...Go up one folder.. | SGI    |
| XtCLabel | OlStr | “Open Folder”         | SGI    |
| XtCLabel | OlStr | “Cancel”              | SGI    |
| XtCLabel | OlStr | “Current Folder:”     | SGI    |
| XtCLabel | OlStr | “Open”                | SGI    |
| XtCLabel | OlStr | “Save”                | SGI    |
| XtCLabel | OlStr | “Save As”             | SGI    |
| XtCLabel | OlStr | “Include”             | SGI    |

Synopsis: The strings used as prompts for various operations: the Go To: button; the Home button in the Go To: menu; the “Go up one folder” item (the first item) in the folder content list; the Open Folder button; the Cancel button; the string in the current folder item; and the Open, Save, Save As, and Include buttons, respectively.

Values: Any `OlStr` value valid in the current locale.

### ***XtNdefaultDocumentName***

| Class                  | Type  | Default     | Access |
|------------------------|-------|-------------|--------|
| XtCDefaultDocumentName | OlStr | “Untitled1” | SGI    |

Synopsis: In a Save operation, the root of the base name of the default document name in the Save type-in field.

Values: Any `OlStr` value valid in the current locale.

*FileChooser Widget*

***XtNdefaultDocumentSuffix***

| <b>Class</b>             | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------------------|-------------|----------------|---------------|
| XtCDefaultDocumentSuffix | OlStr       | “.1”           | SGI           |

Synopsis: In a Save As operation, the suffix to be used with the default document name in the Save As type-in field.

Values: Any OlStr value valid in the current locale.

***Activation Types***

The following table lists the activation types used by the FileChooser.

*Table 7-24 FileChooser Activation Types*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The FileChooser widget has no activation types besides the ones in “Common Activation Types” on page 68.

***See Also***

“FileChooserShell Widget” on page 311.



## FileChooserShell Widget

### Class

**Class Name:** FileChooserShell  
**Class Pointer:** fileChooserShellWidgetClass

### Ancestry

Core-Composite-Shell-WMShell-VendorShell-TransientShell-FileChooserShell

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/FileChSh.h>
```

### Description

The FileChooserShell widget class implements the OPEN LOOK file chooser GUI object. The core of the functionality is encapsulated in the FileChooser widget, which is instantiated by the FileChooserShell (see “FileChooser Widget” on page 284 for more details).

As called for by the *OPEN LOOK GUI Functional Specification*, the FileChooserShell derives its header from the title of the application and the name of the file chooser operation it is performing. The title of the application is obtained from the main top level window of the application. This string may either have been provided by the application (e.g., “Foo Text Editor”), or, it may be the name of the executable used to invoke the application (e.g., “footextedit”). The name of the operation is taken from the string used to label the button used to perform that operation in the file chooser panel. For example, in a OL\_SAVE operation the *default* label of the save button is “Save”, and is the string used in deriving the FileChooserShell header. But, if the label had been customized to “Write”, the latter string would be used.

Therefore, in the examples above the header displayed would be one of:

```
“Foo Text Editor: Save”
“footextedit: Save”
“Foo Text Editor: Write”
“footextedit: Write”
```

Coloration

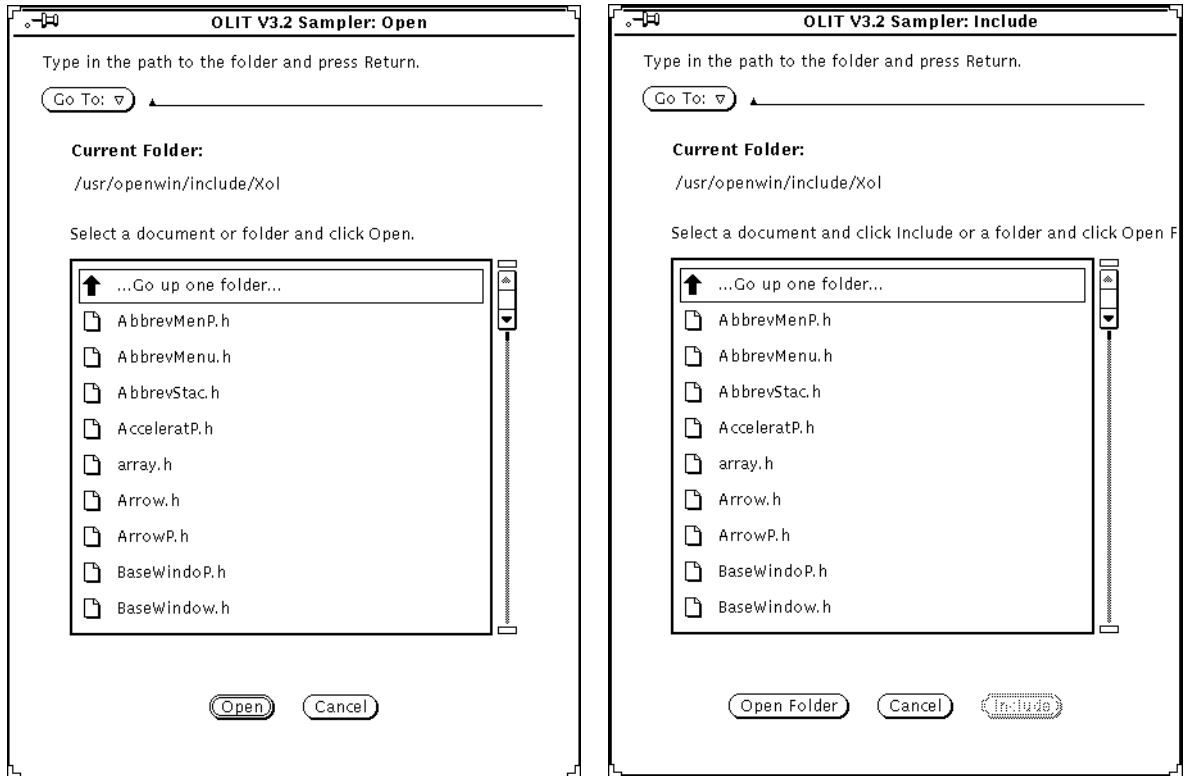


Figure 7-4 FileChooserShell Appearance (Open, Include Operations)

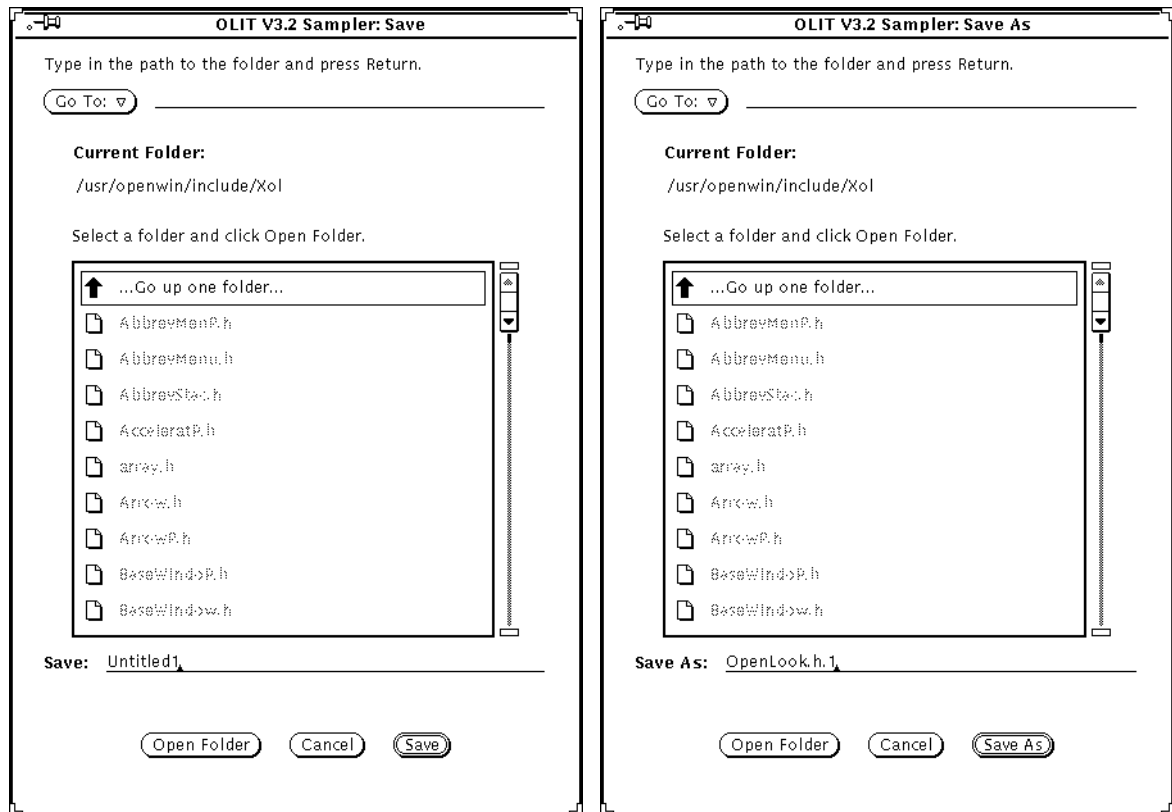


Figure 7-5 FileChooserShell Appearance (Save, Save As Operations)

*FileChooserShell* Widget

*Resources*

*Table 7-25* FileChooserShell Core Resources

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>                        | <b>Access</b> |
|----------------------|------------------|---------------------------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                                  | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                                  | G             |
| XtNbackground        | Pixel            | XtDefaultBackground                   | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap                   | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground                   | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap                   | SGI           |
| XtNborderWidth       | Dimension        | 0                                     | SGI           |
| XtNcolormap          | Colormap         | (see description)                     | GI            |
| XtNdepth             | int              | (parent's or default depth of screen) | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                                  | SGIO          |
| XtNheight            | Dimension        | (calculated)                          | SG            |
| XtNmappedWhenManaged | Boolean          | TRUE                                  | SGI           |
| XtNscreen            | Screen *         | (parent's)                            | GI            |
| XtNsensitive         | Boolean          | TRUE                                  | GIO           |
| XtNtranslations      | XtTranslations   | NULL                                  | SGI           |
| XtNwidth             | Dimension        | (calculated)                          | SG            |
| XtNx                 | Position         | 0                                     | SGI           |
| XtNy                 | Position         | 0                                     | SGI           |

*Table 7-26* FileChooserShell Composite Resources

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 7-27* FileChooserShell Shell Resources

| <b>Name</b>         | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------------|-------------|----------------|---------------|
| XtNallowShellResize | Boolean     | TRUE           | SGI           |

*FileChooserShell Widget**Table 7-27 FileChooserShell Shell Resources (Continued)*

| <b>Name</b>             | <b>Type</b>            | <b>Default</b>        | <b>Access</b> |
|-------------------------|------------------------|-----------------------|---------------|
| XtNcreatePopupChildProc | XtCreatePopupChildProc | NULL                  | SGI           |
| XtNgeometry             | String                 | NULL                  | GI            |
| XtNoverrideRedirect     | Boolean                | FALSE                 | SGI           |
| XtNpopdownCallback      | XtCallbackList         | NULL                  | SGIO          |
| XtNpopupCallback        | XtCallbackList         | NULL                  | SGIO          |
| XtNsaveUnder            | Boolean                | FALSE                 | SGI           |
| XtNvisual               | Visual *               | (parent's)            | GIO           |
| XtNwidthInc             | int                    | XtUnspecifiedShellInt | SGI           |

*Table 7-28 FileChooserShell WMSHELL Resources*

| <b>Name</b>      | <b>Type</b>  | <b>Default</b>        | <b>Access</b> |
|------------------|--------------|-----------------------|---------------|
| XtNbaseHeight    | int          | XtUnspecifiedShellInt | GI            |
| XtNbaseWidth     | int          | XtUnspecifiedShellInt | GI            |
| XtNheightInc     | int          | XtUnspecifiedShellInt | GI            |
| XtNiconMask      | Pixmap       | NULL                  | SGI           |
| XtNiconPixmap    | Pixmap       | NULL                  | SGI           |
| XtNiconWindow    | Window       | NULL                  | SGI           |
| XtNiconX         | int          | XtUnspecifiedShellInt | GI            |
| XtNiconY         | int          | XtUnspecifiedShellInt | GI            |
| XtNinitialState  | InitialState | NormalState           | GI            |
| XtNinput         | Bool         | FALSE                 | G             |
| XtNmaxAspectX    | int          | XtUnspecifiedShellInt | GI            |
| XtNmaxAspectY    | int          | XtUnspecifiedShellInt | GI            |
| XtNmaxHeight     | int          | XtUnspecifiedShellInt | GI            |
| XtNmaxWidth      | int          | XtUnspecifiedShellInt | GI            |
| XtNminAspectX    | int          | XtUnspecifiedShellInt | GI            |
| XtNminAspectY    | int          | XtUnspecifiedShellInt | GI            |
| XtNminHeight     | int          | 315                   | GI            |
| XtNminWidth      | int          | 197                   | GI            |
| XtNtitle         | String       | (see <sup>1</sup> )   | SGI           |
| XtNtitleEncoding | Atom         | (see <sup>2</sup> )   | SGI           |
| XtNtransient     | Boolean      | (see <sup>3</sup> )   | SGI           |
| XtNwaitForWm     | Boolean      | TRUE                  | GI            |
| XtNwidthInc      | int          | XtUnspecifiedShellInt | GI            |

*FileChooserShell Widget*

*Table 7-28 FileChooserShell WMShell Resources (Continued)*

| <b>Name</b>    | <b>Type</b> | <b>Default</b>        | <b>Access</b> |
|----------------|-------------|-----------------------|---------------|
| XtNwindowGroup | Window      | XtUnspecifiedWindow   | GI            |
| XtNwinGravity  | int         | XtUnspecifiedShellInt | GI            |
| XtNwmTimeout   | int         | (see <sup>4</sup> )   | GI            |

1. The application's icon name, if specified; otherwise, the application's name.
2. XA\_STRING if the language procedure is not NULL; otherwise, None.
3. TRUE for a TransientShell; otherwise, FALSE.
4. DEFAULT\_WM\_TIMEOUT, if specified; otherwise, 5 seconds.

*Table 7-29 FileChooserShell VendorShell Resources*

| <b>Name</b>             | <b>Type</b>    | <b>Default</b>                              | <b>Access</b> |
|-------------------------|----------------|---------------------------------------------|---------------|
| XtNbusy                 | Boolean        | FALSE                                       | SGI           |
| XtNdefaultImName        | String         | NULL                                        | SGI           |
| XtNfooterPresent        | Boolean        | TRUE                                        | SGI           |
| XtNfocusWidget          | Widget         | (see <sup>1</sup> )                         | SGI           |
| XtNimFontSet            | OIFont         | XtDefaultFontSet                            | SGI           |
| XtNimStatusStyle        | OImStatusStyle | OL_NO_STATUS                                | GI            |
| XtNleftFooterString     | OIStr          | NULL                                        | SGI           |
| XtNleftFooterVisible    | Boolean        | TRUE                                        | SGI           |
| XtNmenuButton           | Boolean        | FALSE                                       | GI            |
| XtNmenuType             | OIDefine       | OL_MENU_LIMITED                             | SGI           |
| XtNpushpin              | OIDefine       | OL_OUT                                      | SGI           |
| XtNresizeCorners        | Boolean        | TRUE                                        | SGI           |
| XtNrightFooterString    | OIStr          | NULL                                        | SGI           |
| XtNrightFooterVisible   | Boolean        | TRUE                                        | SGI           |
| XtNshellTitle           | OIStr          | (see Description)                           | SGI           |
| XtNtextFormat           | OIStrRep       | (default text format)                       | SGI           |
| XtNuserData             | XtPointer      | NULL                                        | SGI           |
| XtNwindowHeader         | Boolean        | TRUE                                        | GI            |
| XtNwinType              | OIDefine       | OL_WT_CMD                                   | GI            |
| XtNwmProtocolInterested | int            | OL_WM_DELETE_WINDOW I<br>  OL_WM_TAKE_FOCUS |               |

1. The default is the "Go To:" type-in field, if the operation is OL\_OPEN or OL\_INCLUDE; the "Save:"/"Save As" type-in field, if the operation is OL\_SAVE/OL\_SAVE\_AS, respectively.

Table 7-30 FileChooserShell TransientShell Resources

| Name            | Type   | Default | Access |
|-----------------|--------|---------|--------|
| XtNtransientFor | Widget | NULL    | SGI    |

Table 7-31 FileChooserShell Resources

| Name                 | Type           | Default               | Access |
|----------------------|----------------|-----------------------|--------|
| XtNfileChooserWidget | Widget         | (calculated)          | G      |
| XtNoperation         | OlDefine       | OL_OPEN               | GI     |
| XtNpointerWarping    | Boolean        | TRUE                  | SGI    |
| XtNtextFormat        | OlStrRep       | (default text format) | GI     |
| XtNverifyCallback    | XtCallbackList | NULL                  | SGIO   |

**XtNfileChooserWidget**

| Class              | Type   | Default      | Access |
|--------------------|--------|--------------|--------|
| XtCComponentWidget | Widget | (calculated) | G      |

Synopsis: The FileChooser child widget that can be accessed for setting or getting its resources; see “FileChooser Widget” on page 284 for its resources.

**XtNoperation**

| Class        | Type     | Default | Access |
|--------------|----------|---------|--------|
| XtCOperation | OlDefine | OL_OPEN | GI     |

Synopsis: The operation to be performed.

Values: OL\_OPEN/“open”  
 OL\_SAVE/“save”  
 OL\_SAVE\_AS/“save\_as”  
 OL\_INCLUDE/“include”

The four values correspond to the standard operations provided by the *OPEN LOOK GUI Functional Specification*.

*FileChooserShell* Widget

***XtNpointerWarping***

| Class             | Type    | Default | Access |
|-------------------|---------|---------|--------|
| XtCPointerWarping | Boolean | TRUE    | SGI    |

Synopsis: The pointer warping of the file chooser upon popup.

Values: TRUE/"true" - The pointer is warped.  
 FALSE/"false" - The pointer is not warped.

***XtNtextFormat***

| Class         | Type     | Default               | Access |
|---------------|----------|-----------------------|--------|
| XtCTextFormat | OlStrRep | (default text format) | GI     |

Synopsis: The expected data format of all the textual resources of a widget.

Values: OL\_SB\_STR\_REP - Single-byte character representation.  
 OL\_WC\_STR\_REP - Wide character representation.  
 OL\_MB\_STR\_REP - Multibyte character representation.

See "XtNtextFormat" on page 29 for details of initialization and the default value.

---

**Note** - Wide character string representation, OL\_WC\_STR\_REP, is not supported in this release.

---

The values of the all messages are stored in string catalogs for localization purposes, including:

- All status, including error, messages
- All state messages

***XtNverifyCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list to be invoked when popping down a FileChooserShell widget. It allows interposing regardless of the cause of the popdown action (Open, Save, Save As, Cancel, Include, pin-out, dismiss).

Setting the *accept\_verify* field to FALSE will defeat the popdown.



The *call\_data* structure is:

```
typedef struct {
 /* OLIT standard fields */
 int reason;

 /* FileChooser standard fields */
 XtPointer extension;
 OlDefine operation;
 String current_folder;

 /* Callback-dependent fields */
 Boolean accept_verify;
} OlFileChShVerifyCallbackStruct;
```

Fields in the *call\_data* structure are:

|                       |                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------|
| <i>reason</i>         | OL_REASON_VERIFY                                                                     |
| <i>extension</i>      | Reserved for future use                                                              |
| <i>operation</i>      | Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE                        |
| <i>current_folder</i> | The folder that the FileChooser is in (absolute pathname)                            |
| <i>accept_verify</i>  | Set to TRUE by default in the <i>call_data</i> . To reject, set this field to FALSE. |

## Activation Types

The following table lists the activation types used by the FileChooserShell.

Table 7-32 FileChooserShell Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

### *FileChooserShell Widget*

The FileChooserShell widget has no activation types besides the ones in “Common Activation Types” on page 68.

### *See Also*

“FileChooser Widget” on page 284.

## Flat Widgets

Flattened, or flat, widgets give the visual appearance and functionality of many discrete windowed widgets, but are implemented as one widget with a single associated window, created with one convenient toolkit request. Flat widgets consume a fraction of the memory that a similar widget hierarchy requires. The following widgets have flat versions: CheckBox, Exclusives, and Nonexclusives.

In general, flat widgets have the following attributes:

- They are container widgets, responsible for managing the layout of one or more controls, which are called *items* in this manual to distinguish them from real widgets.
- The item classes are limited to one or a select few.
- Typically, after the container is populated, minimal or no manipulation of the items is desired.
- Each container is simply a region that contains zero or more items of a certain type.
- The items within the container do not have an associated window or widget structure.

From the user's perspective, there's no distinguishable difference between a flattened widget interface and a traditionally composed interface. From the programmer's perspective, flattened widgets have a different interface than traditional widgets or gadgets. A single toolkit request can specify an arbitrary number of primitive graphical user interface components (i.e., the items), thus achieving a substantial reduction in the lines of code required to produce a complex graphical interface component.

### *Item Lists and Allowable Resources*

Items of a flat container are specified in list format. For the life of the list, both the flat container and the application share the same list. Each list is an array of application-defined records (typically, in a C-language structure format or as an array), where each record describes a particular item.

For efficiency reasons, each record in the array must have the same form as the other records in the array; i.e., each structure in the list has identical fields. This

restriction applies on a per-list basis only, since each list may have a different set of fields per record if the application desires different attributes.

For example, if an application wanted to specify an “unselect” callback procedure for one group of exclusives, but not for another, the application would specify an `XtNunselectProc` field as an element field for the first list, but not for the second list. For data alignment and parsing reasons, the fields of each record must use the `XtArgVal` type (see “*FlatExclusives Settings Example*”).

The fields of each record are resource values that describe the state of the item. Allowable item resources are a subset of its container’s resource set. The genealogy of flattened widget resources is derived from release 1.0 resources, although new resources have been added (and some old ones removed) to provide greater convenience to the application programmer and achieve a higher degree of consistency between all flat containers and their items. An item inherits any non-specified resource from its container. For example, if the application wanted a particular font color for all items, the application does not have to specify the `XtNfontColor` resource for each item; the application simply sets the font color resource on the container and all items will use that font color. Although item resources are part of its container’s resource set, none of the item resources have any direct effect on the container.

Since the “form” of the item record is defined within the application’s domain, the container must be given a hint about the record’s form so that it can parse the supplied list. A resource name list is the key to unlocking the application’s item list.

While the ordering of fields in each record is not important, the application must give the resource names in the same sequence that their associated values appear in the record. For example, if the records specifying items of a flat exclusives container had a “label” field followed by the “selectProc” callback field, the application must supply the container with the `XtNlabel` resource name followed by the `XtNselectProc` resource name. Inconsistent ordering of the fields will result in undefined behavior when the items are instantiated.

### *FlatExclusives Settings Example*

The following code example illustrates how to create a `FlatExclusives` settings. Notice that all the fields in the application-defined structure,

FlatExclusives, have the type XtArgVal. An alternative form for specifying the FlatExclusives type is:

```
typedef XtArgVal FlatExclusives[number];
```

where *number* is the number of fields per record.

XtNclientData resources are specified for the container only, which allows each item to inherit this value. If each item wanted a different client data, the XtNclientData resource should be added to the other item resources, which would disable the inheriting of the container's client data value. To improve the readability of this example, required type casts of the fields in the FlatExclusives structure initialization deliberately have been omitted. The resources used in the example are described in the individual flat widget sections.

```
typedef struct { /* Application Defined Structure */
 XtArgVal label; /* pointer to a string */
} FlatExclusives;
String exc_fields[] = { XtNlabel };
static void cb()
{ /* something interesting in here */ }
CreateObjects(parent)
 Widget parent;
{
 Arg args[6];
 static FlatExclusives exc_items[] = {
 { "Choice 1" },
 { "Choice 2" },
 { "Choice 3" }
 };
 XtSetArg(args[0], XtNitems, exc_items);
 XtSetArg(args[1], XtNnumItems, XtNumber(exc_items));
 XtSetArg(args[2], XtNitemFields, exc_fields);
 XtSetArg(args[3], XtNnumItemFields, XtNumber(exc_fields));
 XtSetArg(args[4], XtNselectProc, cb);
 XtSetArg(args[5], XtNclientData, "test case");
 XtCreateManagedWidget("exclusives", flatExclusivesWidgetClass,
 parent, args, 6);
} /* End Of CreateObjects() */
```

### Specifying Items

Items of flattened widget containers are specified in list format with each list having a corresponding set of resource names describing how the container is to parse the list. Four common resources are used by each container class to describe the necessary item information:

| Name             | Class            | Type      | Access |
|------------------|------------------|-----------|--------|
| XtNItems         | XtCItems         | XtPointer | SGI    |
| XtNnumItems      | XtCNumItems      | Cardinal  | SGI    |
| XtNitemFields    | XtCItemFields    | String *  | SGI    |
| XtNnumItemFields | XtCNumItemFields | Cardinal  | SGI    |

| Resource         | Use                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNItems         | The list of items. This list must not be in a temporary memory location such as a stack since flat containers reference it after initialization and do not copy it in their private storage.                                                                                                                                                                                                                                                                                                                                                                                       |
| XtNnumItems      | The number of items.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| XtNitemFields    | The list of resource names used to parse the records in the XtNItems list. XtNitemFields does not have to point to static information since the flat container does not use this information after initialization. Although the flat container does not reference this resource's value after initialization, it holds onto it for responding to an XtGetValues() request and supplying it in the OlFlatCallData structure during callbacks. Therefore, if the application plans on querying this resource, the application should make this resource point to static information. |
| XtNnumItemFields | The number of resource names contained in XtNitemFields.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## Callbacks and Flat Widgets

There are two differences in the way callbacks are handled for flat widgets versus traditional widgets. The first difference is that items do not use `XtCallbackLists`; instead, they use a single `XtCallbackProc` procedure.

A flat widget callback procedure has the following form:

```
typedef void (*XtCallbackProc)(
 Widget widget,
 XtPointer client_data,
 XtPointer call_data);
```

Since the items of flattened widget containers are not true widget instances, the *widget* argument supplied to an application's callback procedure indicates the flat container widget that is ultimately responsible for managing the items. For example, the `flatExclusivesWidget` ID would be supplied as the widget ID to the callback procedure for all items within the flat exclusives container. By maintaining this rule, the application always has the correct widget handy in the event that the application wishes to modify the item or its list from within the callback procedure. The value of the `XtNclientData` resource is supplied as the *client\_data* field to the callback procedure. The *call\_data* field is a structure that the application can use to determine information about the item associated with the current callback:

```
typedef struct {
 Cardinal item_index;
 XtPointer items;
 Cardinal num_items;
 String *item_fields;
 Cardinal num_item_fields;
} OlFlatCallData;
```

The fields are:

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| <i>item_index</i>      | Index of the item responsible for the callback.                          |
| <i>items</i>           | The head of item list that contains the item initiating the callback     |
| <i>num_items</i>       | The total number of items in the item list                               |
| <i>item_fields</i>     | The list of resource name used to identify the records in the item list. |
| <i>num_item_fields</i> | The number of resource names contained in <i>item_fields</i> .           |

### Setting/Getting the State of an Item

The application can use two methods to change the state of an item: use the `OlFlatSetValues()` procedure (see page 356) to modify one or more attributes of an item, or directly modify the item list that the container and the application share.

The first approach is very similar to doing an `XtSetValues()` request on a widget, except that the `OlFlatSetValues()` routine requires the item index as well as the widget ID, *args*, and *num\_args*. The following code example illustrates how to change an item's label from within a callback procedure. The example assumes the new label was specified as the client data.

```

Callback(
 Widget widget, /* FlatExclusives Widget ID */
 caddr_t client_data, /* the new static label */
 caddr_t call_data); /* OlFlatCallData struct pointer */
{
 OlFlatCallData *fcd = (OlFlatCallData *)call_data;
 Arg args[1];

 /* Set the label to be the new one passed in
 * with the client data field. */
 XtSetArg(args[0], XtNlabel, client_data);

 OlFlatSetValues(widget, fcd->item_index, args, 1);
} /* End Of Callback() */

```

Notice that the callback procedure did not have to know the number or the order of the item fields. The only requirement was that the `XtNlabel` resource is among the application-specified item fields, because if it was not, the above request would be ignored.

There are some exceptions to this rule. For instance, the flat containers maintain the set item and the default item even if the application did not specify `XtNset` or `XtNdefault`. Having exceptions in this case are worthwhile, since if the exceptions were not made, the application always would have to specify a minimum set of item fields, which would have been an undesirable requirement. See the individual widget sections in this manual for a better description of the exceptions.

If the application does not use the above approach and modifies the item list directly, the application must ensure that all items within the list have valid states, since the container literally treats this type of modification as if the container was given a new list.



*Flat Widgets*

For example, if an application wished to set a new item in a list of exclusive items, it should first unset the currently set item and then set the new item. If the application only set the new item, the container would generate a warning since the item list contains more than one set item.

The following example shows how a callback procedure changes the set item by modifying the item list. This example makes the first item be the set item whenever the last item is selected. Notice that once the list has been touched, the application must “inform” the container of the modification. Also notice that in this example the application needs to know the *structure* of the application to directly change its contents.

```

 /* Application-defined structure from previous example */
typedef struct {
 XtArgVal label; /* pointer to a string */
 XtArgVal select_proc; /* pointer to a callback procedure */
 XtArgVal set; /* this item is currently set */
 XtArgVal sensitive; /* this item is sensitive */
} FlatExclusives;

Callback(
 Widget widget; /* FlatExclusives Widget ID */
 XtPointer client_data; /* application's client data */
 XtPointer call_data; /* OlFlatCallData struct pointer */
{
 OlFlatCallData *fcd = (OlFlatCallData *) call_data;
 if (fcd->num_items == (fcd->item_index + 1))
 {
 FlatExclusives *fexc_items = (FlatExclusives *) fcd->items;
 Arg args[1];

 /* Unset this item and set the first one */
 fexc_items[fcd->item_index].set = FALSE;
 fexc_items[0].set = TRUE;

 /* Inform the container that the list was modified */
 XtSetArg(args[0], XtNitemsTouched, TRUE);
 XtSetValues(widget, args, 1);
 }
} /* End of Callback() */

```

Obtaining the state of an item can be achieved in two ways:

1. Using the `OlFlatGetValues()` routine, specify the index of the item to be queried (see page 355). If this approach is used, the application can query any item resource even though it does not appear in the item fields.

In the initial example, for instance, the application can query the `XtNfontColor` resource from any item even though it does not appear in the `FlatExclusives` structure.

2. Looking directly into the item list, since both the application and flat container share the same instance of the item description.

### *Registering Help on Items*

The application can specify a unique help message for each item in a similar fashion as help is registered for widgets; i.e., through the `OlRegisterHelp()` routine (see page 146). Since items are not real widgets, but are extensions of the flat widget container, the help registration routine has a complex ID:

```
typedef struct {
 Widget widget; /* Flat Widget ID */
 Cardinal item_index; /* item to register help on */
} OlFlatHelpId;
```

The following example registers help on item number 8.

```
static String tag = "Item 8";
static String source = "Item 8's help";
OlFlatHelpId help_id;
help_id.widget = flat_widget;
help_id.item_index = 8;

OlRegisterHelp((XtPointer) &help_id, OL_FLAT_HELP, tag,
 OL_STRING_SOURCE, source);
```

### *See Also*

- “FlatCheckBox Widget” on page 329,
- “FlatExclusives Widget” on page 337,
- “FlatNonexclusives Widget” on page 347,
- “Flat Widget Functions” on page 354,
- “Help Function” on page 146.

## FlatCheckBox Widget

### Class

**Class Name:** FlatCheckBox  
**Class Pointer:** flatCheckBoxWidgetClass

### Ancestry

Core-Primitive-Flat-FlatExclusives-FlatCheckBox

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/FCheckBox.h>
```

### Description

The FlatCheckBox enables the application to create a number of check boxes with a single widget. It provides *items* that appear and behave the same as CheckBox widgets. However, the items only exist within the context of the FlatCheckBox widget and are themselves not real widgets. Attributes of FlatCheckBox items can be read and written using the Flat Widget Functions described on page 354. They cannot be read or written using the Xt Intrinsic functions.

In short, the FlatCheckBox provides the same functionality as a Nonexclusives widget populated with CheckBox widgets; however, it offers improved performance since fewer widgets are created.

See the general explanation of flattened widgets in “Flat Widgets” on page 321.

## FlatCheckBox Widget

### Components

The following diagram illustrates the components of the FlatCheckBox:

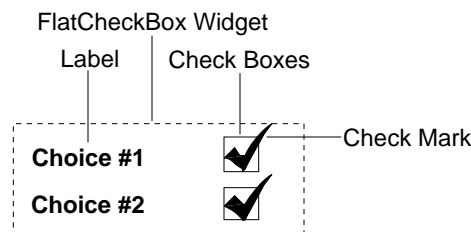


Figure 7-6 FlatCheckBox Components

### Coloration

The background of the FlatCheckBox container is drawn using the parent's `XtNbackground` resource. The labels in the items are drawn using `XtNfontColor`. The checkmarks in the items are drawn using `XtNforeground`.

For 3D, the check box component coloration of each item is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNforeground` is used to render the outline of the check box component for each item as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

If the toolkit resource `XtNmouseless` is set to TRUE and the toolkit resource `XtNinputFocusFeedback` is set to `OL_INPUT_FOCUS_COLOR`, when the FlatCheckBox receives input focus, the background of the check box component of the item with focus will be drawn with the value of `XtNinputFocusColor`. However, if `XtNinputFocusColor` is the same as `XtNbackground`, then the widget inverts `XtNforeground` and `XtNbackground` inside the check box component of the item with focus. Once the input focus leaves the widget, the original coloration is restored.

### ***Keyboard Traversal***

The FlatCheckBox widget is a Primitive widget that manages traversal between the check box items. When the user traverses to a FlatCheckBox widget, the first item in the set will display itself as having input focus (see “CheckBox Widget” on page 237 for a description of this appearance.)

The MOVEUP, MOVEDOWN, MOVERIGHT, and MOVELEFT keys move the input focus between the items. To traverse out of the FlatCheckBox widget, the following keys can be used:

- NEXTFIELD moves to the next traversable widget in the window
- PREVFIELD moves to the previous traversable widget in the window
- NEXTWINDOW moves to the next window in the application
- PREVWINDOW moves to the previous window in the application
- NEXTAPP moves to the first window in the next application
- PREVAPP moves to the first window in the previous application

### ***Keyboard Mnemonic Display***

The FlatCheckBox widget displays the mnemonic accelerator of an item as part of the item’s label. The display attributes of the mnemonic character are specified by the toolkit resource `XtNshowMnemonics` (see page 16). If a character is to be displayed, and the character is present in the label, the character in the label is emphasized in accordance with this resource. If the character to be displayed is not in the label, the character is displayed to the right of the label, on the same baseline and in parentheses. The emphasis in the latter case is again in accordance with the `XtNshowMnemonics` resource.

If truncation is necessary, the mnemonic displayed in parentheses is truncated as a unit.

### ***Keyboard Accelerator Display***

The FlatCheckBox widget displays the keyboard accelerator as part of the item’s label. The string in the `XtNacceleratorText` resource is displayed to the right of the label (or mnemonic) separated by at least one space. The `acceleratorText` is right justified.

If truncation is necessary, the accelerator is truncated as a unit. The accelerator is truncated before the mnemonic or the label.

*FlatCheckBox* Widget

*Resources*

The following tables list the resources for the FlatCheckBox widget. All of the resources are available on the FlatCheckBox container itself. Resources identified by a footnote denote item resources. If these resources are not included in the XtNitemFields list, they are inherited from the container widget. An application can change the default values for item resources by setting them on the container directly. Even though an item resource is not included in the XtNitemFields list, the application can query the value of any item resource with OlFlatGetValues(); see page 355.

Table 7-33 FlatCheckBox Core Resources

| Name                              | Type             | Default             | Access |
|-----------------------------------|------------------|---------------------|--------|
| XtNaccelerators                   | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive <sup>1</sup> | Boolean          | (calculated)        | G      |
| XtNbackground <sup>1</sup>        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap <sup>1</sup>  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor                    | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap                   | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth <sup>1</sup>       | Dimension        | 0                   | SGI    |
| XtNcolormap                       | Colormap         | (parent's)          | SGI    |
| XtNdepth                          | Cardinal         | (parent's)          | GI     |
| XtNdestroyCallback                | XtCallbackList   | NULL                | SGIO   |
| XtNheight                         | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged <sup>1</sup> | Boolean          | TRUE                | SGI    |
| XtNscreen                         | Screen *         | (parent's)          | G      |
| XtNsensitive <sup>1</sup>         | Boolean          | TRUE                | GIO    |
| XtNtranslations                   | XtTranslations   | NULL                | SGI    |
| XtNwidth                          | Dimension        | 0                   | SGI    |
| XtNx                              | Position         | 0                   | SGI    |
| XtNy                              | Position         | 0                   | SGI    |

1. This resource is available on the container and as an item resource.

Table 7-34 FlatCheckBox Primitive Resources

| Name                            | Type           | Default             | Access |
|---------------------------------|----------------|---------------------|--------|
| XtNaccelerator <sup>1</sup>     | String         | NULL                | SGI    |
| XtNacceleratorText <sup>1</sup> | String         | NULL                | SGI    |
| XtNconsumeEvent                 | XtCallbackList | NULL                | SGIO   |
| XtNfont <sup>1</sup>            | OlFont         | XtDefaultFont       | SGID   |
| XtNfontColor <sup>1</sup>       | Pixel          | XtDefaultForeground | SGID   |
| XtNforeground <sup>1</sup>      | Pixel          | XtDefaultForeground | SGID   |
| XtNinputFocusColor <sup>1</sup> | Pixel          | XtDefaultForeground | SGID   |
| XtNmnemonic <sup>1</sup>        | unsigned char  | '\0'                | SGI    |
| XtNreferenceName                | String         | NULL                | GI     |
| XtNreferenceWidget              | Widget         | NULL                | GI     |
| XtNscale                        | int            | 12                  | SGI    |
| XtNtextFormat                   | OlStrRep       | OL_SB_STR_REP       | GI     |
| XtNtraversalOn <sup>1</sup>     | Boolean        | TRUE                | SGI    |
| XtNuserData <sup>1</sup>        | XtPointer      | NULL                | SGI    |

1. This resource is available on the container and as an item resource.

Table 7-35 FlatCheckBox Flat Resources<sup>1</sup>

| Name                  | Type      | Default          | Access |
|-----------------------|-----------|------------------|--------|
| XtNgravity            | int       | CenterGravity    | SGI    |
| XtNhPad               | Dimension | 0                | SGI    |
| XtNhSpace             | Dimension | 0                | SGI    |
| XtNitemFields         | String *  | NULL             | SGI    |
| XtNitemGravity        | int       | NorthWestGravity | SGI    |
| XtNitemMaxHeight      | Dimension | OL_IGNORE        | SGI    |
| XtNitemMaxWidth       | Dimension | OL_IGNORE        | SGI    |
| XtNitemMinHeight      | Dimension | OL_IGNORE        | SGI    |
| XtNitemMinWidth       | Dimension | OL_IGNORE        | SGI    |
| XtNitems              | XtPointer | NULL             | SGI    |
| XtNitemsTouched       | Boolean   | (calculated)     | SG     |
| XtNlabel <sup>2</sup> | OlStr     | NULL             | SGI    |

*FlatCheckBox Widget*

*Table 7-35 FlatCheckBox Flat Resources<sup>1</sup> (Continued)*

| <b>Name</b>                  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------------------|-------------|----------------|---------------|
| XtNlabelImage <sup>2</sup>   | XImage *    | NULL           | SGI           |
| XtNlabelJustify <sup>2</sup> | OlDefine    | OL_LEFT        | SGI           |
| XtNlabelTile <sup>2</sup>    | Boolean     | FALSE          | SGI           |
| XtNlayoutHeight              | OlDefine    | OL_MINIMIZE    | SGI           |
| XtNlayoutType                | OlDefine    | OL_FIXEDROWS   | SGI           |
| XtNlayoutWidth               | OlDefine    | OL_MINIMIZE    | SGI           |
| XtNmanaged <sup>2</sup>      | Boolean     | TRUE           | SGI           |
| XtNmeasure                   | int         | 1              | SGI           |
| XtNnumItemFields             | Cardinal    | 0              | SGI           |
| XtNnumItems                  | Cardinal    | 0              | SGI           |
| XtNsameHeight                | OlDefine    | OL_ALL         | SGI           |
| XtNsameWidth                 | OlDefine    | OL_COLUMNS     | SGI           |
| XtNvPad                      | Dimension   | 0              | SGI           |
| XtNvSpace                    | Dimension   | 4              | SGI           |

1. These resources are defined in "Flat Resources" on page 52.
2. This resource is available on the container and as an item resource.

*Table 7-36 FlatCheckBox FlatExclusives Resources<sup>1</sup>*

| <b>Name</b>                  | <b>Type</b>    | <b>Default</b>         | <b>Access</b> |
|------------------------------|----------------|------------------------|---------------|
| XtNclientData <sup>2</sup>   | XtPointer      | NULL                   | SGI           |
| XtNdefault <sup>2</sup>      | Boolean        | FALSE                  | SGI           |
| XtNdim                       | Boolean        | FALSE                  | SGI           |
| XtNhSpace                    | Dimension      | OL_IGNORE              | SGI           |
| XtNnoneSet                   | Boolean        | FALSE                  | SGI           |
| XtNselectProc <sup>2</sup>   | XtCallbackProc | NULL                   | SGI           |
| XtNset <sup>2</sup>          | Boolean        | FALSE                  | SGI           |
| XtNunselectProc <sup>2</sup> | XtCallbackProc | NULL                   | SGI           |
| XtNvSpace                    | Dimension      | OL_IGNORE <sup>3</sup> | SGI           |

1. These resources are described under FlatExclusives, Table 7-42 on page 342.
2. This resource is available on the container and as an item resource.
3. This default overrides the value of 4 in the Flat class.



Table 7-37 FlatCheckBox Resources

| Name                     | Type     | Default | Access |
|--------------------------|----------|---------|--------|
| XtNposition <sup>1</sup> | OlDefine | OL_LEFT | SGI    |

1. This resource is available on the container and as an item resource.

### ***XtNposition***

| Class       | Type     | Default | Access |
|-------------|----------|---------|--------|
| XtCPosition | OlDefine | OL_LEFT | SGI    |

Synopsis: The side of the check box on which to place the label.

Values: OL\_LEFT/"left" - The label is placed to the left of the check box.  
 OL\_RIGHT/"right" - The label is placed to the right of the check box.

## ***Activation Types***

The following table lists the activation types used by the FlatCheckBox.

Table 7-38 FlatCheckBox Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_SELECT        | SELECT        | XtNselectBtn        |
| OL_SELECTKEY     | SELECT        | XtNselectKey        |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

Activation types not described in the following list are described in "Common Activation Types" on page 68.

---

## *FlatCheckBox Widget*

### ***OL\_SELECT/ OL\_SELECTKEY***

The activation of a FlatCheckBox is described in the *OPEN LOOK GUI Functional Specification* section “Check Boxes” in Chapter 4. When the FlatCheckBox item is activated with either OL\_SELECT or OL\_SELECTKEY, the state of the XtNset resource will be reversed. When the XtNset resource goes to FALSE, the XtNunselectProc callback will be called; when the XtNset resource goes to TRUE, the XtNselectProc callback will be called.

### *See Also*

“CheckBox Widget” on page 237,  
“Flat Widgets” on page 321,  
“FlatExclusives Widget” on page 337,  
“FlatNonexclusives Widget” on page 347,  
“Flat Widget Functions” on page 354,  
“Help Function” on page 146.

## FlatExclusives Widget

### Class

**Class Name:** FlatExclusives  
**Class Pointer:** flatExclusivesWidgetClass

### Ancestry

Core-Primitive-Flat-FlatExclusives

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/FExclusive.h>
```

### Description

The FlatExclusives widget provides the same functionality as an Exclusives widget managing RectButtons. Instead of creating individual Rectbuttons as children of a container widget, it creates items that have the same behavior as the Rectbuttons. It is useful in applications that use large arrays of exclusive settings since it requires fewer widgets to be created.

See the general explanation of flattened widgets in “Flat Widgets” on page 321.

### Selection Control

There are two modes of operation, determined by the FlatExclusives resource XtNnoneSet:

- When XtNnoneSet is FALSE, exactly one item in a FlatExclusives widget must be set. That is, the XtNset resource is TRUE for one of the items. A warning is generated if two or more items are set. If no items are set, the FlatExclusives makes the first item that is both managed and mapped when managed by the set item. No warning is produced in this case. The FlatExclusives maintains this condition by ensuring that when an item is set by the user clicking SELECT over it, the item that was set is cleared and its

## FlatExclusives Widget

XtNunselectProc procedure is called; the item under the pointer is set and its XtNselectProc procedure is called. However, clicking SELECT over an item that is already set does nothing.

- When XtNnoneSet is TRUE, at most one item in a FlatExclusives widget can be “set.” A warning is generated if two or more items are set. The FlatExclusives maintains this condition by ensuring that when an item is set by the user clicking SELECT over it, the item that was set is cleared and its XtNunselectProc procedure is called; the item under the pointer is set and its XtNselectProc procedure is called. Clicking SELECT over an item that is already set unsets it and its XtNselectProc procedure is called.

### Menu Use

The FlatExclusives widget can be added as child in a menu pane to implement a one-of-many menu choice.

### Coloration

When the FlatExclusives widget contains a number of items that is not a multiple of XtNmeasure, the empty space is colored using the parent’s XtNbackground. The labels in the items are drawn using XtNfontColor.

For 3D, the coloration of each FlatExclusives item is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.” XtNbackground is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, XtNforeground is used to render the outline of each FlatExclusives item as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, “Controls.”

If the toolkit resource XtNmouseless is set to TRUE and the toolkit resource XtNinputFocusFeedback is set to OL\_INPUT\_FOCUS\_COLOR, when the FlatExclusives receives input focus, the background of the item with focus will be drawn with the value of XtNinputFocusColor. However, if XtNinputFocusColor is the same as XtNbackground, then the widget inverts XtNforeground and XtNbackground of the item with input focus. Once the input focus leaves the widget, the original coloration is restored.

### Keyboard Traversal

The FlatExclusives widget is a Primitive widget that manages the traversal between a set of items. When the user traverses to a FlatExclusives widget, the first item in the set will display itself as having input focus (see the RectButton widget for a description of this appearance.)

The MOVEUP, MOVEDOWN, MOVERIGHT, and MOVELEFT keys move the input focus between the items. To traverse out of the FlatExclusives widget, the following keys can be used:

- NEXTFIELD moves to the next traversable widget in the window
- PREVFIELD moves to the previous traversable widget in the window
- NEXTWINDOW moves to the next window in the application.
- PREVWINDOW moves to the previous window in the application.
- NEXTAPP moves to the first window in the next application.
- PREVAPP moves to the first window in the previous application.

### Resources

The following tables list the resources for the FlatExclusives widget. All of the resources are available on the FlatExclusives container itself. Resources identified by a footnote denote item resources. If these resources are not included in the XtNitemFields list, they are inherited from the container widget. An application can change the default values for item resources by setting them on the container directly. Even though an item resource is not included in the XtNitemFields list, the application can query the value of any item resource with OlFlatGetValues(); see page 355.

Table 7-39 FlatExclusives Core Resources

| Name                              | Type             | Default             | Access |
|-----------------------------------|------------------|---------------------|--------|
| XtNaccelerators                   | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive <sup>1</sup> | Boolean          | (calculated)        | G      |
| XtNbackground <sup>1</sup>        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap <sup>1</sup>  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor                    | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap                   | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth <sup>1</sup>       | Dimension        | 0                   | SGI    |
| XtNcolormap                       | Colormap         | (parent's)          | SGI    |
| XtNdepth                          | Cardinal         | (parent's)          | GI     |

*FlatExclusives Widget*

*Table 7-39 FlatExclusives Core Resources (Continued)*

| <b>Name</b>                       | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|-----------------------------------|----------------|----------------|---------------|
| XtNdestroyCallback                | XtCallbackList | NULL           | SGIO          |
| XtNheight                         | Dimension      | 0              | SGI           |
| XtNmappedWhenManaged <sup>1</sup> | Boolean        | TRUE           | SGI           |
| XtNscreen                         | Screen *       | (parent's)     | G             |
| XtNsensitive <sup>1</sup>         | Boolean        | TRUE           | GIO           |
| XtNtranslations                   | XtTranslations | NULL           | SGI           |
| XtNwidth                          | Dimension      | 0              | SGI           |
| XtNx                              | Position       | 0              | SGI           |
| XtNy                              | Position       | 0              | SGI           |

1. This resource is available on the container and as an item resource.

*Table 7-40 FlatExclusives Primitive Resources*

| <b>Name</b>                     | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|---------------------------------|----------------|---------------------|---------------|
| XtNaccelerator <sup>1</sup>     | String         | NULL                | SGI           |
| XtNacceleratorText <sup>1</sup> | String         | NULL                | SGI           |
| XtNconsumeEvent                 | XtCallbackList | NULL                | SGIO          |
| XtNfont <sup>1</sup>            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor <sup>1</sup>       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground <sup>1</sup>      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor <sup>1</sup> | Pixel          | XtDefaultForeground | SGID          |
| XtNmnemonic <sup>1</sup>        | unsigned char  | '\0'                | SGI           |
| XtNreferenceName                | String         | NULL                | GI            |
| XtNreferenceWidget              | Widget         | NULL                | GI            |
| XtNscale                        | int            | 12                  | SGI           |
| XtNtextFormat                   | OlStrRep       | OL_SB_STR_REP       | GI            |
| XtNtraversalOn <sup>1</sup>     | Boolean        | TRUE                | SGI           |
| XtNuserData <sup>1</sup>        | XtPointer      | NULL                | SGI           |

1. This resource is available on the container and as an item resource.

Table 7-41 FlatExclusives Flat Resources<sup>1</sup>

| Name                         | Type      | Default          | Access |
|------------------------------|-----------|------------------|--------|
| XtNgravity                   | int       | CenterGravity    | SGI    |
| XtNhPad                      | Dimension | 0                | SGI    |
| XtNhSpace                    | Dimension | 0                | SGI    |
| XtNitemFields                | String *  | NULL             | SGI    |
| XtNitemGravity               | int       | NorthWestGravity | SGI    |
| XtNitemMaxHeight             | Dimension | OL_IGNORE        | SGI    |
| XtNitemMaxWidth              | Dimension | OL_IGNORE        | SGI    |
| XtNitemMinHeight             | Dimension | OL_IGNORE        | SGI    |
| XtNitemMinWidth              | Dimension | OL_IGNORE        | SGI    |
| XtNitems                     | XtPointer | NULL             | SGI    |
| XtNitemsTouched              | Boolean   | (calculated)     | SG     |
| XtNlabel <sup>2</sup>        | OlStr     | NULL             | SGI    |
| XtNlabelImage <sup>2</sup>   | XImage *  | NULL             | SGI    |
| XtNlabelJustify <sup>2</sup> | OlDefine  | OL_LEFT          | SGI    |
| XtNlabelTile <sup>2</sup>    | Boolean   | FALSE            | SGI    |
| XtNlayoutHeight              | OlDefine  | OL_MINIMIZE      | SGI    |
| XtNlayoutType                | OlDefine  | OL_FIXEDROWS     | SGI    |
| XtNlayoutWidth               | OlDefine  | OL_MINIMIZE      | SGI    |
| XtNmanaged <sup>2</sup>      | Boolean   | TRUE             | SGI    |
| XtNmeasure                   | int       | 1                | SGI    |
| XtNnumItemFields             | Cardinal  | 0                | SGI    |
| XtNnumItems                  | Cardinal  | 0                | SGI    |
| XtNsamesHeight               | OlDefine  | OL_ALL           | SGI    |
| XtNsamesWidth                | OlDefine  | OL_COLUMNS       | SGI    |
| XtNvPad                      | Dimension | 0                | SGI    |
| XtNvSpace                    | Dimension | 4                | SGI    |

1. These resources are defined in "Flat Resources" on page 52.

2. This resource is available on the container and as an item resource.

*FlatExclusives Widget*

Table 7-42 FlatExclusives Resources

| Name                         | Type           | Default                | Access |
|------------------------------|----------------|------------------------|--------|
| XtNclientData <sup>1</sup>   | XtPointer      | NULL                   | SGI    |
| XtNdefault <sup>1</sup>      | Boolean        | FALSE                  | SGI    |
| XtNdim                       | Boolean        | FALSE                  | SGI    |
| XtNhSpace                    | Dimension      | OL_IGNORE              | G      |
| XtNnoneSet                   | Boolean        | FALSE                  | SGI    |
| XtNselectProc <sup>1</sup>   | XtCallbackProc | NULL                   | SGI    |
| XtNset <sup>1</sup>          | Boolean        | FALSE                  | SGI    |
| XtNunselectProc <sup>1</sup> | XtCallbackProc | NULL                   | SGI    |
| XtNvSpace                    | Dimension      | OL_IGNORE <sup>2</sup> | G      |

1. This resource is available on the container and as an item resource.

2. This default overrides the value of 4 in the Flat class.

***XtNclientData***

| Class         | Type      | Default | Access |
|---------------|-----------|---------|--------|
| XtCClientData | XtPointer | NULL    | SGI    |

Synopsis: The client data supplied to all callback procedures.

**Note** – The widget must not modify the value in the storage area pointed to by XtNclientData. The application is responsible for allocating and freeing this area.

***XtNdefault***

| Class      | Type    | Default | Access |
|------------|---------|---------|--------|
| XtCDefault | Boolean | FALSE   | SGI    |

Synopsis: When used on the container, whether one of the items is a default item.

Values: TRUE/"true"  
FALSE/"false"

Setting this resource on the container widget indicates whether or not one of the items should be a default item. If the application sets this value on the container to:



*FlatExclusives Widget*

|       |                                                                                                                |
|-------|----------------------------------------------------------------------------------------------------------------|
| TRUE  | The container will set the first managed and mapped item as the default item if a default item does not exist. |
| FALSE | The container will unset its default item if one exists.                                                       |

Even if the application does not use `XtNdefault` in its item fields list, the container will correctly maintain the default item and the application can change the default item via `OlFlatSetValues()`.

When used on the item, this resource specifies whether or not the item is a default item. If an attempt is made to set more than one item as the default, a warning is generated and the first item to be specified as the default is selected to be the default.

***XtNdim***

| Class  | Type    | Default | Access |
|--------|---------|---------|--------|
| XtCDim | Boolean | FALSE   | SGI    |

Synopsis: The visual display of substate changes.

Values: TRUE/"true" - The item shows a dimmed visual indicating that the item represents the state of one or more objects, that as a group, are in different states.  
FALSE/"false" - Otherwise.

It is not necessary to use this resource if the application modifies the list with the `OlFlatSetValues()` procedure, nor is it necessary to use this resource whenever the application supplies a new list to the flat container. The `XtNdim` resource has the same effect as setting `XtNsensitive` for non-flat widgets.

***XtNhSpace/  
XtNvSpace***

| Class     | Type      | Default   | Access |
|-----------|-----------|-----------|--------|
| XtCHSpace | Dimension | OL_IGNORE | G      |
| XtCVSpace | Dimension | OL_IGNORE | G      |

Synopsis: The amount of horizontal/vertical space to leave between items.

Values: 0 = XtNhSpace  
0 = XtNvSpace

If the items are of different sizes in a row or column, the spacing applies to the widest or tallest dimension of all items in the row or column. Because OPEN LOOK specifies that exclusives be grouped edge-to-edge, `XtNhSpace` and `XtNvSpace` should always be zero.

*FlatExclusives Widget*

***XtNnoneSet***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCNoneSet   | Boolean     | FALSE          | SGI           |

Synopsis: Whether all the items can be in unset mode or not.  
 Values: TRUE/"true" - Zero or more items can be set at any time. The user can select the currently set item and toggle it back to an unset state.  
 FALSE/"false" - Exactly one item must be in the set state always. Attempting to select the currently set item does nothing.

***XtNselectProc***

| <b>Class</b>    | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|-----------------|----------------|----------------|---------------|
| XtCCallbackProc | XtCallbackProc | NULL           | SGI           |

Synopsis: The callback procedure invoked when an unset item is set by user input.

***XtNset***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCSet       | Boolean     | FALSE          | SGI           |

Synopsis: The current state of the item.  
 Values: TRUE/"true" - The item is set.  
 FALSE/"false" - The item is unset.

This resource is never inherited from the container, so its default value is always FALSE. Even if the application does not use XtNset in its item fields list, the container will correctly maintain the set item and the application can change the set item via OlFlatSetValues().

***XtNunselectProc***

| <b>Class</b>    | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|-----------------|----------------|----------------|---------------|
| XtCCallbackProc | XtCallbackProc | NULL           | SGI           |

Synopsis: The callback procedure invoke2zd when an set item is unset by user input.

## Activation Types

The following table lists the activation types used by the FlatExclusives.

Table 7-43 FlatExclusives Activation Types

| Activation Type   | Semantics     | Resource Name       |
|-------------------|---------------|---------------------|
| OL_CANCEL         | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION  | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP           | HELP          | XtNhelpKey          |
| OL_MENU           | MENU          | XtNmenuBtn          |
| OL_MENUDEFAULT    | MENUDEFAULT   | XtNmenuDefaultBtn   |
| OL_MENUDEFAULTKEY | MENUDEFAULT   | XtNmenuDefaultKey   |
| OL_MENUKEY        | MENU          | XtNmenuKey          |
| OL_MOVEDOWN       | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT       | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT      | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP         | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD      | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD      | PREVFIELD     | XtNprevFieldKey     |
| OL_SELECT         | SELECT        | XtNselectBtn        |
| OL_SELECTKEY      | SELECT        | XtNselectKey        |
| OL_TOGGLEPUSHPIN  | TOGGLEPUSHPIN | XtNtogglePushpinKey |

Activation types not described in the following table are described in “Common Activation Types” on page 68.”

### ***OL\_MENU/ OL\_MENUKEY***

The FlatExclusives will respond only to the OL\_MENU and OL\_MENUKEY activation types if it is a descendant of a Menu widget. When this is the case, the OL\_MENU and OL\_MENUKEY will behave as the OL\_SELECT and OL\_SELECTKEY, respectively.

### ***OL\_MENUDEFAULT/ OL\_MENUDEFAULTKEY***

The OL\_MENUDEFAULT and OL\_MENUDEFAULTKEY activation types apply only to FlatExclusives that are descendants of a Menu. These activation types

---

## *FlatExclusives Widget*

will set the MenuButton `XtNdefault` resource to `TRUE`, and change the display of the widget according to the *OPEN LOOK GUI Functional Specification* section “Changing Menu Defaults” in Chapter 15.

### ***OL\_SELECT/ OL\_SELECTKEY***

The activation of a FlatExclusives is described in the *OPEN LOOK GUI Functional Specification* section “Exclusive Settings” in Chapter 4 and in “Using Menus” in Chapter 15. When the FlatExclusives item is activated with either `OL_SELECT` or `OL_SELECTKEY`, the `XtNset` resource will be set to `TRUE` and the `XtNselectProc` callback will be called. The FlatExclusives item that was previously set will have the `XtNset` resource changed to `FALSE` and the `XtNunselectProc` callback will be called.

### *See Also*

“Exclusives Widget” on page 277,  
“Flat Widgets” on page 321,  
“FlatCheckBox Widget” on page 329,  
“FlatNonexclusives Widget” on page 347,  
“Flat Widget Functions” on page 354,  
“Help Function” on page 146.

## *FlatNonexclusives Widget*

### *Class*

*Class Name:* FlatNonexclusives  
*Class Pointer:* flatNonexclusivesWidgetClass

### *Ancestry*

Core-Primitive-Flat-FlatExclusives-FlatNonexclusives

### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/FNonexclus.h>
```

### *Description*

The FlatNonexclusives widget provides the same functionality as a Nonexclusives widget managing RectButtons. Instead of creating individual Rectbuttons as children of a container widget, it creates items that have the same behavior as the Rectbuttons. It is useful in applications that use large arrays of exclusive settings since it requires fewer widgets to be created.

See the general explanation of flattened widgets in “Flat Widgets” on page 321.

### *Default Spacing*

The default spacing between items is 50% of the prevailing point size for the container’s font.

### *Menu Use*

The FlatNonexclusives can be added as child in a menu pane to implement a several-of-many menu choice.

## FlatNonexclusives Widget

### Coloration

The background of the FlatNonexclusives container is drawn using the parent's `XtNbackground` resource. The labels in the items are drawn using `XtNfontColor`.

For 3D, the coloration of each FlatNonexclusives item is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNforeground` is used to render the outline of each FlatNonexclusives item as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

If the toolkit resource `XtNmouseless` is set to TRUE and the toolkit resource `XtNinputFocusFeedback` is set to `OL_INPUT_FOCUS_COLOR`, when the FlatNonexclusives receives input focus, the background of the item with focus will be drawn with the value of `XtNinputFocusColor`. However, if `XtNinputFocusColor` is the same as `XtNbackground`, then the widget inverts `XtNforeground` and `XtNbackground` of the item with input focus. Once the input focus leaves the widget, the original coloration is restored.

### Keyboard Traversal

The FlatNonexclusives widget is a Primitive widget that manages the traversal between a set of items. When the user traverses to a FlatNonexclusives widget, the first item in the set will display itself as having input focus (see the *RectButton Widget* for a description of this appearance.)

The `MOVEUP`, `MOVEDOWN`, `MOVERIGHT`, and `MOVELEFT` keys move the input focus between the items. To traverse out of the FlatNonexclusives widget, the following keys can be used:

- `NEXTFIELD` moves to the next traversable widget in the window
- `PREVFIELD` moves to the previous traversable widget in the window
- `NEXTWINDOW` moves to the next window in the application
- `PREVWINDOW` moves to the previous window in the application
- `NEXTAPP` moves to the first window in the next application
- `PREVAPP` moves to the first window in the previous application

## Resources

The following tables list the resources for the FlatNonexclusives widget. All of the resources are available on the FlatNonexclusives container itself. Resources identified by a footnote denote item resources. If these resources are not included in the `XtNitemFields` list, they are inherited from the container widget. An application can change the default values for item resources by setting them on the container directly. Even though an item resource is not included in the `XtNitemFields` list, the application can query the value of any item resource with `OlFlatGetValues()`; see page 355.

Table 7-44 FlatNonexclusives Core Resources

| Name                              | Type             | Default             | Access |
|-----------------------------------|------------------|---------------------|--------|
| XtNaccelerators                   | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive <sup>1</sup> | Boolean          | (calculated)        | G      |
| XtNbackground <sup>1</sup>        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap <sup>1</sup>  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor                    | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap                   | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth <sup>1</sup>       | Dimension        | 0                   | SGI    |
| XtNcolormap                       | Colormap         | (parent's)          | SGI    |
| XtNdepth                          | Cardinal         | (parent's)          | GI     |
| XtNdestroyCallback                | XtCallbackList   | NULL                | SGIO   |
| XtNheight                         | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged <sup>1</sup> | Boolean          | TRUE                | SGI    |
| XtNscreen                         | Screen *         | (parent's)          | G      |
| XtNsensitive <sup>1</sup>         | Boolean          | TRUE                | GIO    |
| XtNtranslations                   | XtTranslations   | NULL                | SGI    |
| XtNwidth                          | Dimension        | 0                   | SGI    |
| XtNx                              | Position         | 0                   | SGI    |
| XtNy                              | Position         | 0                   | SGI    |

1. This resource is available on the container and as an item resource.

*FlatNonexclusives Widget*

*Table 7-45 FlatNonexclusives Primitive Resources*

| <b>Name</b>                     | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|---------------------------------|----------------|---------------------|---------------|
| XtNaccelerator <sup>1</sup>     | String         | NULL                | SGI           |
| XtNacceleratorText <sup>1</sup> | String         | NULL                | SGI           |
| XtNconsumeEvent                 | XtCallbackList | NULL                | SGIO          |
| XtNfont <sup>1</sup>            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor <sup>1</sup>       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground <sup>1</sup>      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor <sup>1</sup> | Pixel          | XtDefaultForeground | SGID          |
| XtNmnemonic <sup>1</sup>        | unsigned char  | '\0'                | SGI           |
| XtNreferenceName                | String         | NULL                | GI            |
| XtNreferenceWidget              | Widget         | NULL                | GI            |
| XtNscale                        | int            | 12                  | SGI           |
| XtNtextFormat                   | OlStrRep       | OL_SB_STR_REP       | GI            |
| XtNtraversalOn <sup>1</sup>     | Boolean        | TRUE                | SGI           |
| XtNuserData <sup>1</sup>        | XtPointer      | NULL                | SGI           |

1. This resource is available on the container and as an item resource.

*Table 7-46 FlatNonexclusives Flat Resources<sup>1</sup>*

| <b>Name</b>           | <b>Type</b> | <b>Default</b>   | <b>Access</b> |
|-----------------------|-------------|------------------|---------------|
| XtNgravity            | int         | CenterGravity    | SGI           |
| XtNhPad               | Dimension   | 0                | SGI           |
| XtNhSpace             | Dimension   | 0                | SGI           |
| XtNitemFields         | String *    | NULL             | SGI           |
| XtNitemGravity        | int         | NorthWestGravity | SGI           |
| XtNitemMaxHeight      | Dimension   | OL_IGNORE        | SGI           |
| XtNitemMaxWidth       | Dimension   | OL_IGNORE        | SGI           |
| XtNitemMinHeight      | Dimension   | OL_IGNORE        | SGI           |
| XtNitemMinWidth       | Dimension   | OL_IGNORE        | SGI           |
| XtNitems              | XtPointer   | NULL             | SGI           |
| XtNitemsTouched       | Boolean     | (calculated)     | SG            |
| XtNlabel <sup>2</sup> | OlStr       | NULL             | SGI           |



*FlatNonexclusives Widget**Table 7-46 FlatNonexclusives Flat Resources<sup>1</sup> (Continued)*

| <b>Name</b>                  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------------------|-------------|----------------|---------------|
| XtNlabelImage <sup>2</sup>   | XImage *    | NULL           | SGI           |
| XtNlabelJustify <sup>2</sup> | OlDefine    | OL_LEFT        | SGI           |
| XtNlabelTile <sup>2</sup>    | Boolean     | FALSE          | SGI           |
| XtNlayoutHeight              | OlDefine    | OL_MINIMIZE    | SGI           |
| XtNlayoutType                | OlDefine    | OL_FIXEDROWS   | SGI           |
| XtNlayoutWidth               | OlDefine    | OL_MINIMIZE    | SGI           |
| XtNmanaged <sup>2</sup>      | Boolean     | TRUE           | SGI           |
| XtNmeasure                   | int         | 1              | SGI           |
| XtNnumItemFields             | Cardinal    | 0              | SGI           |
| XtNnumItems                  | Cardinal    | 0              | SGI           |
| XtNsameHeight                | OlDefine    | OL_ALL         | SGI           |
| XtNsameWidth                 | OlDefine    | OL_COLUMNS     | SGI           |
| XtNvPad                      | Dimension   | 0              | SGI           |
| XtNvSpace                    | Dimension   | 4              | SGI           |

1. These resources are defined in "Flat Resources" on page 52.

2. This resource is available on the container and as an item resource.

*Table 7-47 FlatNonexclusives FlatExclusives Resources<sup>1</sup>*

| <b>Name</b>                  | <b>Type</b>    | <b>Default</b>         | <b>Access</b> |
|------------------------------|----------------|------------------------|---------------|
| XtNclientData <sup>2</sup>   | XtPointer      | NULL                   | SGI           |
| XtNdefault <sup>2</sup>      | Boolean        | FALSE                  | SGI           |
| XtNdim                       | Boolean        | FALSE                  | SGI           |
| XtNhSpace                    | Dimension      | OL_IGNORE              | SGI           |
| XtNnoneSet                   | Boolean        | FALSE                  | SGI           |
| XtNselectProc <sup>2</sup>   | XtCallbackProc | NULL                   | SGI           |
| XtNset <sup>2</sup>          | Boolean        | FALSE                  | SGI           |
| XtNunselectProc <sup>2</sup> | XtCallbackProc | NULL                   | SGI           |
| XtNvSpace                    | Dimension      | OL_IGNORE <sup>3</sup> | SGI           |

1. These resources are defined under FlatExclusives, Table 7-42 on page 342.

2. This resource is available on the container and as an item resource.

3. This default overrides the value of 4 in the Flat class.

*FlatNonexclusives Widget*

The FlatNonexclusives widget has no resources other than those inherited from its superclasses.

**Activation Types**

The following table lists the activation types used by the FlatNonexclusives.

*Table 7-48 FlatNonexclusives Activation Types*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MENU                | MENU             | XtNmenuBtn           |
| OL_MENUDEFAULT         | MENUDEFAULT      | XtNmenuDefaultBtn    |
| OL_MENUDEFAULTKEY      | MENUDEFAULT      | XtNmenuDefaultKey    |
| OL_MENUKEY             | MENU             | XtNmenuKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_SELECT              | SELECT           | XtNselectBtn         |
| OL_SELECTKEY           | SELECT           | XtNselectKey         |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

Activation types not described in the following table are described in “Common Activation Types” on page 68.

***OL\_MENU/  
OL\_MENUKEY***

The FlatNonexclusives will respond only to the OL\_MENU and OL\_MENUKEY activation types if it is a descendant of a Menu widget. When this is the case, the OL\_MENU and OL\_MENUKEY will behave as the OL\_SELECT and OL\_SELECTKEY, respectively.

***OL\_MENUDEFAULT/  
OL\_MENUDEFAULTKEY***

The `OL_MENUDEFAULT` and `OL_MENUDEFAULTKEY` activation types apply only to `FlatNonexclusives` that are descendants of a `Menu`. These activation types will set the `MenuButton XtNdefault` resource to `TRUE`, and change the display of the widget according to the *OPEN LOOK GUI Functional Specification* section “Changing Menu Defaults” in Chapter 15.

***OL\_SELECT/  
OL\_SELECTKEY***

The activation of a `FlatNonexclusives` is described in the *OPEN LOOK GUI Functional Specification* section “Nonexclusive Settings” in Chapter 4 and in “Using Menus” in Chapter 15. When the `FlatNonexclusives` item is activated with either `OL_SELECT` or `OL_SELECTKEY`, the `XtNset` resource will be set to `TRUE` and the `XtNselectProc` callback will be called. The `FlatNonexclusives` item that was previously set will have the `XtNset` resource changed to `FALSE` and the `XtNunselectProc` callback will be called.

***See Also***

“Flat Widgets” on page 321,  
“FlatCheckBox Widget” on page 329,  
“FlatExclusives Widget” on page 337,  
“Flat Widget Functions” on page 354,  
“Nonexclusives Widget” on page 428,  
“Help Function” on page 146.

## Flat Widget Functions

### Flat Widget Functions

There are several convenience routines for querying or manipulating flattened widget attributes. All of these routines issue a warning if the widget ID is not a subclass of a flat widget.

#### *OlFlatCallAcceptFocus*

```
#include <Xol/OpenLook.h>
Boolean OlFlatCallAcceptFocus(
 Widget widget,
 Cardinal index,
 Time time);
```

If the specified item is capable of accepting input focus, focus is assigned to the item and `OlFlatCallAcceptFocus()` returns TRUE; otherwise, it returns FALSE.

#### *OlFlatGetFocusItem*

```
#include <Xol/OpenLook.h>
Cardinal OlFlatGetFocusItem(
 Widget widget);
```

`OlFlatGetFocusItem()` returns the item within the flattened widget that has focus. It returns `OL_NO_ITEM` if no item within the widget has focus.

#### *OlFlatGetItemIndex*

```
#include <Xol/OpenLook.h>
Cardinal OlFlatGetItemIndex(
 Widget widget,
 Position x,
 Position y);
```

`OlFlatGetItemIndex()` returns the item that contains the given *x* and *y* coordinates. It returns `OL_NO_ITEM` if no item contains the coordinate pair.

### *OlFlatGetItemGeometry*

```
#include <Xol/OpenLook.h>
void OlFlatGetItemGeometry(
 Widget widget,
 Cardinal index,
 Position *x_ret,
 Position *y_ret,
 Dimension *w_ret,
 Dimension *h_ret);
```

`OlFlatGetItemGeometry()` returns the location, width, and height of an item with respect to its flattened widget container. If the supplied item *index* is invalid, a warning is issued and the return values are set to zero.

### *OlFlatGetValues*

```
#include <Xol/OpenLook.h>
void OlFlatGetValues(
 Widget widget,
 Cardinal index,
 ArgList args,
 Cardinal num_args);
```

`OlFlatGetValues()` queries the attributes of an item. This routine is very similar to `XtGetValues()`. Applications can query any attribute of an item even if the attribute was not specified in the `XtNitemFields` resource of the flat widget container.

### *OlVaFlatGetValues*

```
#include <Xol/OpenLook.h>
void OlVaFlatGetValues(
 Widget widget,
 Cardinal index,
 ...);
```

`OlVaFlatGetValues()` is the variable-argument interface to `OlFlatGetValues()`. The variable length list of resource name/value pairs is terminated by a NULL resource name.

### *OlFlatSetValues*

```
#include <Xol/OpenLook.h>
void OlFlatSetValues(
 Widget widget,
 Cardinal index,
 ArgList args,
 Cardinal num_args);
```

`OlFlatSetValues()` sets the attributes of an item. This routine is very similar to `XtSetValues()`. Applications can set values of item attributes even if the attribute name was specified in the `XtNitemFields` resource of the flat widget container or if the item's attribute is always maintained (i.e., implicitly) by the flat widget container regardless of the `XtNitemFields` entries.

For example, the `FlatExclusives` widget always maintains the value of an item's `XtNset` attribute even if `XtNset` was not in the `XtNitemFields` resource (see “`FlatExclusives Widget`” on page 337). Therefore, an application can set the value of `XtNset` even though `XtNset` was not specified explicitly in the `XtNitemFields` resource for the widget. `XtNfont`, on the other hand, is not implicitly maintained by the `FlatExclusives` widget, so an application must specify `XtNfont` in the `XtNitemFields` resource if that application wants to change the font value via `OlFlatSetValues()`.

### *OlVaFlatSetValues*

```
#include <Xol/OpenLook.h>
void OlVaFlatGetValues(
 Widget widget,
 Cardinal index,
 ...);
```

`OlVaFlatGetValues()` is the variable-argument interface to `OlFlatSetValues()`. The variable length list of resource name/value pairs is terminated by a NULL resource name.

### *See Also*

- “Flat Widgets” on page 321,
- “FlatCheckBox Widget” on page 329,
- “FlatExclusives Widget” on page 337,
- “FlatNonexclusives Widget” on page 347,
- “Help Function” on page 146.

## FontChooser Widget

### Class

**Class Name:** FontChooser  
**Class Pointer:** fontChooserWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-RubberTile-FontChooser

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/FontCh.h>
```

### Description

The FontChooser widget provides a user interface to choose fonts conveniently in an OLIT application. The widget does this by presenting lists of meaningful typographical attributes for the user to choose from. The typographical attributes displayed are: TYPEFACE, STYLE, and SIZE. Note that these attributes are not one-to-one mapped to XLFD fields of the same names.

The typographical attributes are extracted either:

- Directly from the XLFD fonts announced by the underlying X11 server, or
- From FontSet definitions stored in the OpenWindows.fs database for the current locale. For a detailed description of the OpenWindows.fs FontSet database, see “Fontset Definitions in OpenWindows.fs” on page 358. These definitions are used only when the FontChooser operates in the internationalized mode (see “XtNtextFormat” on page 29).

The FontChooser widget has the following graphical elements:

- Scrolling lists for font attributes: TYPEFACE, STYLE, and SIZE
- Numeric field for typing in the SIZE of a scalable font
- Optional Extension container (for applications to extend the FontChooser)
- Font Preview Area (can be disabled)
  - Control for switching preview on/off

## FontChooser Widget

- Area for displaying preview text
- Apply, Revert, and Cancel buttons

The FontChooser widget is automatically instantiated when an instance of the FontChooserShell widget is created (see “FontChooserShell Widget” on page 375).

### Coloration

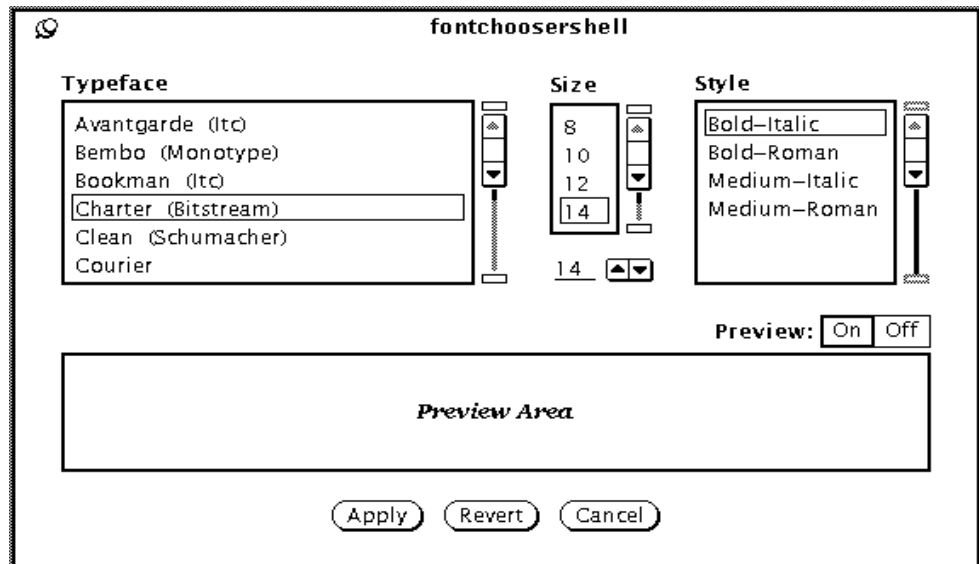


Figure 7-7 FontChooser Appearance

### Fontset Definitions in *OpenWindows.fs*

The fontset definition database contains the following:

#### Fontset Specifier

A font set is specified by using the fontset name as the resource specification and the corresponding list of X Logical Font Description (XLFD) font names as the resource value. The following example shows the fontset name

```
"-sun-myounjjo-medium-r-normal--16-140-75-75-p-140-korean-0"
```



---

## FontChooser Widget

has been defined to consist of two fonts. The keyword “definition” indicates that the value following it is a list of XLFD font names.

```
-sun-myungjo-medium-r-normal--16-140-75-75-p-140-korean-0:definition,\
-b&h-lucida-medium-r-normal-sans-0-0-0-0-p-0-iso8859-1, \
-sun-myungjo-medium-r-normal--16-140-75-75-c-140-ksc5601.1987-0
```

The OLIT FontChooser displays the fontset name, which is to the left of the keyword “definition.” However, in all its callbacks, the field *font\_name* or *current\_font\_name* provide the list of XLFD font names corresponding to the fontset name. This has been done to ensure that information provided in the callbacks is available for the programmer to manipulate directly using Xlib and Xt interfaces.

### Fontset Name Aliases

Other fontset names may be aliased to a particular fontset name using the keyword *alias*. The following example aliases the name “hngmnj14” to the fontset name “-sun-myungjo-medium-r-normal--16-140-75-75-c-140-korean-0”:

```
hngmnj14: alias, -sun-myungjo-medium-r-normal--16-140-75-75-c-140-korean-0
```

---

**Note** – These definitions are currently ignored by the FontChooser widget.

---

### Default Font Family

Font attributes *FONT\_FAMILY* can be used to create or find a fontset object. The default font family for a particular locale may be defined in the fontset definition database file as below:

```
! Default Font Set
xv_font_set.default_family: FONT_FAMILY_SANS_SERIF
```

The font family *FONT\_FAMILY\_SANS\_SERIF* has been defined as the default for the *FONT\_FAMILY* attribute.

---

**Note** – These definitions are currently ignored by the FontChooser widget.

---

### Default Font Scales

The point sizes corresponding to the font scales of a particular locale may be specified in the fontset definition database in the following manner:

*FontChooser Widget*

```
xv_font_set.small: 12
xv_font_set.medium: 14
xv_font_set.large: 16
xv_font_set.extra_large: 20
```

---

**Note** – These definitions are currently ignored by the FontChooser widget.

---

*Invoking FontChooser Using the “propertiesKey” (OL\_PROPERTY)*

There are situations where the FontChooser needs to be brought up in response to pressing the “propertiesKey.” A common situation is when text is selected in a document and hitting the propertiesKey brings up the FontChooser with the font of the text as the current font.

To achieve this, register an `XtNconsumeEvent` callback to the widget where the `propertiesKey` is expected. The `call_data` for this callback points to an `OlVirtualEventRec` (see “`OlLookupInputEvent`” on page 212 for more details of this datatype). If the `virtual_name` member of this structure is `OL_PROPERTY`, then the callback can popup a FontChooser and set the `consumed` member of the structure to `TRUE`.

The name of the font of the current text selection can be passed as `XtNinitialFontName` in the FontChooser to get it to display it as its current font.

---

**Note** – `OlAddCallback()` should be used when adding an `XtNconsumeEvent` callback (see “`XtNconsumeEvent`” on page 26).

---

*Resources*

Table 7-49 FontChooser Core Resources

| Name                              | Type                          | Default                          | Access |
|-----------------------------------|-------------------------------|----------------------------------|--------|
| <code>XtNaccelerators</code>      | <code>AcceleratorTable</code> | NULL                             | SGI    |
| <code>XtNancestorSensitive</code> | Boolean                       | TRUE                             | G      |
| <code>XtNbackground</code>        | Pixel                         | (parent's)                       | SGID   |
| <code>XtNbackgroundPixmap</code>  | Pixmap                        | <code>XtUnspecifiedPixmap</code> | SGI    |
| <code>XtNborderColor</code>       | Pixel                         | <code>XtDefaultForeground</code> | SGID   |
| <code>XtNborderPixmap</code>      | Pixmap                        | <code>XtUnspecifiedPixmap</code> | SGI    |

Table 7-49 FontChooser Core Resources (Continued)

| Name                 | Type           | Default      | Access |
|----------------------|----------------|--------------|--------|
| XtNborderWidth       | Dimension      | 0            | SGI    |
| XtNcolormap          | Colormap       | (parent's)   | SGI    |
| XtNdepth             | int            | (parent's)   | GI     |
| XtNdestroyCallback   | XtCallbackList | NULL         | SGIO   |
| XtNheight            | Dimension      | (calculated) | SGI    |
| XtNmappedWhenManaged | Boolean        | TRUE         | SGI    |
| XtNscreen            | Screen *       | (parent's)   | G      |
| XtNsensitive         | Boolean        | TRUE         | GIO    |
| XtNtranslations      | XtTranslations | NULL         | SGI    |
| XtNwidth             | Dimension      | (calculated) | SGI    |
| XtNx                 | Position       | 0            | SGI    |
| XtNy                 | Position       | 0            | SGI    |

Table 7-50 FontChooser Composite Resources

| Name              | Type        | Default | Access |
|-------------------|-------------|---------|--------|
| XtNchildren       | WidgetList  | NULL    | G      |
| XtNinsertPosition | XtOrderProc | NULL    | SGI    |
| XtNnumChildren    | Cardinal    | 0       | G      |

Table 7-51 FontChooser Manager Resources

| Name                 | Type           | Default                   | Access |
|----------------------|----------------|---------------------------|--------|
| XtNconsumeEvent      | XtCallbackList | NULL                      | SGIO   |
| XtNinputFocusColor   | Pixel          | (calculated; see page 27) | SGID   |
| XtNreferenceName     | String         | NULL                      | GI     |
| XtNreferenceWidget   | Widget         | NULL                      | GI     |
| XtNtraversalOn       | Boolean        | TRUE                      | SGI    |
| XtNunrealizeCallback | XtCallbackList | NULL                      | SGIO   |
| XtNuserData          | XtPointer      | NULL                      | SGI    |



Table 7-53 FontChooser Resources (Continued)

| Name                    | Type     | Default             | Access |
|-------------------------|----------|---------------------|--------|
| XtNpreviewSwitchOnLabel | OlStr    | “On”                | GI     |
| XtNtextFormat           | OlStrRep | OlDefaultTextFormat | GI     |
| XtNtypefaceLabel        | OlStr    | “Typeface”          | GI     |

**XtNapplyCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the APPLY button is selected.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 String current_font_name;
 OlFont current_font;
} OlFCApplyCallbackStruct;
```

Fields in the *call\_data* structure are:

|                          |                                                                                                                                                                      |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i>            | OL_REASON_APPLY_FONT                                                                                                                                                 |
| <i>current_font_name</i> | An XLFD font specification corresponding to the current choices of attributes.                                                                                       |
| <i>current_font</i>      | An XFontStruct * corresponding to <i>current_font_name</i> when XtNtextFormat is OL_SB_STR_REP, or an XFontSet when XtNtextFormat is OL_MB_STR_REP or OL_WC_STR_REP. |

The *current\_font* will be freed by the widget after the callback returns. Therefore, the application should use this field only for operations that are local to the callback; i.e., operations that would not depend on *current\_font* to persist after the callback returns.

**XtNapplyLabel**

| Class        | Type  | Default | Access |
|--------------|-------|---------|--------|
| XtCApplLabel | OlStr | “Apply” | GI     |

Synopsis: The label displayed on the APPLY button.

Values: Any OlStr value valid in the current locale.

*FontChooser Widget*

This resource has a localized default, which means that the default value shown is passed through `dgettext(3)` to arrive at the final default.

***XtNattributeListHeight***

| Class                  | Type      | Default      | Access |
|------------------------|-----------|--------------|--------|
| XtCAttributeListHeight | Dimension | (calculated) | SGI    |

Synopsis: The number of items shown in the scrolling lists.

Values:  $2 < XtNattributeListHeight$

The height of the SIZE scrolling list is `XtNattributeListHeight - 1` to accommodate the numeric field below it.

***XtNcancelCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the CANCEL button is selected.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 String current_font_name;
 OlFont current_font;
} OlFCCancelCallbackStruct;
```

Fields in the *call\_data* structure are:

- reason*                    OL\_REASON\_CANCEL
- current\_font\_name*    An XLFD font specification corresponding to the current choices of attributes.
- current\_font*            An XFontStruct \* corresponding to *current\_font\_name* when `XtNtextFormat` is OL\_SB\_STR\_REP, or an XFontSet when `XtNtextFormat` is OL\_MB\_STR\_REP or OL\_WC\_STR\_REP.

The *current\_font* will be freed by the widget after the callback returns. Therefore, the application should use this field only for operations that are local to the callback; i.e., operations that would not depend on *current\_font* to persist after the callback returns.

***XtNcancelLabel***

| Class          | Type  | Default  | Access |
|----------------|-------|----------|--------|
| XtCCancelLabel | OlStr | "Cancel" | GI     |

Synopsis: The label displayed on the CANCEL button.

Values: Any OlStr value valid in the current locale.

This resource has a localized default, which means that the default value shown is passed through `dgettext(3)` to arrive at the final default.

***XtNchangedCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when there is any change in the selections of the attributes.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 String current_font_name;
 OlFont current_font;
 String previous_font_name;
 OlFont previous_font;
} OlFCChangedCallbackStruct;
```

Fields in the *call\_data* structure are:

|                           |                                                                                                                                                                       |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i>             | OL_REASON_CHANGED_FONT                                                                                                                                                |
| <i>current_font_name</i>  | An XLFD font specification corresponding to the current choices of attributes.                                                                                        |
| <i>current_font</i>       | An XFontStruct * corresponding to <i>current_font_name</i> when XtNtextFormat is OL_SB_STR_REP, or an XFontSet when XtNtextFormat is OL_MB_STR_REP or OL_WC_STR_REP.  |
| <i>previous_font_name</i> | An XLFD font specification corresponding to the choices of attributes before the change.                                                                              |
| <i>previous_font</i>      | An XFontStruct * corresponding to <i>previous_font_name</i> when XtNtextFormat is OL_SB_STR_REP, or an XFontSet when XtNtextFormat is OL_MB_STR_REP or OL_WC_STR_REP. |

FontChooser Widget

The *current\_font* and *previous\_font* will be freed by the widget after the callback returns. Therefore, the application should use these fields only for operations that are local to the callback; i.e., operations that would not depend on *current\_font* and *previous\_font* to persist after the callback returns.

**XtNcharsetInfo**

| Class          | Type   | Default     | Access |
|----------------|--------|-------------|--------|
| XtCCharsetInfo | String | “iso8859-1” | G      |

Synopsis: Constrains the set of fonts used by the FontChooser to a certain encoding.

Values: A string containing the combination of the fields CHARSET\_REGISTRY and CHARSET\_ENCODING (separated by a “-”), which are defined as part of the X Logical Font Description (XLF D) Conventions. Typically, these two fields identify the character set and text encoding that the font handles.

When XtNtextFormat is OL\_SB\_STR\_REP, this is used to restrict the fonts only to those relevant to the current locale.

**Note** – XtNcharsetInfo is highly locale-specific and should be used with care. The localized defaults are set up carefully to handle the normal case for supported locales.

**XtNerrorCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the FontChooser detects an error condition.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 int error_num;
 String font_name;
} OlFCErrorCallbackStruct;
```

Fields in the *call\_data* structure are:

*reason*                      OL\_REASON\_ERROR





*FontChooser Widget*

***XtNinitialFontName***

| Class              | Type   | Default | Access |
|--------------------|--------|---------|--------|
| XtCInitialFontName | String | NULL    | GI     |

**Synopsis:** The attributes of the font that will be selected by the FontChooser when it starts up. If the `XtNinitialFontName` cannot be found among the fonts used by the FontChooser, then the first entry in sorted order will be selected at first.

**Values:** A fully specified XLFD string. If `XtNtextFormat` is `OL_SB_STR_REP`, this should be the fully-specified name of a font available on the underlying X11 server. If `XtNtextFormat` is not `OL_SB_STR_REP`, it should be the fully-specified definition of a font set in the `OpenWindows.fs` file for the locale; see “OLIT Toolkit Resources” on page 7.

In addition, pressing the “Revert” button anytime during an interaction with the FontChooser will cause it to revert back to the value in `XtNinitialFontName`.

***XtNmaximumPointSize***

| Class               | Type     | Default | Access |
|---------------------|----------|---------|--------|
| XtCMaximumPointSize | Cardinal | 99      | GI     |

**Synopsis:** Restricts the FontChooser to operate with point sizes up to its value.

**Values:** Any positive integer.

***XtNnoPreviewText***

| Class            | Type  | Default              | Access |
|------------------|-------|----------------------|--------|
| XtCNoPreviewText | OlStr | “Preview turned off” | GI     |

**Synopsis:** The message that the FontChooser displays in its preview area when preview is turned off.

**Values:** Any valid string in the widget’s `textFormat`.

This resource has a localized default, i.e., the default value shown is passed through `dgettext(3)` to arrive at the final default.

***XtNpreferredPointSizes***

| Class                  | Type   | Default            | Access |
|------------------------|--------|--------------------|--------|
| XtCPreferredPointSizes | String | "8 10 12 14 18 24" | GI     |

Synopsis: The list of point sizes displayed by the FontChooser in its SIZE scrolling list.

Values: A string containing a list of positive integers separated by spaces.

This resource has a localized default, which means that the default value shown is passed through `dgettext(3)` to arrive at the final default.

***XtNpreviewBorderWidth***

| Class          | Type      | Default      | Access |
|----------------|-----------|--------------|--------|
| XtCBorderWidth | Dimension | (calculated) | SGI    |

Synopsis: The border width of the preview area in pixels.

Values: Any nonnegative integer.

***XtNpreviewBackground***

| Class         | Type  | Default             | Access |
|---------------|-------|---------------------|--------|
| XtCBackground | Pixel | XtDefaultBackground | GI     |

Synopsis: The pixel value used to color the background of the preview area.

Values: Any valid pixel on the screen.

***XtNpreviewFontColor***

| Class        | Type  | Default             | Access |
|--------------|-------|---------------------|--------|
| XtCFontColor | Pixel | XtDefaultForeground | GI     |

Synopsis: The pixel value used to color the text in the preview area.

Values: Any valid pixel on the screen.

***XtNpreviewForeground***

| Class         | Type  | Default             | Access |
|---------------|-------|---------------------|--------|
| XtCForeground | Pixel | XtDefaultForeground | GI     |

Synopsis: The pixel value used to color the foreground in the preview area.

Values: Any valid pixel on the screen.

*FontChooser Widget*

***XtNpreviewHeight***

| Class            | Type      | Default      | Access |
|------------------|-----------|--------------|--------|
| XtCPreviewHeight | Dimension | (calculated) | SGI    |

Synopsis: The height of the preview area in pixels.

Values: Any nonnegative integer.

***XtNpreviewPresent***

| Class             | Type    | Default | Access |
|-------------------|---------|---------|--------|
| XtCPreviewPresent | Boolean | TRUE    | SGI    |

Synopsis: The presence of a preview area.

Values: TRUE/"true" - A preview area will be present.

FALSE/"false" - A preview area will not be present.

***XtNpreviewSwitchLabel***

| Class                 | Type  | Default    | Access |
|-----------------------|-------|------------|--------|
| XtCPreviewSwitchLabel | OlStr | "Preview:" | GI     |

Synopsis: This is the caption for the preview on/off control.

Values: Any valid string in the widget's `textFormat`.

This resource has a localized default, i.e., the default value shown is passed through `dgettext(3)` to arrive at the final default.

***XtNpreviewSwitchOffLabel***

| Class                    | Type  | Default | Access |
|--------------------------|-------|---------|--------|
| XtCPreviewSwitchOffLabel | OlStr | "Off"   | GI     |

Synopsis: This is the label on the button to switch preview OFF.

Values: Any valid string in the widget's `textFormat`.

This resource has a localized default, i.e., the default value shown is passed through `dgettext(3)` to arrive at the final default.

**XtNpreviewSwitchOnLabel**

| Class                   | Type  | Default | Access |
|-------------------------|-------|---------|--------|
| XtCPreviewSwitchOnLabel | OlStr | "On"    | GI     |

Synopsis: This is the label on the button to switch preview ON.

Values: Any valid string in the widget's `textFormat`.

This resource has a localized default, i.e., the default value shown is passed through `dgettext(3)` to arrive at the final default.

**XtNpreviewText**

| Class          | Type  | Default    | Access |
|----------------|-------|------------|--------|
| XtCPreviewText | OlStr | "%T %S %s" | SGI    |

Synopsis: Allows the application to configure the text that the FontChooser displays in its preview area when preview is on. The text that the FontChooser displays in its preview area.

Values: Any valid string in the widget's `textFormat`. The following three substrings are special if they appear in this string:

|    |                                  |
|----|----------------------------------|
| %T | Replaced by the current typeface |
| %S | Replaced by the current style    |
| %s | Replaced by the current size     |

**XtNrevertCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the REVERT button is selected and `XtNinitialFontName` is set to a valid XLFD specification.

The `call_data` structure is:

```
typedef struct {
 int reason;
 String current_font_name;
 OlFont current_font;
 String revert_font_name;
 OlFont revert_font;
} OlFCRevertCallbackStruct;
```

Fields in the `call_data` structure are:

`reason`                      OL\_REASON\_REVERT\_FONT

*FontChooser Widget*

- current\_font\_name*    An XLFD font specification corresponding to the current choices of attributes.
- current\_font*        An `XFontStruct *` corresponding to *current\_font\_name* when `XtNtextFormat` is `OL_SB_STR_REP`, or an `XFontSet` when `XtNtextFormat` is `OL_MB_STR_REP` or `OL_WC_STR_REP`.
- revert\_font\_name*    An XLFD font specification corresponding to the font it is reverting to.
- revert\_font*         An `XFontStruct *` corresponding to *revert\_font\_name* when `XtNtextFormat` is `OL_SB_STR_REP`, or an `XFontSet` when `XtNtextFormat` is `OL_MB_STR_REP` or `OL_WC_STR_REP`.

The *current\_font* and *revert\_font* will be freed by the widget after the callback returns. Therefore, the application should use these fields only for operations that are local to the callback; i.e., operations that would not depend on *current\_font* and *revert\_font* to persist after the callback returns.

***XtNrevertLabel***

| Class          | Type  | Default  | Access |
|----------------|-------|----------|--------|
| XtCRevertLabel | OlStr | "Revert" | GI     |

Synopsis:    The label displayed on the REVERT button.  
 Values:     Any `OlStr` value valid in the current locale.

This resource has a localized default, which means that the default value shown is passed through `dgettext(3)` to arrive at the final default.

***XtNsizeLabel***

| Class        | Type  | Default | Access |
|--------------|-------|---------|--------|
| XtCSizeLabel | OlStr | "Size"  | GI     |

Synopsis:    The title of the scrolling list displaying the list of SIZE attributes.  
 Values:     Any `OlStr` value valid in the current locale.

This resource has a localized default, which means that the default value shown is passed through `dgettext(3)` to arrive at the final default.

**XtNstyleLabel**

| Class         | Type  | Default | Access |
|---------------|-------|---------|--------|
| XtCStyleLabel | OlStr | "Style" | GI     |

Synopsis: The title of the scrolling list displaying the list of STYLE attributes.

Values: Any OlStr value valid in the current locale.

This resource has a localized default, which means that the default value shown is passed through `dgettext(3)` to arrive at the final default.

**XtNtextFormat**

| Class         | Type     | Default             | Access |
|---------------|----------|---------------------|--------|
| XtCTextFormat | OlStrRep | OlDefaultTextFormat | GI     |

Synopsis: The text representation type for the widget. The toolkit computes the default setting.

Values: OL\_SB\_STR\_REP - Use the fonts announced by the X11 server.  
 OL\_MB\_STR\_REP or OL\_WC\_STR\_REP - Use fontsets specified in the locale-specific fontset database stored in the file  
`$OPENWINHOME/lib/locale/%L/OW_FONT_SETS/OpenWindows.fs`  
 (where %L is replaced by the locale name). See "Fontset Definitions in OpenWindows.fs" on page 358.

Because the text representation type determines whether `XFontStruct *` or `XFontSet` is needed for rendering text, this resource controls the source of font names for the FontChooser.

**XtNtypefaceLabel**

| Class            | Type  | Default    | Access |
|------------------|-------|------------|--------|
| XtCTypefaceLabel | OlStr | "Typeface" | GI     |

Synopsis: The title of the scrolling list displaying the list of TYPEFACE attributes.

Values: Any OlStr value valid in the current locale.

This resource has a localized default, which means that the default value shown is passed through `dgettext(3)` to arrive at the final default.

---

## FontChooser Widget

### Activation Types

The following table lists the activation types used by the FontChooser.

Table 7-55 FontChooser Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

The FontChooser widget has no activation types besides the ones in “Common Activation Types” on page 68.

### See Also

“RubberTile Widget” on page 502.



## FontChooserShell Widget

### Class

**Class Name:** FontChooserShell  
**Class Pointer:** fontChooserShellWidgetClass

### Ancestry

Core-Composite-Shell-WMShell-VendorShell-TransientShell-FontChooserShell

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/FontChSh.h>
```

### Description

The FontChooserShell widget provides a mechanism to directly instantiate a FontChooser popup as defined in the *OPEN LOOK GUI Functional Specification*.

The FontChooserShell internally instantiates a FontChooser widget as its child to provide most of the functionality. It adds the FontChooser popup/down semantics defined in the *OPEN LOOK GUI Functional Specification* to this basic functionality.

### Components

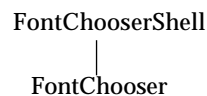


Figure 7-8 FontChooserShell Components

Coloration

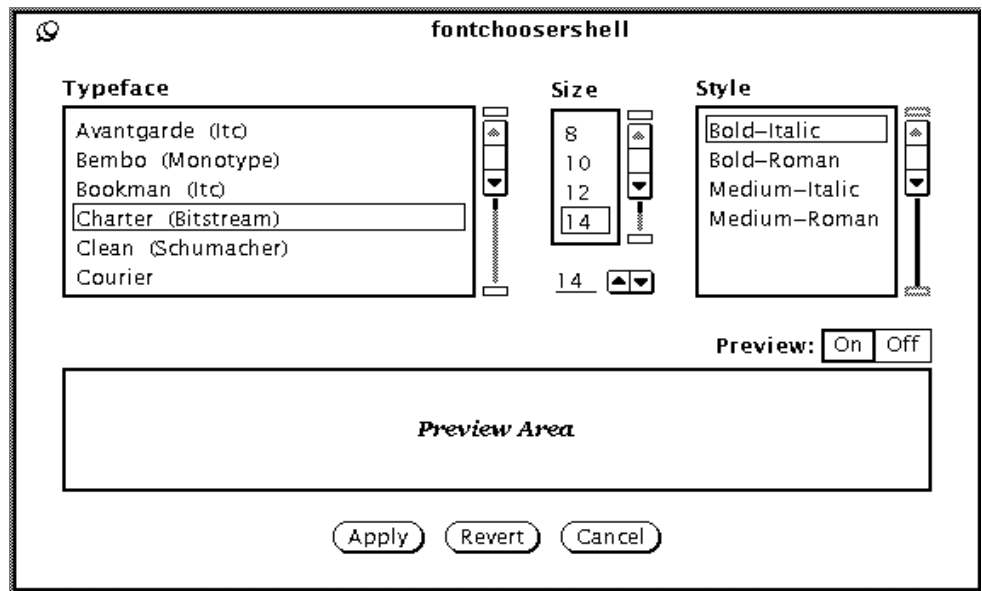


Figure 7-9 FontChooserShell Appearance

Resources

Table 7-56 FontChooserShell Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | (parent's)          | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 0                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | GI     |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | (calculated)        | SG     |

*FontChooserShell Widget**Table 7-56 FontChooserShell Core Resources (Continued)*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNmappedWhenManaged | Boolean        | TRUE           | SGI           |
| XtNscreen            | Screen *       | (parent's)     | GI            |
| XtNsensitive         | Boolean        | TRUE           | GIO           |
| XtNtranslations      | XtTranslations | NULL           | SGI           |
| XtNwidth             | Dimension      | (calculated)   | SG            |
| XtNx                 | Position       | 0              | SGI           |
| XtNy                 | Position       | 0              | SGI           |

*Table 7-57 FontChooserShell Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 7-58 FontChooserShell Shell Resources*

| <b>Name</b>             | <b>Type</b>            | <b>Default</b> | <b>Access</b> |
|-------------------------|------------------------|----------------|---------------|
| XtNallowShellResize     | Boolean                | TRUE           | SGI           |
| XtNcreatePopupChildProc | XtCreatePopupChildProc | NULL           | SGI           |
| XtNgeometry             | String                 | NULL           | GI            |
| XtNoverrideRedirect     | Boolean                | FALSE          | SGI           |
| XtNpopupdownCallback    | XtCallbackList         | NULL           | SGIO          |
| XtNpopupCallback        | XtCallbackList         | NULL           | SGIO          |
| XtNsaveUnder            | Boolean                | FALSE          | SGI           |
| XtNvisual               | Visual *               | (parent's)     | GIO           |

*Table 7-59 FontChooserShell WMSHELL Resources*

| <b>Name</b>   | <b>Type</b> | <b>Default</b>        | <b>Access</b> |
|---------------|-------------|-----------------------|---------------|
| XtNbaseHeight | int         | XtUnspecifiedShellInt | SGI           |
| XtNbaseWidth  | int         | XtUnspecifiedShellInt | SGI           |
| XtNheightInc  | int         | XtUnspecifiedShellInt | SGI           |

*FontChooserShell Widget*

*Table 7-59 FontChooserShell WMSHELL Resources (Continued)*

| <b>Name</b>      | <b>Type</b>  | <b>Default</b>        | <b>Access</b> |
|------------------|--------------|-----------------------|---------------|
| XtNiconMask      | Pixmap       | NULL                  | SGI           |
| XtNiconPixmap    | Pixmap       | NULL                  | SGI           |
| XtNiconWindow    | Window       | NULL                  | SGI           |
| XtNiconX         | int          | XtUnspecifiedShellInt | GI            |
| XtNiconY         | int          | XtUnspecifiedShellInt | GI            |
| XtNinitialState  | InitialState | NormalState           | SGI           |
| XtNinput         | Bool         | FALSE                 | G             |
| XtNmaxAspectX    | int          | XtUnspecifiedShellInt | SGI           |
| XtNmaxAspectY    | int          | XtUnspecifiedShellInt | SGI           |
| XtNmaxHeight     | int          | OL_IGNORE             | SGI           |
| XtNmaxWidth      | int          | OL_IGNORE             | SGI           |
| XtNminAspectX    | int          | XtUnspecifiedShellInt | SGI           |
| XtNminAspectY    | int          | XtUnspecifiedShellInt | SGI           |
| XtNminHeight     | int          | OL_IGNORE             | SGI           |
| XtNminWidth      | int          | OL_IGNORE             | SGI           |
| XtNtitle         | String       | NULL                  | SGI           |
| XtNtitleEncoding | Atom         | XA_STRING             | SGI           |
| XtNtransient     | Boolean      | TRUE                  | SGI           |
| XtNwaitForWm     | Boolean      | TRUE                  | SGI           |
| XtNwidthInc      | int          | XtUnspecifiedShellInt | SGI           |
| XtNwindowGroup   | Window       | XtUnspecifiedWindow   | SGI           |
| XtNwinGravity    | int          | XtUnspecifiedShellInt | GI            |
| XtNwmTimeout     | int          | 5000 (msec)           | SGI           |

*Table 7-60 FontChooserShell VendorShell Resources*

| <b>Name</b>      | <b>Type</b>       | <b>Default</b>    | <b>Access</b> |
|------------------|-------------------|-------------------|---------------|
| XtNbusy          | Boolean           | FALSE             | SGI           |
| XtNconsumeEvent  | XtCallbackList    | NULL              | SGIO          |
| XtNdefaultImName | String            | NULL              | SGI           |
| XtNfooterPresent | Boolean           | FALSE             | SGI           |
| XtNfocusWidget   | Widget            | (see description) | SGI           |
| XtNimFontSet     | OIFont            | XtDefaultFontSet  | SGI           |
| XtNimStatusStyle | OIFontStatusStyle | OL_NO_STATUS      | GI            |

*FontChooserShell Widget**Table 7-60* FontChooserShell VendorShell Resources (Continued)

| <b>Name</b>             | <b>Type</b>    | <b>Default</b>                            | <b>Access</b> |
|-------------------------|----------------|-------------------------------------------|---------------|
| XtNleftFooterString     | OIStr          | NULL                                      | SGI           |
| XtNleftFooterVisible    | Boolean        | TRUE                                      | SGI           |
| XtNmenuButton           | Boolean        | (see description)                         | GI            |
| XtNmenuType             | OIDefine       | (see description)                         | SGI           |
| XtNpushpin              | OIDefine       | (see description)                         | SGI           |
| XtNresizeCorners        | Boolean        | (see description)                         | SGI           |
| XtNrightFooterString    | OIStr          | NULL                                      | SGI           |
| XtNrightFooterVisible   | Boolean        | TRUE                                      | SGI           |
| XtNshellTitle           | OIStr          | NULL                                      | SGI           |
| XtNuserData             | XtPointer      | NULL                                      | SGI           |
| XtNwindowHeader         | Boolean        | (see description)                         | GI            |
| XtNwmProtocol           | XtCallbackList | NULL                                      | SGIO          |
| XtNwmProtocolInterested | int            | OL_WM_DELETE_WINDOW  <br>OL_WM_TAKE_FOCUS | I             |

*Table 7-61* FontChooserShell TransientShell Resources

| <b>Name</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-----------------|-------------|----------------|---------------|
| XtNtransientFor | Widget      | NULL           | SGI           |

*Table 7-62* FontChooserShell Resources

| <b>Name</b>          | <b>Type</b> | <b>Default</b>      | <b>Access</b> |
|----------------------|-------------|---------------------|---------------|
| XtNfontChooserWidget | Widget      | (calculated)        | G             |
| XtNtextFormat        | OIStrRep    | OIDefaultTextFormat | GI            |

***XtNfontChooserWidget***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCWidget    | Widget      | (calculated)   | G             |

**Synopsis:** The FontChooser child widget that can be accessed for setting or getting its resources; see “FontChooser Widget” on page 357 for its resources.

*FontChooserShell* Widget

***XtNtextFormat***

| <b>Class</b>  | <b>Type</b> | <b>Default</b>      | <b>Access</b> |
|---------------|-------------|---------------------|---------------|
| XtCTextFormat | OIStrRep    | OIDefaultTextFormat | GI            |

Synopsis: The text format propagated to the child *FontChooser* widget.

Values: OL\_SB\_STR\_REP - Single-byte character representation.  
 OL\_WC\_STR\_REP - Wide character representation.  
 OL\_MB\_STR\_REP - Multibyte character representation.

See “*XtNtextFormat*” on page 29 for details of initialization and the default value.

***Activation Types***

The following table lists the activation types used by the *FontChooserShell*.

*Table 7-63* NoticeShell Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The *FontChooserShell* widget has no activation types besides the ones in “Common Activation Types” on page 68.

***See Also***

“*FontChooser* Widget” on page 357.

## *FooterPanel Widget*

---

**Note** – The FooterPanel widget is obsolete but remains in the toolkit for backward compatibility. Its functionality has been superseded by the VendorShell’s XtNFooterPresent, XtNleftFooterVisible, XtNrightFooterVisible, XtNleftFooterString, and XtNrightFooterString resources. See “VendorShell Resources” on page 42 for descriptions of these resources.

---

### *Class*

*Class Name:* FooterPanel  
*Class Pointer:* footerPanelWidgetClass

### *Ancestry*

Core-Composite-Constraint-Manager-FooterPanel

### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/FooterPane.h>
```

### *Description*

The FooterPanel is a simple composite widget that provides a consistent interface for attaching a footer message to the bottom of a base window. The FooterPanel composite accepts two children: a top child and a footer child. (These are attached to the top and bottom of the FooterPanel widget, respectively.) The children are identified in the order they are added: the top child is the first child added; the footer child is the second.

The initial height of the FooterPanel widget is the sum of the initial heights of its children. The initial width is the widest of the initial widths of its children.

*FooterPanel Widget*

*Sizing*

The FooterPanel widget attempts to allow its children to grow or shrink to any size, by asking its parent to allow it to grow to the width of the widest child and the height of the sum of its children’s height.

When it is not allowed to grow to this desired size, or when it is resized smaller by its parent, the FooterPanel imposes the size restriction as follows: It resizes both children to its width, but forces the top Child to absorb all the height restriction; it does not resize the height of the footer child. Conversely, when it is resized larger by its parent, the FooterPanel widget gives all the height increase to the top child and resizes both children to the new width.

The FooterPanel widget never overlaps its children. If necessary, it will resize the top Child to zero height. If its height becomes too small to accommodate the footer child’s height, it clips the footer child.

*Limitations*

The FooterPanel widget works with all OLIT widgets except those that are subclassed from the Shell widget class.

*Coloration*

The FooterPanel widget has no coloration of its own and its two child widgets are colored independently.

*Resources*

Table 7-64 FooterPanel Core Resources

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |



*FooterPanel Widget**Table 7-64 FooterPanel Core Resources (Continued)*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNcolormap          | Colormap       | (parent's)     | SGI           |
| XtNdepth             | int            | (parent's)     | GI            |
| XtNdestroyCallback   | XtCallbackList | NULL           | SGIO          |
| XtNheight            | Dimension      | 0              | SGI           |
| XtNmappedWhenManaged | Boolean        | TRUE           | SGI           |
| XtNscreen            | Screen *       | (parent's)     | G             |
| XtNsensitive         | Boolean        | TRUE           | GIO           |
| XtNtranslations      | XtTranslations | NULL           | SGI           |
| XtNwidth             | Dimension      | 0              | SGI           |
| XtNx                 | Position       | 0              | SGI           |
| XtNy                 | Position       | 0              | SGI           |

*Table 7-65 FooterPanel Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 7-66 FooterPanel Manager Resources*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor   | Pixel          | Red            | SGID          |
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

---

## FooterPanel Widget

### Activation Types

The following table lists the activation types used by the FooterPanel.

*Table 7-67* FooterPanel Activations Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

The FooterPanel widget has no activation types besides the ones in “Common Activation Types” on page 68.

## Form Widget

### Class

**Class Name:** Form  
**Class Pointer:** formWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-Form

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/Form.h>
```

### Description

The Form widget is a constraint-based manager that provides a layout language used to establish spatial relationships between its children. It then manipulates these relationships when the Form is resized, new children are added to the Form, or its children are moved, resized, unmanaged, remanaged, rearranged, or destroyed. The Form composite widget works with all the OLIT widgets, except those that are sub-classed from the Shell widget class.

### Spanning Constraints

The Form provides a set of constraint resources that dictate the layout of its children when they are created and when the Form is resized. These constraints can be set in the horizontal, vertical, or both directions.

### Horizontal Constraints

These constraints control the positioning of a child in the horizontal direction and the changes in its layout or dimensions when the Form is resized horizontally.

**Vertical Constraints**

These constraints control the positioning of a child in the vertical direction and the changes in its layout or dimensions when the Form is resized vertically.

**Automatic Resizing**

The Form calculates new sizes or positions for its children whenever they change size or position. The new form size thus generated is passed as a geometry request to the parent of the form. Once resized, the Form rearranges its children based on the children’s constraints.

**Child Management**

When a widget within a form is unmanaged or destroyed, it is removed from the constraint processing and the constraints are reprocessed to resize the form and reposition and/or resize the form’s contents. Any widgets that referenced it are rereferenced to the widget that it had been referencing. For the unmanaged case, if the widget is remanaged, the widgets that were previously referencing it are rereferenced to it, thereby reestablishing the original layout.

**Coloration**

The following diagram illustrates the resources used for Form Coloration.

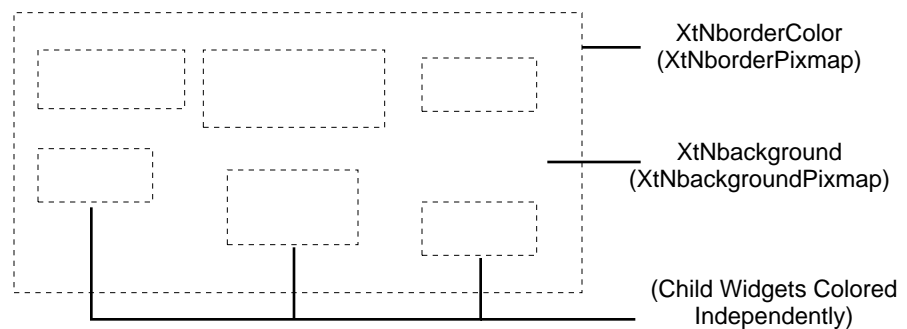


Figure 7-10 Form Coloration

## Form Geometry Management Algorithm

Whenever the Form widget is resized smaller or larger than its current size it recomputes the positions and dimensions of its children based on their constraints.

### Terminology

*Form\_Width/Form\_Height*: The width/height of the Form is set through external resize commands on the Form. Typically, these are set by the user through the Window Manager.

*Reference\_Tree*: The Form represents the layout of its children through a tree data structure referred to as a *Reference\_Tree*. It internally constructs two *Reference\_Trees*:

- A *horizontal Reference\_Tree*, whose branches consist of widgets that “xReference” one another (i.e., they are linked by the `XtNxRefWidget` or `XtNxRefName` resources), and
- A *vertical Reference\_Tree*, whose branches consist of widgets that “yReference” one another (i.e., they are linked by the `XtNyRefWidget` or `XtNyRefName` resources).

*Branch\_Width/Branch\_Height*: The width of a branch in the horizontal *Reference\_Tree* or the height of a branch in the vertical *Reference\_Tree*.

*Child\_Width/Child\_Height*: The initial dimensions or dimensions set through `XtSetValues()` for a child widget.

*Consume\_Width/Consume\_Height*: The difference between *Form\_Width/Form\_Height* and *Branch\_Width/Branch\_Height*.

*Resizable\_Widget*: A widget whose `XtNxResizable/XtNyResizable` is TRUE.

### Resize Algorithm

The Resize algorithm used by the Form applies the same logic to each branch in the horizontal and vertical *Reference\_Trees*.

For each branch in the horizontal *Reference\_Tree*:

1. The Form computes the *Branch\_Width* by adding up the *Child\_Widths* and the `XtNxOffset` resource of each widget in this branch.

## Form Widget

---

2. The `Form_Width` may be larger or smaller than the computed `Branch_Width` of this branch. The Form attempts to shrink or expand this branch so that its width becomes equal to `Form_Width`. It does so in the following manner:
  - a. If `Form_Width > Branch_Width`: The Form tries to distribute the extra width (i.e.,  $\text{Consume\_Width} = \text{Form\_Width} - \text{Branch\_Width}$ ) by repositioning or resizing the children in this branch. It attempts this in the following order:
    - i. **Reposition widgets:** Find out widgets whose `XtNxVaryOffset` is `TRUE` and uniformly increase their offsets from their `xReferenceWidgets` and thus distribute `Consume_Width` equally among them.
    - ii. **Resize widgets:** If the previous step fails and `XtNxAttachRight` is `TRUE` for the rightmost child widget, resize the first `Resizable_Widget` starting from the right, so that `Consume_Width` is completely consumed by that widget.
    - iii. **Do nothing:** If the previous step fails, no repositioning or resizing occurs.
  - b. If `Form_Width < Branch_Width`: The Form tries to reposition or shrink widgets in this branch so that all children can be accommodated. It attempts this in the following order:
    - i. **Reposition widgets:** Find out widgets whose `XtNxVaryOffset` is `TRUE` and uniformly decrease their offsets from their `xReferenceWidgets` until every offset becomes zero or `Consume_Width` is totally accounted for.
    - ii. **Resize widgets:** If the previous step fails or if `Consume_Width` is still not zero, shrink all `Resizable_Widgets` uniformly, until all their widths become 1 or `Consume_Width` is totally accounted for.
    - iii. **Do nothing:** If the previous step fails, no further repositioning or resizing occurs.

The same algorithm is repeated for each branch in the vertical `Reference_Tree`.

*Resources**Table 7-68* Form Core Resources

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | 0                   | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 7-69* Form Composite Resources

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 7-70* Form Manager Resources

| <b>Name</b>        | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|--------------------|----------------|----------------|---------------|
| XtNconsumeEvent    | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor | Pixel          | Red            | SGID          |

## Form Widget

Table 7-70 Form Manager Resources (Continued)

| Name                 | Type           | Default | Access |
|----------------------|----------------|---------|--------|
| XtNreferenceName     | String         | NULL    | GI     |
| XtNreferenceWidget   | Widget         | NULL    | GI     |
| XtNtraversalOn       | Boolean        | TRUE    | SGI    |
| XtNunrealizeCallback | XtCallbackList | NULL    | SGIO   |
| XtNuserData          | XtPointer      | NULL    | SGI    |

## Constraint Resources

Each child widget attached to the Form composite widget is constrained by the resources detailed in the following table. In essence, these resources become resources for each child widget and can be set and read just like any other resources defined for the child.

Table 7-71 Form Resources

| Name             | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtNxAddWidth     | Boolean | FALSE   | SGI    |
| XtNxAttachOffset | int     | 0       | SGI    |
| XtNxAttachRight  | Boolean | FALSE   | SGI    |
| XtNxOffset       | int     | 0       | SGI    |
| XtNxRefName      | String  | NULL    | SGI    |
| XtNxRefWidget    | Widget  | (form)  | SGI    |
| XtNxResizable    | Boolean | FALSE   | SGI    |
| XtNxVaryOffset   | Boolean | FALSE   | SGI    |
| XtNyAddHeight    | Boolean | FALSE   | SGI    |
| XtNyAttachBottom | Boolean | FALSE   | SGI    |
| XtNyAttachOffset | int     | 0       | SGI    |
| XtNyOffset       | int     | 0       | SGI    |
| XtNyRefName      | String  | NULL    | SGI    |
| XtNyRefWidget    | Widget  | (form)  | SGI    |
| XtNyResizable    | Boolean | FALSE   | SGI    |
| XtNyVaryOffset   | Boolean | FALSE   | SGI    |



***XtNxAddWidth/  
XtNyAddHeight***

| Class         | Type    | Default | Access |
|---------------|---------|---------|--------|
| XtCXAddWidth  | Boolean | FALSE   | SGI    |
| XtCYAddHeight | Boolean | FALSE   | SGI    |

Synopsis: The addition of the width/height of the corresponding reference widget to the location of a widget when determining the position of a widget.

Values: TRUE/"true" - Add the width/height.  
FALSE/"false" - Do not add the width/height.

***XtNxAttachOffset/  
XtNyAttachOffset***

| Class            | Type | Default | Access |
|------------------|------|---------|--------|
| XtCXAttachOffset | int  | 0       | SGI    |
| XtCYAttachOffset | int  | 0       | SGI    |

Synopsis: The separation between the Form widget and its children.

Values: 0 < XtNxAttachOffset  
0 < XtNyAttachOffset

When a widget is attached to the right or bottom edge of the form, the separation between the widget and the form defaults to zero pixels. These resources allow that separation to be set to some other value. Also, for widgets that are not attached to the right or bottom edge of the form, these resources specify the minimum spacing between the widget and the form.

***XtNxAttachRight/  
XtNyAttachBottom***

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCXAttachRight  | Boolean | FALSE   | SGI    |
| XtCYAttachBottom | Boolean | FALSE   | SGI    |

Synopsis: The reference direction on the form.

Values: TRUE/"true" - See below.  
FALSE/"false" - See below.

These resources indicate that the specified widget should always be attached to the right edge or bottom edge of the Form. Whenever the Form gets resized, it will reposition or resize this child or its siblings so that it always maintains the

*Form Widget*

above condition. These resources are used in conjunction with the `XtNxVaryOffset`, `XtNyVaryOffset`, `XtNxResizable`, and `XtNyResizable` resources to control whether a child gets repositioned or resized when the parent Form is resized.

***XtNxOffset/  
XtNyOffset***

| Class                   | Type | Default | Access |
|-------------------------|------|---------|--------|
| <code>XtCXOffset</code> | int  | 0       | SGI    |
| <code>XtCYOffset</code> | int  | 0       | SGI    |

Synopsis: The offset from the reference widget.

Values: 0 < `XtNxOffset`  
0 < `XtNyOffset`

The location of a widget is determined by the widget it references. By default, a widget's location exactly matches its reference widget's location. These resources specify the offset of a widget from its reference widget's location. The actual location of a widget is computed as:

```

if (XtNxAddWidth == TRUE)
 x_location = x_location_of_ref_widget +
 width_of_ref_widget + XtNxOffset
else
 x_location = x_location_of_ref_widget + XtNxOffset
if (XtNyAddHeight == TRUE)
 y_location = y_location_of_ref_widget +
 height_of_ref_widget + XtNyOffset
else
 y_location = y_location_of_ref_widget + XtNyOffset

```

***XtNxRefName/  
XtNyRefName***

| Class                    | Type   | Default | Access |
|--------------------------|--------|---------|--------|
| <code>XtCXRefName</code> | String | NULL    | SGI    |
| <code>XtCYRefName</code> | String | NULL    | SGI    |

Synopsis: The reference widget by name.

Values: The name of a widget already created as a child of the form.

When a widget is added as a child of the form, its position is determined by the widget it references. These resources allow the name of the reference widget to be given. The form will convert this name to a widget to use for the

*Form Widget*

referencing. Any widget that is a direct child of the form or the form widget itself can be used as a reference widget. If one of these resources is set and the corresponding resource, `XtNxRefWidget` or `XtNyRefWidget`, is also set, they are required to agree—the name given in `XtNxRefName` or `XtNyRefName` must match the name of the identified widget.

***XtNxRefWidget/  
XtNyRefWidget***

| Class                      | Type   | Default | Access |
|----------------------------|--------|---------|--------|
| <code>XtCXRefWidget</code> | Widget | (form)  | SGI    |
| <code>XtCYRefWidget</code> | Widget | (form)  | SGI    |

Synopsis: The reference widget by ID.

Values: The ID of a widget already created as a child.

Instead of naming the reference widget, an application can give the ID of the reference widget using these resources. If both a widget ID and a widget name is given for a reference in the same dimension (x or y), they are required to refer to the same widget.

***XtNxResizable/  
XtNyResizable***

| Class                      | Type    | Default | Access |
|----------------------------|---------|---------|--------|
| <code>XtCXResizable</code> | Boolean | FALSE   | SGI    |
| <code>XtCYResizable</code> | Boolean | FALSE   | SGI    |

Synopsis: The Form policy for resizing children.

Values: TRUE/"true" - The Form will resize its children.

FALSE/"false" - The Form will not resize its children.

When a Form's size changes, it recomputes the dimensions and positions of its children, to accommodate all of them within its new size. These resources indicate to the Form that this child widget can be resized to achieve the new layout.

*Form Widget*

***XtNxVaryOffset/  
XtNyVaryOffset***

| <b>Class</b>   | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|----------------|-------------|----------------|---------------|
| XtCXVaryOffset | Boolean     | FALSE          | SGI           |
| XtCYVaryOffset | Boolean     | FALSE          | SGI           |

Synopsis: The variable spacing between a widget and its reference widget.  
 Values: TRUE/"true" - See below.  
 FALSE/"false" - See below.

When a Form's size changes, it recomputes the dimensions and positions of its children to accommodate all of them within its new size. These resources indicate to the Form that this child widget can be repositioned (i.e., the offset from its reference widget can be varied) to achieve the new layout. The offset of any widget that directly references the Form never varies.

***Activation Types***

The following table lists the activation types used by the Form.

*Table 7-72 Form Activation Types*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The Form widget has no activation types besides the ones in "Common Activation Types" on page 68.

***See Also***

"RubberTile Widget" on page 502.

### *Gauge Widget*

#### *Class*

*Class Name:* Gauge  
*Class Pointer:* gaugeWidgetClass

#### *Ancestry*

Core-Primitive-Gauge

#### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/Gauge.h>
```

#### *Description*

The Gauge widget displays a numeric value graphically. It is similar to a Slider except that it is read-only.

### Components

The Gauge widget consists of the following components:

- Bar with shaded region indicating the current value (oriented either horizontally or vertically)
- Ticks (optional)
- Minimum value (optional)
- Maximum value (optional)
- Current value (optional—must be created and managed by the application)

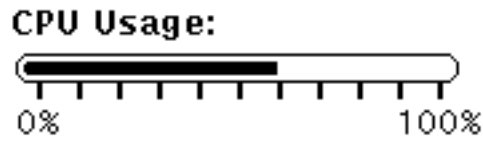


Figure 8-1 Gauge Horizontal Orientation

### Coloration

For 3D and 2D, the area surrounding the Gauge and its labels is drawn with the parent's `XtNbackground`. `XtNfontColor` is used to draw the minimum and maximum value labels.

For 3D, the gauge component and tickmark coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1. `XtNforeground` is used to draw the current-value indicator bar inside the Gauge.

For 2D, `XtNbackground` and `XtNforeground` are used to render the gauge component and the tickmarks, as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

### Application Notification

The application is responsible for setting the current value of the Gauge as well as creating a widget (such as a `StaticText` or `TextLine`) to display the current value numerically.

*Resources**Table 8-1 Gauge Core Resources*

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | 0                   | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 8-2 Gauge Primitive Resources*

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|--------------------|----------------|---------------------|---------------|
| XtNaccelerator     | String         | NULL                | SGI           |
| XtNacceleratorText | String         | NULL                | SGI           |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor | Pixel          | Red                 | SGID          |
| XtNmnemonic        | unsigned char  | '\0'                | SGI           |
| XtNreferenceName   | String         | NULL                | GI            |

Gauge Widget

Table 8-2 Gauge Primitive Resources (Continued)

| Name               | Type      | Default       | Access |
|--------------------|-----------|---------------|--------|
| XtNreferenceWidget | Widget    | NULL          | GI     |
| XtNscale           | int       | 12            | SGI    |
| XtNtextFormat      | OlStrRep  | OL_SB_STR_REP | GI     |
| XtNtraversalOn     | Boolean   | TRUE          | SGI    |
| XtNuserData        | XtPointer | NULL          | SGI    |

Table 8-3 Gauge Resources

| Name             | Type      | Default     | Access |
|------------------|-----------|-------------|--------|
| XtNleftMargin    | Dimension | OL_IGNORE   | SGI    |
| XtNmaxLabel      | OlStr     | NULL        | SGI    |
| XtNminLabel      | OlStr     | NULL        | SGI    |
| XtNorientation   | OlDefine  | OL_VERTICAL | GI     |
| XtNrecomputeSize | Boolean   | FALSE       | SGI    |
| XtNrightMargin   | Dimension | OL_IGNORE   | SGI    |
| XtNsliderMax     | int       | 100         | SGI    |
| XtNsliderMin     | int       | 0           | SGI    |
| XtNsliderValue   | int       | 0           | SGI    |
| XtNspan          | Dimension | OL_IGNORE   | SGI    |
| XtNticks         | int       | 0           | SGI    |
| XtNtickUnit      | OlDefine  | OL_NONE     | SGI    |

***XtNleftMargin***

| Class     | Type      | Default   | Access |
|-----------|-----------|-----------|--------|
| XtCMargin | Dimension | OL_IGNORE | SGI    |

Synopsis: The number of pixels in the margin to the left of the gauge.

Values: OL\_IGNORE/"ignore" or any valid Dimension.

***XtNmaxLabel***

| Class    | Type  | Default | Access |
|----------|-------|---------|--------|
| XtCLabel | OlStr | NULL    | SGI    |

Synopsis: The label to be placed next to the maximum value position.



Values: Any `OlStr` value valid in the current locale.

For a vertical gauge, the label is placed to the right of the maximum value position. If there is not enough space for the entire label and `XtNrecomputeSize` is `TRUE`, then the widget will request more space to show the entire label.

For a horizontal gauge, the label is placed centered and below the maximum value position. If there is not enough room to center the label and `XtNrecomputeSize` is set to `FALSE`, the end of the label will be aligned with the right end of the outline of the gauge. If this label collides with the minimum label, some part of the labels will overlap. If there is not enough room to center the label and `XtNrecomputeSize` is set to `TRUE`, then the widget will request more space to center the label below the maximum value position.

### ***XtNminLabel***

| Class                 | Type               | Default           | Access           |
|-----------------------|--------------------|-------------------|------------------|
| <code>XtCLabel</code> | <code>OlStr</code> | <code>NULL</code> | <code>SGI</code> |

Synopsis: The label to be placed next to the minimum value position.

Values: Any `OlStr` value valid in the current locale.

For a vertical gauge, the label is placed to the right of the minimum value position. If there is not enough space for the entire label and `XtNrecomputeSize` is `TRUE`, then the widget will request more space to show the entire label.

For a horizontal gauge, the label is placed centered and below the minimum value position. If there is not enough room to center the label and `XtNrecomputeSize` is set to `FALSE`, the beginning of the label will be aligned with the left end of the outline of the gauge and is drawn to the right. If this label collides with the maximum label, some part of the labels will overlap. If there is not enough room to center the label and `XtNrecomputeSize` is set to `TRUE`, the widget will request for more space to center the label below the minimum value position.

### ***XtNorientation***

| Class                       | Type                  | Default                  | Access          |
|-----------------------------|-----------------------|--------------------------|-----------------|
| <code>XtCOrientation</code> | <code>OlDefine</code> | <code>OL_VERTICAL</code> | <code>GI</code> |

Synopsis: The direction for the visual presentation of the widget.

*Gauge Widget*

Values: OL\_HORIZONTAL/"horizontal" - Define a horizontal gauge.  
 OL\_VERTICAL/"vertical" - Define a vertical gauge.

***XtNrecomputeSize***

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCRecomputeSize | Boolean | FALSE   | SGI    |

Synopsis: The widget's resize policy.  
 Values: TRUE/"true" - The widget will resize itself whenever needed, to compensate for the space needed to show the tick marks and the labels.  
 FALSE/"false" - The widget will not resize itself.

The gauge widget uses XtNspan, the sizes of the labels, and XtNtickUnit to determine the preferred size.

***XtNrightMargin***

| Class     | Type      | Default   | Access |
|-----------|-----------|-----------|--------|
| XtCMargin | Dimension | OL_IGNORE | SGI    |

Synopsis: The number of pixels in the margin to the right of the gauge.  
 Values: OL\_IGNORE/"ignore" or any valid Dimension.

***XtNsliderMax/  
XtNsliderMin***

| Class        | Type | Default | Access |
|--------------|------|---------|--------|
| XtCSliderMax | int  | 100     | SGI    |
| XtCSliderMin | int  | 0       | SGI    |

Synopsis: The range of values tracked by the Gauge widget.  
 Values: XtNsliderMin < XtNsliderMax

***XtNsliderValue***

| Class          | Type | Default | Access |
|----------------|------|---------|--------|
| XtCSliderValue | int  | 0       | SGI    |

Synopsis: The values represented by the current position of the end of the shaded portion of the gauge.  
 Values: XtNsliderMin ≤ XtNsliderValue ≤ XtNsliderMax

***XtNspan***

| Class   | Type      | Default   | Access |
|---------|-----------|-----------|--------|
| XtCspan | Dimension | OL_IGNORE | SGI    |

Synopsis: If `XtNrecomputeSize` is set to `TRUE`, the preferred length of the gauge, not counting the space needed for the labels.

Values: `OL_IGNORE`/"ignore" or any valid `Dimension`.

The gauge widget uses the span value, the sizes of the labels, and `XtNtickUnit` to determine the preferred size.

***XtNticks***

| Class    | Type | Default | Access |
|----------|------|---------|--------|
| XtCTicks | int  | 0       | SGI    |

Synopsis: The interval between tick marks.

Values: The unit of the interval value is determined by `XtNtickUnit`.

***XtNtickUnit***

| Class       | Type     | Default | Access |
|-------------|----------|---------|--------|
| XtCTickUnit | OlDefine | OL_NONE | SGI    |

Synopsis: The interpretation of the `XtNticks` resource.

Values: `OL_NONE`/"none" - Display no tick marks and ignore `XtNticks`.  
`OL_PERCENT`/"percent" - Interpret `XtNticks` as the percentage of the gauge value range.  
`OL_SLIDERVALUE`/"slidervalue" - Interpret `XtNticks` in gauge value units.

To be consistent with the Scrollbar widget, the effective spacing between tick marks, designated in `XtNticks` and `XtNtickUnit`, should be less than or equal to the spacing in `XtNgranularity`.

***Activation Types***

The following table lists the activation types used by the Gauge.

Table 8-4 Gauge Activation Types

| Activation Type               | Semantics     | Resource Name                    |
|-------------------------------|---------------|----------------------------------|
| <code>OL_CANCEL</code>        | CANCEL        | <code>XtNcancelKey</code>        |
| <code>OL_DEFAULTACTION</code> | DEFAULTACTION | <code>XtNdefaultActionKey</code> |

*Gauge Function*

*Table 8-4 Gauge Activation Types (Continued)*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The Gauge widget is a read-only control that has no activation types besides the ones in “Common Activation Types” on page 68.

*See Also*

“Slider Widget” on page 586.

*Gauge Function*

The following convenience function is used to set gauge values.

*OlSetGaugeValue*

```
#include <Xol/Gauge.h>
extern void OlSetGaugeValue(
 Widget w,
 int value);
```

This function is an alternative and faster method of setting the current value of a Gauge widget. The effect is equivalent to doing XtSetValues() on the XtNsliderValue resource of the widget.

## MenuButton Widget

### Class

**Class Name:** MenuButton  
**Class Pointer:** menuButtonWidgetClass, menuButtonGadgetClass

### Ancestry

Core-Primitive-Button-MenuButton

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/MenuButton.h>
```

### Description

The MenuButton is used to create a popup menu. It appears similar to the OblongButton widget (see OblongButton Widget on page 464), except that it also has a mark, called a menumark, near the right end of the button. When the user invokes the MENU command on the MenuButton, a menu pops up. The MenuButton provides the features of menu default selection and menu previewing as well as the features of the MenuShell widget.

### Components

The MenuButton consists of an oblong border containing a label and a menumark. A popup menu is attached. Each MenuButton also has the components of the MenuShell widget.

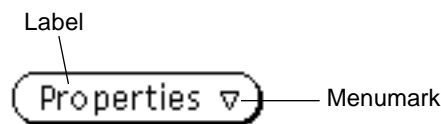


Figure 8-2 MenuButton Appearance

### Subwidgets

The MenuButton widget automatically creates and attaches a MenuShell widget. An application can add menu items to this menu by obtaining the value of the XtNmenuPane resource and adding children to this widget.

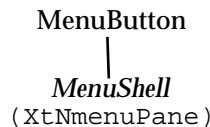


Figure 8-3 MenuButton Subwidget

### Popping Up the MenuShell Subwidget

When the MenuButton widget is not in a menu, pressing or clicking MENU on the MenuButton pops up the MenuButton’s menu in the direction of the menumark. In this case the menumark points down; therefore, the menu will pop up below the MenuButton.

When the MenuButton widget is in a *stay-up* menu (implementing a cascade menu), pressing or clicking MENU when the pointer is within or on the button’s border pops up the button’s menu in the direction of the menumark.

When the MenuButton widget is in a *popup* menu (implementing a cascade menu), moving the pointer into the menumark region pops up the menu in the direction of the menumark. The position is computed when the movement into the menumark region is first detected, but rapid pointer motion and internal delays in popping up the menu may let the pointer wander.

Moving the pointer out of the MenuButton widget, but not directly into the newly popped up menu, causes that menu to be popped down. This occurs even if the pointer is moved into and out of the newly popped up menu in the interim.

### Menu Previewing

The MenuButton widget supports previewing of the default menu item if the toolkit resource XtNselectDoesPreview is set to TRUE. In this case, if SELECT is activated on the MenuButton, the widget will display the label of the default item within the border of the MenuButton while the SELECT mouse

---

## *MenuButton Widget*

button is down. If the user releases the SELECT mouse button inside the MenuButton border, the default menu item will be selected.

### *Menu Placement—Not Enough Space*

If the right or bottom edge of the screen is too close to allow the menu placement described above, the menu pops up aligned with the edge of the screen and the pointer is shifted horizontally to keep it 4 points from the left edge of the menu items. If the left edge of the screen is too close, the menu pops up 4 points from the edge and the pointer is shifted to lie on the edge. The pointer does not jump back after the menu is dismissed.

### *Selecting the Default Item*

Each MenuButton widget has a default item belonging to the MenuShell subwidget. If SELECT is activated on the MenuButton (and the XtNselectDoesPreview toolkit resource is set to TRUE) and the default item is inactive (the XtNsensitive resource is FALSE) or busy (the XtNbusy resource is TRUE), the system beeps.

If SELECT is activated on the MenuButton (and the XtNselectDoesPreview toolkit resource is set to TRUE) and the default item is another MenuButton (for a cascading menu), its default will be selected; this recurses through the menu tree until a non-MenuButton default widget is found.

### *Coloration*

For 3D, MenuButton coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.” XtNbackground is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1. XtNfontColor is used to draw the label.

For 2D, XtNbackground and XtNfontColor are used to render the MenuButton as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, “Controls.”

If the toolkit resource XtNmouseless is set to TRUE and the toolkit resource XtNinputFocusFeedback is set to OL\_INPUT\_FOCUS\_COLOR, then the background of the MenuButton will be drawn with the value of

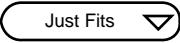
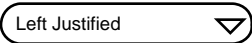
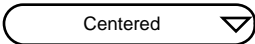

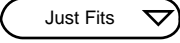
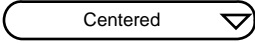
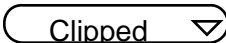
*MenuButton Widget*

XtNinputFocusColor when the widget receives input focus. However, if XtNinputFocusColor is the same as XtNbackground, then the MenuButton inverts XtNfontColor and XtNbackground. Once the input focus leaves the widget, the original coloration is restored.

**Label Appearance**

The XtNwidth, XtNheight, XtNrecomputeSize, and XtNlabelJustify resources interact to produce a truncated, clipped, centered, or left-justified label as shown in the following table.

Table 8-5 MenuButton Label Appearance

| XtNwidth           | XtNrecomputeSize | XtNlabelJustify | Result                                                                                |
|--------------------|------------------|-----------------|---------------------------------------------------------------------------------------|
| any value          | TRUE             | any             |  |
| > needed for label | FALSE            | OL_LEFT         |  |
| > needed for label | FALSE            | OL_CENTER       |  |
| < needed for label | FALSE            | any             |  |
| XtNheight          | XtNrecomputeSize | XtNlabelJustify | Result                                                                                |
| any value          | TRUE             | any             |  |
| > needed for label | FALSE            | any             |  |
| < needed for label | FALSE            | any             |  |

When the label is centered or left-justified, the extra space is filled with the background color of the MenuButton widget, as determined by the XtNbackground and XtNbackgroundPixmap resources. When a text label is truncated, the truncation occurs at a character boundary and a “more arrow” is inserted to show that part of the label is missing. The arrow requires that more of the label be truncated than would otherwise be necessary. If the width of the button is too small to show even one character with the triangle, only the triangle is shown. If the width is so small that the entire triangle cannot be shown, the arrow is clipped on the right.



*Resources**Table 8-6* MenuButton Core Resources

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | (calculated)        | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | (calculated)        | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 8-7* MenuButton Primitive Resources

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|--------------------|----------------|---------------------|---------------|
| XtNaccelerator     | String         | NULL                | n/a           |
| XtNacceleratorText | String         | NULL                | n/a           |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor | Pixel          | Red                 | SGID          |
| XtNmnemonic        | unsigned char  | '\0'                | n/a           |
| XtNreferenceName   | String         | NULL                | GI            |
| XtNreferenceWidget | Widget         | NULL                | GI            |

*MenuButton Widget*

*Table 8-7 MenuButton Primitive Resources (Continued)*

| <b>Name</b>    | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|----------------|-------------|----------------|---------------|
| XtNscale       | int         | 12             | SGI           |
| XtNtextFormat  | OlStrRep    | OL_SB_STR_REP  | GI            |
| XtNtraversalOn | Boolean     | TRUE           | SGI           |
| XtNuserData    | XtPointer   | NULL           | SGI           |

*Table 8-8 MenuButton Resources*

| <b>Name</b>      | <b>Type</b> | <b>Default</b>  | <b>Access</b> |
|------------------|-------------|-----------------|---------------|
| XtNdefault       | Boolean     | FALSE           | SGI           |
| XtNlabel         | OlStr       | (instance name) | SGI           |
| XtNlabelImage    | XImage *    | NULL            | SGI           |
| XtNlabelJustify  | OlDefine    | OL_LEFT         | SGI           |
| XtNlabelType     | OlDefine    | OL_STRING       | SGI           |
| XtNmenuMark      | OlDefine    | (calculated)    | SGI           |
| XtNrecomputeSize | Boolean     | TRUE            | SGI           |

The following table lists the MenuButton resources that are propagated to the MenuShell subwidget.

*Table 8-9 MenuShell Subwidget Resources<sup>1</sup>*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNcenter         | Boolean     | TRUE           | I             |
| XtNhPad           | Dimension   | 6              | I             |
| XtNhSpace         | Dimension   | 6              | I             |
| XtNlayoutType     | OlDefine    | OL_FIXEDCOLS   | I             |
| XtNmeasure        | int         | 1              | I             |
| XtNmenuPane       | Widget      | (special)      | G             |
| XtNpushpin        | OlDefine    | OL_NONE        | I             |
| XtNpushpinDefault | Boolean     | FALSE          | I             |
| XtNsameSize       | OlDefine    | OL_COLUMNS     | I             |
| XtNshellTitle     | OlStr       | (widget name)  | SGI           |
| XtNvPad           | Dimension   | 3              | I             |
| XtNvSpace         | Dimension   | 1              | I             |

1. These subwidget resources are described in the sections “ControlArea Widget” on page 249 and “MenuShell Widget” on page 414.

***XtNdefault***

| Class      | Type    | Default | Access |
|------------|---------|---------|--------|
| XtCDefault | Boolean | FALSE   | SGI    |

Synopsis: Whether the MenuButton is the default choice in its immediate shell.

Values: TRUE/"true" - If the button is in a menu, an oval ring is drawn around the button to show that the button is the default choice of one or more buttons.  
FALSE/"false" - This button is not the default control of the shell.

Setting `XtNdefault` to TRUE has the effect of setting `XtNdefault` of the previous default control for the shell to be FALSE. The `OL_MENUDEFAULT` and `OL_MENUDEFAULTKEY` activation types modify this resource.

***XtNlabel***

| Class    | Type  | Default         | Access |
|----------|-------|-----------------|--------|
| XtCLabel | OlStr | (instance name) | SGI    |

Synopsis: The text for the Label.

Values: Any `OlStr` value valid in the current locale.

This resource is ignored if the `XtNlabelType` resource has the value `OL_IMAGE`.

The MenuButton label is colored using the `XtNfontColor` resource.

***XtNlabelImage***

| Class         | Type     | Default | Access |
|---------------|----------|---------|--------|
| XtCLabelImage | XImage * | NULL    | SGI    |

Synopsis: The image for the Label.

This resource is ignored unless the `XtNlabelType` resource has the value `OL_IMAGE`. If the image is of type `XYBitmap`, the image is highlighted when appropriate by reversing the 0 and 1 values of each pixel (that is, by XORing the image data). If the image is of type `XYPixmap` or `ZPixmap`, the image is not highlighted, although the space around the image inside the border is highlighted.

*MenuButton Widget*

If the image is smaller than the space available for it inside the border and `XtNlabelTile` is FALSE, the image is centered vertically and either centered or left-justified horizontally, depending on the value of the `XtNlabelJustify` resource. If the image is larger than the space available for it, it is clipped so that it does not display outside the border. If the `XtNdefault` resource is TRUE so that the border is doubled, the space available is that inside the inner line of the border.

***XtNlabelJustify***

| Class           | Type     | Default | Access |
|-----------------|----------|---------|--------|
| XtCLabelJustify | OlDefine | OL_LEFT | SGI    |

Synopsis: The justification of a label within the widget.

Values: OL\_LEFT/"left" - Left-justify the label.  
 OL\_CENTER/"center" - Center the label.

***XtNlabelType***

| Class        | Type     | Default   | Access |
|--------------|----------|-----------|--------|
| XtCLabelType | OlDefine | OL_STRING | SGI    |

Synopsis: The form that the label takes.

Values: OL\_STRING/"string" - The label is text.  
 OL\_IMAGE/"image" - The label is an image.

***XtNmenuMark***

| Class       | Type     | Default      | Access |
|-------------|----------|--------------|--------|
| XtCMenuMark | OlDefine | (calculated) | SGI    |

Synopsis: The direction of the menu arrow.

Values: OL\_DOWN/"down" - The menu arrow points down.  
 OL\_RIGHT/"right" - The menu arrow points to the right.

The default is OL\_RIGHT if the immediate shell ancestor is a MenuShell or a subclass thereof; otherwise, it is OL\_DOWN.

**XtNmenuPane**

| Class       | Type   | Default | Access |
|-------------|--------|---------|--------|
| XtCMenuPane | Widget | NULL    | G      |

Synopsis: The widget where menu items can be added.

Values: ID of the menupane widget contained in the MenuButton's MenuShell.

This resource is available once the MenuButton widget has been created.

**XtNrecomputeSize**

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCRecomputeSize | Boolean | TRUE    | SGI    |

Synopsis: The resize policy of the widget.

Values: TRUE/"true" - The MenuButton widget will do normal size calculations that may cause its geometry to change and automatically set the XtNheight and XtNwidth resources.  
FALSE/"false" - The MenuButton widget will leave its size alone; this may cause truncation of the visible image being shown by the MenuButton widget if the fixed size is too small, or may cause padding if the fixed size is too large. The location of the padding is determined by the XtNlabelJustify resource.

**Activation Types**

The following table lists the activation types used by the MenuButton.

Table 8-10 MenuButton Activation Types

| Activation Type   | Semantics     | Resource Name       |
|-------------------|---------------|---------------------|
| OL_CANCEL         | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION  | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP           | HELP          | XtNhelpKey          |
| OL_MENU           | MENU          | XtNmenuBtn          |
| OL_MENUDEFAULT    | MENUDEFAULT   | XtNmenuDefaultBtn   |
| OL_MENUDEFAULTKEY | MENUDEFAULT   | XtNmenuDefaultKey   |
| OL_MENUKEY        | MENU          | XtNmenuKey          |
| OL_MOVEDOWN       | DOWN          | XtNdownKey          |
| OL_MOVELEFT       | MOVELEFT      | XtNleftKey          |

*MenuButton Widget*

*Table 8-10 MenuButton Activation Types (Continued)*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | UP               | XtNupKey             |
| OL_MULTIDOWN           | JUMP DOWN        | XtNmultiDownKey      |
| OL_MULTILEFT           | JUMP LEFT        | XtNmultiLeftKey      |
| OL_MULTIRIGHT          | JUMP RIGHT       | XtNmultiRightKey     |
| OL_MULTIUP             | JUMP UP          | XtNmultiUpKey        |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_NEXTWINDOW          | NEXTWINDOW       | XtNnextWinKey        |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_PREVWINDOW          | PREVWINDOW       | XtNprevWinKey        |
| OL_SELECT              | SELECT           | XtNselectBtn         |
| OL_SELECTKEY           | SELECT           | XtNselectKey         |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

***OL\_MENU***

The OL\_MENU activation type can be used to pop up the menu in two different modes: press-drag-release and click-move-click. These modes are described in the *OPEN LOOK GUI Functional Specification* section “Using Menu Buttons” in Chapter 15. The position of the menu depends on the space available on the screen and is described in the *OPEN LOOK GUI Functional Specification* section “Menu Placement” in Chapter 15.

***OL\_MENUDEFAULT/  
OL\_MENUDEFAULTKEY***

The OL\_MENUDEFAULT and OL\_MENUDEFAULTKEY activation types apply only to MenuButtons that are descendants of a Menu. These activation types will set the MenuButton `xtNdefault` resource to TRUE, and change the display of the widget according to the *OPEN LOOK GUI Functional Specification* section “Changing Menu Defaults” in Chapter 15.

***OL\_MENUKEY***

The `OL_MENUKEY` activation type can be used to pop up the menu according to the *OPEN LOOK Mouseless Specification* section 4.2.

***OL\_SELECT***

The activation of the `MenuButton` widget with the `SELECT` button depends on the value of the toolkit resource `XtNselectDoesPreview`. When the resource `XtNselectDoesPreview` is `FALSE`, this activation type will behave exactly as the `OL_MENU` activation type. When `XtNselectDoesPreview` is `TRUE`, `SELECT` can be used as a shortcut to display and activate the menu default as described in the *OPEN LOOK GUI Functional Specification* Chapter 15 and Chapter 5. If `SELECT` is released within the `MenuButton`, the default menu item will be activated with the `OL_SELECTKEY` activation type.

***OL\_SELECTKEY***

The `OL_SELECTKEY` activation type can be used to pop up the menu according to the *OPEN LOOK Mouseless Specification* section 4.2.

***See Also***

“`AbbrevMenuButton` Widget” on page 217,  
“`ControlArea` Widget” on page 249  
“`MenuShell` Widget” on page 414.

---

*MenuShell Widget**MenuShell Widget**Class*

*Class Name:* MenuShell  
*Class Pointer:* menuShellWidgetClass

*Ancestry*

Core-Composite-Shell-WMShell-VendorShell-TransientShell-MenuShell

*Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/Menu.h>
```

*Description*

The MenuShell widget is used to create a menu not associated with either a MenuButton or an AbbrevMenuButton. For example, a MenuShell widget can be attached to a button, such as an OblongButton widget, but this does not make the button into a menu button. However, all the features of the MenuShell widget (except those related to menu creation) also pertain to the MenuButton menu.

*Components*

A menu contains a set of items that are presented to the user for selection. These are specified by the application as widgets attached to the menu. One of these items is a default item. (A menu always has exactly one default item.) The items are laid out in a control area. A menu also has a title, a title separator, a border or window border, a drop shadow, and an optional pushpin. The application chooses the label for the Title and whether a menu has a pushpin.

A popup or stay-up menu shows the title, border, pushpin (if available), items, and drop shadow. The title is left out if the menu is from either a menu button or an abbreviated menu button. A pinned menu shows the window border, title, pushpin, items, but no drop shadow.



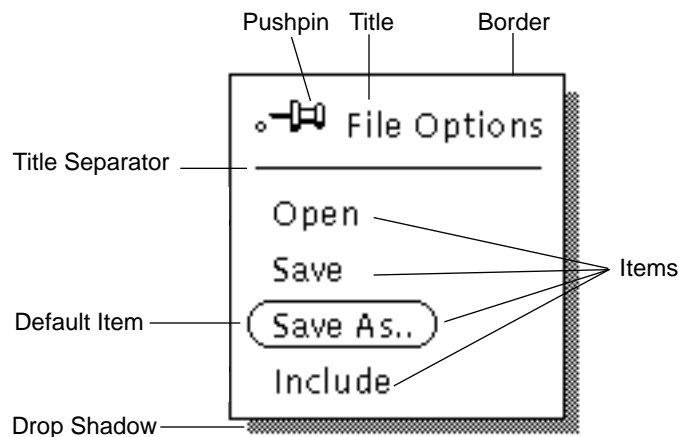


Figure 8-4 MenuShell Menu Components

### Subwidgets

The MenuShell contains one subwidget, a “menupane,” which is a container for the components of the menu. This widget is provided automatically and is accessible through the `XtNmenuPane` resource.

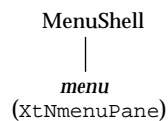


Figure 8-5 MenuShell Subwidgets

### Subclass of Shell Widget

The MenuShell widget is a subclass of Shell widget, so the `XtCreatePopupShell()` routine is used to create it instead of the more common `XtCreateWidget()`.

### Menupane

The menupane is not described as a separate widget here; the only interface to it for the application programmer is as a parent widget to which the widgets comprising the menu items are attached. The menu items are not attached directly to the MenuShell widget, since a shell widget can take only one child.

***Associating a Menu with a Widget***

A menu can be associated with any widget, including Primitive widgets. The connection is made by creating the menu widget as a child of the other widget. Being a shell widget, the MenuShell widget is not a normal widget-child of its parent, but a popup child. If the application allows it, the menu augments the parent's event list so that the popping up of the menu is handled automatically.

***Popup Control***

Pressing MENU when the pointer is over the parent of the MenuShell widget causes the menu to be popped up. The menu is presented as a popup menu, where the items are available for a *press-drag-release* type of selection (see below).

Clicking MENU when the pointer is over the parent of the MenuShell widget also causes the menu to be popped up, but the menu is presented as a stay-up menu, where the items are available for a *click-move-click* type of selection, instead (see below).

A "slow click" (a press with a delay before the release) may show the menu as a popup on the press, then as a stay-up on the release.

***Selection Control***

The MenuShell arranges for its children to respond to either the press-drag-release or the click-move-click type of selection.

With the press-drag-release type of control, the user can keep MENU pressed and move the pointer to the item of choice; releasing MENU selects the item and pops the menu down. If the pointer is not over an item when MENU is released, the menu simply pops down.

With the click-move-click type of control, the user can move the pointer to the item of choice (MENU has already been released to end a click); clicking SELECT or MENU selects the item and pops the menu down. If the pointer is not over an item when SELECT or MENU is clicked, the menu simply pops down.

These selection methods apply to all menu items *except* menu buttons. For example, in Figure 8-4 on page 415, `Locate Owner` can be selected using the methods described here. For the other items in the figure (which are menu

buttons), see “MenuButton Widget” on page 403 for the explanation of menu button selection behavior.

### ***Converting Stay-up to Popup Menu***

Pressing MENU in a stay-up menu converts it to a popup menu. Thus, the click-move-click selection control becomes a press-drag-release selection control.

### ***Highlighting Items***

In the press-drag-release type of selection control, each menu item highlights while the pointer is over it. The form of the highlighting depends on the type of widget making up the item. Again, the MenuShell widget arranges for its children to respond in this way. No highlighting occurs when the click-move-click type of selection control is used.

### ***Use of the Pushpin***

The pushpin is presented to the user like any of the items to be selected from the menu, except that it is always the topmost item, and it is presented visually as an “adornment” of the header, next to the title (if present). The user can select the pushpin, pushing it in to cause the menu to remain on the display as a popup window, or a *pinned menu*, and pulling it out to make the menu a stay-up menu. To the user, a pinned menu behaves indistinguishably from a command window.

### ***Default Item***

If none of the menu items is explicitly set as the default item, the menu picks the first menu item to be the default item. If the menu contains a pushpin and no other menu item is explicitly set as the default item, the pushpin is chosen as the default item.

### ***Popup Position***

If the menu is not from a menu button, the menu pops up so that the default item is vertically centered 4 points to the right of the pointer. If the right or bottom edge of the screen is too close to allow this placement, the menu pops up with its edge aligned to the edge of the screen, and the mouse pointer is shifted horizontally to keep it 4 points from the left edge of the default item.

For the popup position when the menu is from a menu button, see “MenuButton Widget” on page 403.

### Coloration

The following diagram illustrates the resources that affect MenuShell coloration in 2D.

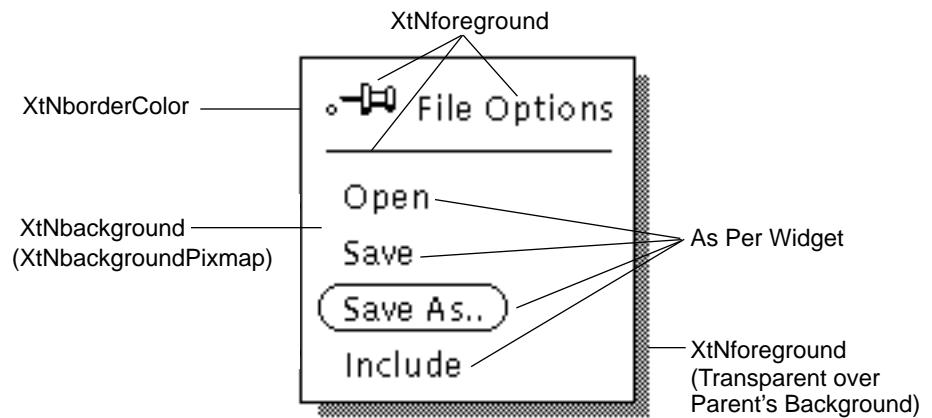


Figure 8-6 MenuShell Menu Coloration

Coloration is identical for 3D, except that the coloration of the pushpin is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1 of the pushpin, and the BG2 (shaded), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

### Programmatic Menu Popup and Popdown

Four convenience routines are provided to programmatically control the mapping and unmapping of menus.

#### *OlMenuPopup*

```
void OlMenuPopup(
 Widget menu,
 Widget emanate,
 Cardinal item_index,
 OlDefine state,
 Boolean set_position,
 Position x,
 Position y,
 OlMenuPositionProc position_proc);
```

*MenuShell Widget*

|                      |                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>menu</i>          | A menuShellWidget ID obtained by creating a menu explicitly.                                                                                                                                                                                                                                                                                                                       |
| <i>emanate</i>       | The object that the menu is currently associated with; it is supplied to the <i>position_proc</i> when the menu positioning is done. If this field is NULL, the menu's parent object is used as the emanate object for later positioning calculations.                                                                                                                             |
| <i>item_index</i>    | If <i>emanate</i> is a flattened widget, this parameter specifies the particular item.                                                                                                                                                                                                                                                                                             |
| <i>state</i>         | The state the menu should be in when it is visible on the screen: one of: OL_PINNED_MENU, OL_PRESS_DRAG_MENU, or OL_STAYUP_MENU.                                                                                                                                                                                                                                                   |
| <i>set_position</i>  | A flag indicating whether the following two arguments ( <i>x</i> and <i>y</i> ) are used to help position the menu. If the flag is FALSE, the current Pointer Location is used to initialize <i>x</i> and <i>y</i> .                                                                                                                                                               |
| <i>x, y</i>          | These coordinates are used by the positioning routine. Typically, these values represent the pointer position with respect to the RootWindow (e.g., <code>xevent-&gt;xbutton.x_root</code> and <code>xevent-&gt;xbutton.y_root</code> ). However, if the menu's state is OL_PINNED_MENU, these coordinates represent the desired upper-left hand corner of the pinned menu.        |
| <i>position_proc</i> | The procedure called to determine the menu's position if the menu's state is either OL_PINNED_MENU or OL_STAYUP_MENU. If the menu's state is OL_PINNED_MENU, the <i>position_proc</i> value is ignored. If this procedure is NULL, the default positioning routine (i.e., the one associated with the emanate widget or the menu's parent) is used. The type of this procedure is: |

```
typedef void (*OlMenuPositionProc)(
 Widget menu,
 Widget emanate,
 Cardinal item_index,
 OlDefine state,
 Position *mx,
 Position *my,
 Position *px,
 Position *py);
```

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <i>menu</i>       | A menuShellWidget ID obtained by creating a menu explicitly.       |
| <i>emanate</i>    | Menu's emanate widget.                                             |
| <i>item_index</i> | Emanate <i>item_index</i> or OL_NO_ITEM                            |
| <i>state</i>      | The state of the menu; either OL_PRESS_DRAG_MENU or OL_STAYUP_MENU |

MenuShell Widget

*mx, my* Pointers containing the menu's current x and y locations. If the position routine wants to move the menu, it should change these values. The position routine should not move the menu explicitly.

*px, py* The x and y locations supplied to the `OlMenuPopup()` routine. If the position routine changes these values, the pointer is warped to the new location.

***OlMenuPost***

```
void OlMenuPost(
 Widget menu);
```

This convenience routine is equivalent to:

```
OlMenuPopup(menu, NULL, OL_NO_ITEM, OL_PRESS_DRAG_MENU, FALSE,
 0, 0, (OlMenuPositionProc) NULL);
```

***OlMenuPopdown***

```
void OlMenuPopdown(
 Widget menu,
 Boolean dismiss_pinned);
```

This routine pops down a menu. If a menu is pinned, a value of TRUE for *dismiss\_pinned* is required to dismiss it. If a menu does not have a pushpin or the menu is not pinned, the *dismiss\_pinned* field is ignored.

***OlMenuUnpost***

```
void OlMenuUnpost(
 Widget menu);
```

This convenience routine is equivalent to:

```
OlMenuPopdown(menu, FALSE);
```

***Avoiding Permanent Toolkit Grabs***

To avoid permanent toolkit grabs, the application should add the `OlCallbackPopdownMenu` callback procedure to the `XtNpopupCallback` callback list of any shell (except `NoticeShell` and `MenuShell`) that may be popped up with an exclusive toolkit grab by the application.

```
void OlCallbackPopdownMenu(
 Widget w,
 XtPointer client_data,
 XtPointer call_data)
```

The arguments are:

|                    |                            |
|--------------------|----------------------------|
| <i>w</i>           | Specifies the widget       |
| <i>client_data</i> | Not used by this procedure |
| <i>call_data</i>   | Not used by this procedure |

### ***Keyboard Traversal***

By default, all menus allow traversal among the traversable controls added to the widget.

Popping up a Menu via the keyboard is done by traversing to a `MenuButton`, using `NEXTFIELD`, `PREVFIELD`, `MOVEUP`, `MOVEDOWN`, `MOVERIGHT`, or `LEFT`, and pressing the `MENUKEY` key. If a menu is attached to a control other than a `MenuButton`, it can be popped up by traversing to that control and pressing the `MENUKEY`.

Keyboard traversal within a menu is done using the `PREVFIELD`, `NEXTFIELD`, `MOVEUP`, `MOVEDOWN`, `MOVELEFT` and `MOVERIGHT` keys. The `PREVFIELD`, `MOVEUP`, and `MOVELEFT` keys move the input focus to the previous Menu item with keyboard traversal enabled. If the input focus is on the first item of the Menu, then pressing one of these keys will wrap to the last item of the Menu with keyboard traversal enabled. The `NEXTFIELD`, `MOVEDOWN`, and `MOVERIGHT` keys move the input focus to the next Menu item with keyboard traversal enabled. If the input focus is on the last item of the Menu, then pressing one of these keys will wrap to the first item of the Menu with keyboard traversal enabled.

To traverse out of the menu, the following keys can be used:

- `CANCEL` dismisses the menu and returns focus to the originating control
- `NEXTWINDOW` moves to the next window in the application
- `PREVWINDOW` moves to the previous window in the application
- `NEXTAPP` moves to the first window in the next application
- `PREVAPP` moves to the first window in the previous application

`NEXTWINDOW`, `PREVWINDOW`, `NEXTAPP`, and `PREVAPP` are provided by the `OLWM` window manager and are set via different resources.

MenuShell Widget

Resources

Table 8-11 MenuShell Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | GI     |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | 0                   | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

Table 8-12 MenuShell Composite Resources

| Name              | Type        | Default | Access |
|-------------------|-------------|---------|--------|
| XtNchildren       | WidgetList  | NULL    | G      |
| XtNinsertPosition | XtOrderProc | NULL    | SGI    |
| XtNnumChildren    | Cardinal    | 0       | G      |

Table 8-13 MenuShell Shell Resources

| Name                    | Type                   | Default | Access |
|-------------------------|------------------------|---------|--------|
| XtNallowShellResize     | Boolean                | TRUE    | SGI    |
| XtNcreatePopupChildProc | XtCreatePopupChildProc | NULL    | SGI    |



Table 8-13 MenuShell Shell Resources (Continued)

| Name                | Type           | Default    | Access |
|---------------------|----------------|------------|--------|
| XtNgeometry         | String         | NULL       | GI     |
| XtNoverrideRedirect | Boolean        | FALSE      | SGI    |
| XtNpopupCallback    | XtCallbackList | NULL       | SGIO   |
| XtNpopupCallback    | XtCallbackList | NULL       | SGIO   |
| XtNsaveUnder        | Boolean        | TRUE       | SGI    |
| XtNvisual           | Visual *       | (parent's) | GIO    |

Table 8-14 MenuShell WMSHELL Resources

| Name             | Type         | Default               | Access |
|------------------|--------------|-----------------------|--------|
| XtNbaseHeight    | int          | XtUnspecifiedShellInt | SGI    |
| XtNbaseWidth     | int          | XtUnspecifiedShellInt | SGI    |
| XtNheightInc     | int          | XtUnspecifiedShellInt | SGI    |
| XtNiconMask      | Pixmap       | NULL                  | SGI    |
| XtNiconPixmap    | Pixmap       | NULL                  | SGI    |
| XtNiconWindow    | Window       | NULL                  | SGI    |
| XtNiconX         | int          | XtUnspecifiedShellInt | SGI    |
| XtNiconY         | int          | XtUnspecifiedShellInt | SGI    |
| XtNinitialState  | InitialState | NormalState           | SGI    |
| XtNinput         | Bool         | FALSE                 | G      |
| XtNmaxAspectX    | int          | XtUnspecifiedShellInt | SGI    |
| XtNmaxAspectY    | int          | XtUnspecifiedShellInt | SGI    |
| XtNmaxHeight     | int          | OL_IGNORE             | SGI    |
| XtNmaxWidth      | int          | OL_IGNORE             | SGI    |
| XtNminAspectX    | int          | XtUnspecifiedShellInt | SGI    |
| XtNminAspectY    | int          | XtUnspecifiedShellInt | SGI    |
| XtNminHeight     | int          | OL_IGNORE             | SGI    |
| XtNminWidth      | int          | OL_IGNORE             | SGI    |
| XtNtitle         | String       | NULL                  | SGI    |
| XtNtitleEncoding | Atom         | XA_STRING             | SGI    |
| XtNtransient     | Boolean      | TRUE                  | SGI    |
| XtNwaitForWm     | Boolean      | TRUE                  | SGI    |
| XtNwidthInc      | int          | XtUnspecifiedShellInt | SGI    |
| XtNwindowGroup   | Window       | XtUnspecifiedWindow   | SGI    |

MenuShell Widget

Table 8-14 MenuShell WMShell Resources (Continued)

| Name          | Type | Default               | Access |
|---------------|------|-----------------------|--------|
| XtNwinGravity | int  | XtUnspecifiedShellInt | SGI    |
| XtNwmTimeout  | int  | 5000 (msec)           | SGI    |

Table 8-15 MenuShell VendorShell Resources

| Name                    | Type           | Default                                | Access |
|-------------------------|----------------|----------------------------------------|--------|
| XtNbusy                 | Boolean        | FALSE                                  | SGI    |
| XtNconsumeEvent         | XtCallbackList | NULL                                   | SGIO   |
| XtNdefaultImName        | String         | NULL                                   | SGI    |
| XtNfooterPresent        | Boolean        | FALSE                                  | SGI    |
| XtNfocusWidget          | Widget         | (see description)                      | SGI    |
| XtNimFontSet            | OIFont         | XtDefaultFontSet                       | SGI    |
| XtNimStatusStyle        | OImStatusStyle | OL_NO_STATUS                           | GI     |
| XtNleftFooterString     | OIStr          | NULL                                   | SGI    |
| XtNleftFooterVisible    | Boolean        | TRUE                                   | SGI    |
| XtNmenuButton           | Boolean        | (see description)                      | GI     |
| XtNmenuType             | OIDefine       | (see description)                      | SGI    |
| XtNpushpin              | OIDefine       | (see description)                      | SGI    |
| XtNresizeCorners        | Boolean        | (see description)                      | SGI    |
| XtNrightFooterString    | OIStr          | NULL                                   | SGI    |
| XtNrightFooterVisible   | Boolean        | TRUE                                   | SGI    |
| XtNshellTitle           | OIStr          | NULL                                   | SGI    |
| XtNuserData             | XtPointer      | NULL                                   | SGI    |
| XtNwindowHeader         | Boolean        | (see description)                      | GI     |
| XtNwmProtocol           | XtCallbackList | NULL                                   | SGIO   |
| XtNwmProtocolInterested | int            | OL_WM_DELETE_WINDOW   OL_WM_TAKE_FOCUS | I      |

Table 8-16 MenuShell TransientShell Resources

| Name            | Type   | Default | Access |
|-----------------|--------|---------|--------|
| XtNtransientFor | Widget | NULL    | SGI    |

Table 8-17 MenuShell Resources

| Name              | Type     | Default | Access |
|-------------------|----------|---------|--------|
| XtNmenuAugment    | Boolean  | TRUE    | GI     |
| XtNmenuPane       | Widget   | NULL    | G      |
| XtNpushpin        | OlDefine | OL_NONE | GI     |
| XtNpushpinDefault | Boolean  | FALSE   | GI     |

The MenuPane subwidget attached to the MenuShell widget is constrained by the following resources. These resources can be set and read just like any other resources for the MenuShell.

Table 8-18 MenuPane Subwidget Resources<sup>1</sup>

| Name          | Type      | Default      | Access |
|---------------|-----------|--------------|--------|
| XtNcenter     | Boolean   | TRUE         | I      |
| XtNhPad       | Dimension | 6            | I      |
| XtNhSpace     | Dimension | 6            | I      |
| XtNlayoutType | OlDefine  | OL_FIXEDCOLS | I      |
| XtNmeasure    | int       | 1            | I      |
| XtNsameSize   | OlDefine  | OL_COLUMNS   | I      |
| XtNvPad       | Dimension | 3            | I      |
| XtNvSpace     | Dimension | 1            | I      |

1. These subwidget resources are described in the section "ControlArea Widget" on page 249.

### ***XtNmenuAugment***

| Class          | Type    | Default | Access |
|----------------|---------|---------|--------|
| XtCMenuAugment | Boolean | TRUE    | GI     |

**Synopsis:** The one responsible for popping up the menu.

**Values:** TRUE/"true" - The MenuShell widget augments its parent's event handling so that the pressing or clicking of MENU automatically pops up the menu.

FALSE/"false" - The application is responsible for detecting when the menu should be popped up, and for calling the routine `OlMenuPost(menu_widget)` to pop up the menu. See page 420.

*MenuShell Widget*

***XtNmenuPane***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCMenuPane  | Widget      | NULL           | G             |

Synopsis: The container widget that will be the parent of the menu items.

The value of this resource is available after the MenuShell widget has been created.

***XtNpushpin***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCPushpin   | OlDefine    | OL_NONE        | GI            |

Synopsis: Whether the MenuShell widget has a pushpin.

Values: OL\_NONE/"none" - No pushpin will be included in the list of menu items.  
 OL\_OUT/"out" - A pushpin will be included as an item the user can select; if the user selects the pushpin, the menu will be made into an OPEN LOOK window. Note that the pushpin item is always at the top of the menu list.

Unlike other widgets, the value OL\_IN is not allowed for the MenuShell widget.

***XtNpushpinDefault***

| <b>Class</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtCPushpinDefault | Boolean     | FALSE          | SGI           |

Synopsis: Whether the pushpin is the default item.

Values: TRUE/"true" - The pushpin is the default item.  
 FALSE/"false" - The pushpin is not the default item.

If a menu has a pushpin and none of the menupane items has been designated as the default, the pushpin automatically becomes the menu's default item and the value of the resource will be updated to reflect this.

## Activation Types

The following table lists the activation types used by the MenuShell.

Table 8-19 MenuShell Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVERIGHT     | RIGHT         | XtNrightKey         |
| OL_MOVEUP        | UP            | XtNupKey            |
| OL_MULTIDOWN     | JUMP DOWN     | XtNmultiDownKey     |
| OL_MULTILEFT     | JUMP LEFT     | XtNmultiLeftKey     |
| OL_MULTIRIGHT    | JUMP RIGHT    | XtNmultiRightKey    |
| OL_MULTIUP       | JUMP UP       | XtNmultiUpKey       |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_NEXTWINDOW    | NEXTWINDOW    | XtNnextWinKey       |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_PREVWINDOW    | PREVWINDOW    | XtNprevWinKey       |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

The MenuShell widget has no activation types besides the ones in “Common Activation Types” on page 68.

### See Also

“ControlArea Widget” on page 249,  
“MenuButton Widget” on page 403.

*Nonexclusives Widget*

*Nonexclusives Widget*

*Class*

*Class Name:* Nonexclusives  
*Class Pointer:* nonexclusivesWidgetClass

*Ancestry*

Core-Composite-Constraint-Manager-Nonexclusives

*Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/Nonexclusi.h>
```

*Description*

The Nonexclusives widget provides layout management and selection control for a set of rectangular buttons or check boxes. It provides a simple way to build a several-of-many button selection object.

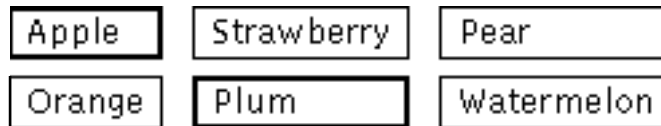


Figure 8-7 Nonexclusive Buttons Example

*Layout and Labels*

The Nonexclusives widget lays out the rectangular buttons or check boxes in a grid in the order they are added as child widgets by the application. The number of rows or columns in this grid can be controlled by the application. If the grid has more than one row, the Nonexclusives widget forces the rectangular buttons or check boxes in each column to be the same size as the widest in the column.

If the grid is a single row, each button will be only as wide as necessary to display the label.

---

## *Nonexclusives Widget*

The rectangular buttons or check boxes are separated by a distance that is 50% of the prevailing point size for the containing window.

### *Use in Menu*

The Nonexclusives widget can be added as a single child to a menupane to implement a several-of-many menu choice. Only `RectButton` widgets can be used in a Nonexclusives widget in a menu.

### *Restrictions on Children*

Child widgets of a Nonexclusives widget are required to be all from the same class--from either `rectButtonWidgetClass` or `checkBoxWidgetClass`.

### *Coloration*

Any space not colored by the Nonexclusives children will be colored with the value of the `XtNbackground` or `XtNbackgroundPixmap` resource.

### *Keyboard Traversal*

The Nonexclusives widget manages the traversal between a set of `RectButtons`. When the user traverses to a Nonexclusives widget, the first `RectButton` in the set will receive input focus. The `MOVEUP`, `MOVEDOWN`, `MOVERIGHT`, and `MOVELEFT` keys move the input focus between the `RectButtons`. To traverse out of the Nonexclusives widget, the following keys can be used:

- `NEXTFIELD` moves to the next traversable widget in the window
- `PREVFIELD` moves to the previous traversable widget in the window
- `NEXTWINDOW` moves to the next window in the application
- `PREVWINDOW` moves to the previous window in the application
- `NEXTAPP` moves to the first window in the next application
- `PREVAPP` moves to the first window in the previous application

These controls have two states: “set” and “not set.” Pressing the `SELECTKEY` on a nonexclusive control will toggle the current state. If the control is in a menu, then the `MENUKEY` will also toggle the current state. If the control is “set,” then toggling the control will call the `XtNunselect` callback list. If the control is “not set,” then toggling the control will call the `XtNselect` callback list.

*Nonexclusives Widget Resources*

*Table 8-20 Nonexclusives Core Resources*

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | 0                   | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 8-21 Nonexclusives Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 8-22 Nonexclusives Manager Resources*

| <b>Name</b>        | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|--------------------|----------------|----------------|---------------|
| XtNconsumeEvent    | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor | Pixel          | Red            | SGID          |



*Nonexclusives Widget**Table 8-22* Nonexclusives Manager Resources (Continued)

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 8-23* Nonexclusives Resources

| <b>Name</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtNlayoutType    | OlDefine    | OL_FIXEDROWS   | SGI           |
| XtNmeasure       | int         | 1              | SGI           |
| XtNrecomputeSize | Boolean     | TRUE           | SGI           |

***XtNlayoutType***

| <b>Class</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtCLayoutType | OlDefine    | OL_FIXEDROWS   | SGI           |

Synopsis: The type of layout of the child widgets.

Values: OL\_FIXEDROWS/"fixedrows" - Set a fixed number of rows.  
 OL\_FIXEDCOLS/"fixedcols" - Set a fixed number of columns.

The choices are to specify the number of rows or the number of columns. Only one of these dimensions can be specified directly; the other is determined by the number of child widgets added, and will always be enough to show all the child widgets.

***XtNmeasure***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCMeasure   | int         | 1              | SGI           |

Synopsis: The number of rows or columns in the layout of the child widgets.

Values: 0 < XtNmeasure

If there are not enough child widgets to fill a row or column, the remaining space is left empty.

*Nonexclusives Widget*

***XtNrecomputeSize***

| <b>Class</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtCRecomputeSize | Boolean     | TRUE           | SGI           |

Synopsis: The resize policy of the widget.

Values: TRUE/"true" - The widget resizes itself to accommodate changes in its children's sizes due to changes in resources such as fonts or labels.  
 FALSE/"false" - The widget does not resize itself.

***Activation Types***

The following table lists the activation types used by the Nonexclusives.

*Table 8-24* Nonexclusives Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | LEFT             | XtNleftKey           |
| OL_MOVERIGHT           | RIGHT            | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_NEXTWINDOW          | NEXTWINDOW       | XtNnextWinKey        |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The Nonexclusives widget has no activation types besides the ones in "Common Activation Types" on page 68.

***See Also***

"Exclusives Widget" on page 277,  
 "FlatNonexclusives Widget" on page 347.

## *NoticeShell Widget*

### *Class*

*Class Name:* NoticeShell  
*Class Pointer:* noticeShellWidgetClass

### *Ancestry*

Core-Composite-Shell-WMShell-VendorShell-TransientShell-NoticeShell

### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/Notice.h>
```

### *Description*

The NoticeShell widget creates a popup window to notify the user of some condition. It is typically used to allow the user to confirm a choice, or to warn the user of a problem. When a NoticeShell is popped up, all other interaction with the application is precluded until the NoticeShell is popped down.

### *Components*

The NoticeShell widget has four main components:

- Text Area – Where the message to the user is displayed
- Control Area – A container for one or more widgets that the user uses to control how to continue with an application
- Default Button – A button to acknowledge notice or to confirm action
- Emanate Widget – Typically the control requiring immediate attention that was activated by the user. (Emanate is not a named widget class. It is named this because it is the widget from which the NoticeShell “emanates,” or pops up.) The application identifies the emanate widget to the NoticeShell widget.

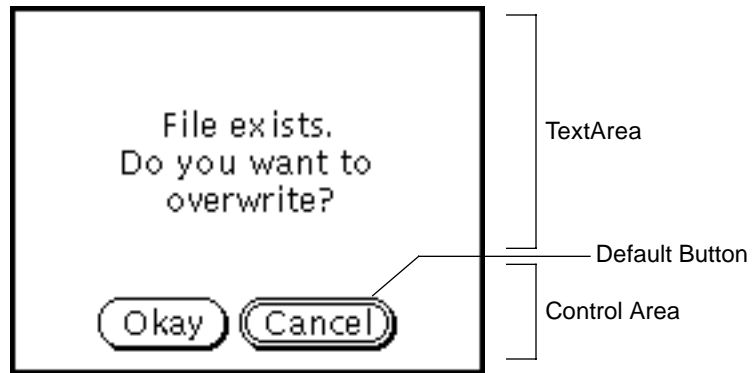


Figure 8-8 NoticeShell Widget

### Subwidgets

The NoticeShell contains two subwidgets, a StaticText and a ControlArea, created automatically, and accessible through the XtNtextArea and XtNcontrolArea resources, respectively.

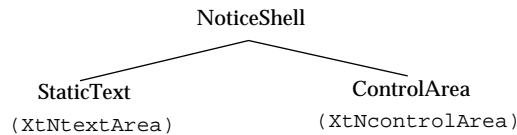


Figure 8-9 MenuShell Subwidgets

### Subclass of the Shell Widget

The NoticeShell widget is a subclass of Shell; therefore, the XtCreatePopupShell() routine is used to create a notice instead of the normal XtCreateWidget().

### Popping the Notice Up and Down

The application controls when the NoticeShell widget is to be displayed or popped up, via the XtPopup() routine. However, the application does not need to control when the NoticeShell widget is popped down. If the control area subwidget of the NoticeShell contains any OblongButton widgets, the

---

## *NoticeShell Widget*

NoticeShell widget will pop down when the user clicks SELECT on one of these buttons. The application can also pop down the NoticeShell itself using `XtPopDown()`.

### *Busy Button, Busy Application*

When the NoticeShell pops up, the application may ignore further input from the user if the `grab_kind` argument to `XtPopup` is set to `XtGrabExclusive` (see the description of `XtPopup` in the *X Toolkit Intrinsics Reference Manual*). This prevents the user from interacting with anything other than the notice. As feedback of this to the user, the NoticeShell causes the headers of all the base windows and popup windows to be stippled in the *busy* pattern, and causes a stipple pattern in the emanate widget. The latter stipple pattern is caused by setting the `XtNbusy` resource to TRUE in the emanate widget. If the emanate widget does not recognize this resource, nothing will happen.

On popping down, the NoticeShell widget clears all stipple patterns and “unfreezes” the application, assuming the `grab_kind` argument to `XtPopup` had been set to `XtGrabExclusive`.

### *Text and ControlAreas*

The Text and Control Areas of the NoticeShell are handled by separate widget interfaces. The widget IDs of these two areas can be obtained from the NoticeShell widget after it has been created using the `XtNtextArea` and `XtNcontrolArea` resources, respectively.

By default, the Text and ControlAreas abut with no space between them. The application can control the distance between them by setting margins on the ControlArea widget.

### *Keyboard Traversal*

The NoticeShell widget limits keyboard traversal of the application to the buttons within the ControlArea.

The user can traverse between the controls in the ControlArea using the NEXTFIELD, PREVFIELD, MOVEUP, MOVEDOWN, MOVERIGHT, and MOVELEFT keys. The NEXTAPP key will traverse to the next application, and the PREVAPP key will traverse to the previous application, but the NEXTWINDOW and

*NoticeShell Widget*

PREVWINDOW keys are disabled. When keyboard traversal is used to move back to the NoticeShell's application, focus goes to the NoticeShell.

*Coloration*

The following diagram illustrates the resources that affect NoticeShell coloration.

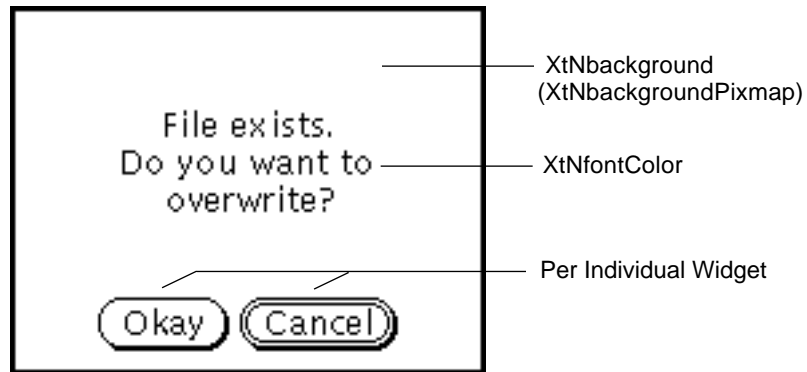


Figure 8-10 NoticeShell Coloration

*Resources*

Table 8-25 NoticeShell Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |

Table 8-25 NoticeShell Core Resources (Continued)

| Name            | Type           | Default    | Access |
|-----------------|----------------|------------|--------|
| XtNscreen       | Screen *       | (parent's) | GI     |
| XtNsensitive    | Boolean        | TRUE       | GIO    |
| XtNtranslations | XtTranslations | NULL       | SGI    |
| XtNwidth        | Dimension      | 0          | SGI    |
| XtNx            | Position       | 0          | SGI    |
| XtNy            | Position       | 0          | SGI    |

Table 8-26 NoticeShell Composite Resources

| Name              | Type        | Default | Access |
|-------------------|-------------|---------|--------|
| XtNchildren       | WidgetList  | NULL    | G      |
| XtNinsertPosition | XtOrderProc | NULL    | SGI    |
| XtNnumChildren    | Cardinal    | 0       | G      |

Table 8-27 NoticeShell Shell Resources

| Name                    | Type                   | Default    | Access |
|-------------------------|------------------------|------------|--------|
| XtNallowShellResize     | Boolean                | TRUE       | SGI    |
| XtNcreatePopupChildProc | XtCreatePopupChildProc | NULL       | SGI    |
| XtNgeometry             | String                 | NULL       | GI     |
| XtNoverrideRedirect     | Boolean                | FALSE      | SGI    |
| XtNpopupdownCallback    | XtCallbackList         | NULL       | SGIO   |
| XtNpopupCallback        | XtCallbackList         | NULL       | SGIO   |
| XtNsaveUnder            | Boolean                | TRUE       | SGI    |
| XtNvisual               | Visual *               | (parent's) | GIO    |

Table 8-28 NoticeShell WMSHELL Resources

| Name          | Type   | Default               | Access |
|---------------|--------|-----------------------|--------|
| XtNbaseHeight | int    | XtUnspecifiedShellInt | SGI    |
| XtNbaseWidth  | int    | XtUnspecifiedShellInt | SGI    |
| XtNheightInc  | int    | XtUnspecifiedShellInt | SGI    |
| XtNiconMask   | Pixmap | NULL                  | SGI    |

*NoticeShell Widget*

*Table 8-28 NoticeShell WMSHELL Resources (Continued)*

| <b>Name</b>      | <b>Type</b>  | <b>Default</b>        | <b>Access</b> |
|------------------|--------------|-----------------------|---------------|
| XtNiconPixmap    | Pixmap       | NULL                  | SGI           |
| XtNiconWindow    | Window       | NULL                  | SGI           |
| XtNiconX         | int          | XtUnspecifiedShellInt | SGI           |
| XtNiconY         | int          | XtUnspecifiedShellInt | SGI           |
| XtNinitialState  | InitialState | NormalState           | SGI           |
| XtNinput         | Bool         | FALSE                 | G             |
| XtNmaxAspectX    | int          | XtUnspecifiedShellInt | SGI           |
| XtNmaxAspectY    | int          | XtUnspecifiedShellInt | SGI           |
| XtNmaxHeight     | int          | OL_IGNORE             | SGI           |
| XtNmaxWidth      | int          | OL_IGNORE             | SGI           |
| XtNminAspectX    | int          | XtUnspecifiedShellInt | SGI           |
| XtNminAspectY    | int          | XtUnspecifiedShellInt | SGI           |
| XtNminHeight     | int          | OL_IGNORE             | SGI           |
| XtNminWidth      | int          | OL_IGNORE             | SGI           |
| XtNtitle         | String       | NULL                  | SGI           |
| XtNtitleEncoding | ATom         | XA_STRING             | SGI           |
| XtNtransient     | Boolean      | TRUE                  | SGI           |
| XtNwaitForWm     | Boolean      | TRUE                  | SGI           |
| XtNwidthInc      | int          | XtUnspecifiedShellInt | SGI           |
| XtNwindowGroup   | Window       | XtUnspecifiedWindow   | SGI           |
| XtNwinGravity    | int          | XtUnspecifiedShellInt | SGI           |
| XtNwmTimeout     | int          | 5000 (msec)           | SGI           |

*Table 8-29 NoticeShell VendorShell Resources*

| <b>Name</b>         | <b>Type</b>       | <b>Default</b>    | <b>Access</b> |
|---------------------|-------------------|-------------------|---------------|
| XtNbusy             | Boolean           | FALSE             | SGI           |
| XtNconsumeEvent     | XtCallbackList    | NULL              | SGIO          |
| XtNdefaultImName    | String            | NULL              | SGI           |
| XtNfooterPresent    | Boolean           | FALSE             | SGI           |
| XtNfocusWidget      | Widget            | (see description) | SGI           |
| XtNimFontSet        | OlFont            | XtDefaultFontSet  | SGI           |
| XtNimStatusStyle    | XtOlImStatusStyle | OL_NO_STATUS      | GI            |
| XtNleftFooterString | OlStr             | NULL              | SGI           |



*NoticeShell Widget**Table 8-29* NoticeShell VendorShell Resources (Continued)

| <b>Name</b>             | <b>Type</b>    | <b>Default</b>                            | <b>Access</b> |
|-------------------------|----------------|-------------------------------------------|---------------|
| XtNleftFooterVisible    | Boolean        | TRUE                                      | SGI           |
| XtNmenuButton           | Boolean        | (see description)                         | GI            |
| XtNmenuType             | OlDefine       | (see description)                         | SGI           |
| XtNpushpin              | OlDefine       | (see description)                         | SGI           |
| XtNresizeCorners        | Boolean        | (see description)                         | SGI           |
| XtNrightFooterString    | OlStr          | NULL                                      | SGI           |
| XtNrightFooterVisible   | Boolean        | TRUE                                      | SGI           |
| XtNshellTitle           | OlStr          | NULL                                      | SGI           |
| XtNuserData             | XtPointer      | NULL                                      | SGI           |
| XtNwindowHeader         | Boolean        | (see description)                         | GI            |
| XtNwmProtocol           | XtCallbackList | NULL                                      | SGIO          |
| XtNwmProtocolInterested | int            | OL_WM_DELETE_WINDOW  <br>OL_WM_TAKE_FOCUS |               |

*Table 8-30* NoticeShell TransientShell Resources

| <b>Name</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-----------------|-------------|----------------|---------------|
| XtNtransientFor | Widget      | NULL           | SGI           |

*Table 8-31* NoticeShell Resources

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNcontrolArea    | Widget      | NULL           | G             |
| XtNemanateWidget  | Widget      | (parent's)     | SGI           |
| XtNpointerWarping | Boolean     | TRUE           | SGI           |
| XtNtextArea       | Widget      | NULL           | G             |
| XtNtextFormat     | OlStrRep    | OL_SB_STR_REP  | GI            |

*NoticeShell Widget*

The following table lists resources passed to the StaticText subwidget maintained by the NoticeShell; they become resources of the StaticText subwidget and can be set and read just like any other NoticeShell resources.

*Table 8-32 NoticeShell StaticText Subwidget Resources<sup>1</sup>*

| <b>Name</b>  | <b>Type</b> | <b>Default</b>      | <b>Access</b> |
|--------------|-------------|---------------------|---------------|
| XtNalignment | OIDefine    | OL_LEFT             | I             |
| XtNfont      | OIFont      | XtDefaultFont       | I             |
| XtNfontColor | Pixel       | XtDefaultForeground | I             |
| XtNlineSpace | int         | 0                   | I             |
| XtNstring    | OIStr       | NULL                | I             |
| XtNstrip     | Boolean     | TRUE                | I             |
| XtNwrap      | Boolean     | TRUE                | I             |

1. These subwidget resource are defined in the section “StaticText Widget” on page 600.

The following table lists resources passed to the ControlArea subwidget maintained by the NoticeShell; they become resources of the ControlArea subwidget and can be set and read just like any other NoticeShell resources.

*Table 8-33 NoticeShell ControlArea Subwidget Resources<sup>1</sup>*

| <b>Name</b>   | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtNhPad       | Dimension   | 4              | I             |
| XtNhSpace     | Dimension   | 4              | I             |
| XtNlayoutType | OIDefine    | OL_FIXEDROWS   | I             |
| XtNmeasure    | int         | 1              | I             |
| XtNsameSize   | OIDefine    | OL_COLUMNS     | I             |
| XtNvPad       | Dimension   | 4              | I             |
| XtNvSpace     | Dimension   | 4              | I             |

1. These subwidget resource are defined in the section “ControlArea Widget” on page 249.

***XtNcontrolArea***

| <b>Class</b>   | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|----------------|-------------|----------------|---------------|
| XtCControlArea | Widget      | NULL           | G             |

Synopsis: The widget ID of the ControlArea widget.

*NoticeShell Widget*

This value is available once the NoticeShell widget has been created. Any widgets of the class `OblongButton` added to the Control Area are assumed to be window disposition controls; that is, when the user activates one of them, the NoticeShell widget pops itself down.

***XtNemanateWidget***

| Class                         | Type   | Default    | Access |
|-------------------------------|--------|------------|--------|
| <code>XtCEmanateWidget</code> | Widget | (parent's) | SGI    |

Synopsis: The widget ID of the emanate widget.

On popping up, the NoticeShell widget attempts to set this widget to be busy, by making its `XtNbusy` resource TRUE; if the widget doesn't recognize the resource, nothing happens.

On popping down, the NoticeShell widget clears the `XtNbusy` resource. When the NoticeShell widget pops up, it tries not to cover this widget; this may fail depending on its location and the size of the NoticeShell widget. The default for this resource is the parent. The parent, however, cannot be a gadget (an `OblongButtonGadget`, for instance). To emanate a NoticeShell from a gadget, specify another widget as the parent and set `XtNemanateWidget` to the gadget.

***XtNpointerWarping***

| Class                          | Type    | Default | Access |
|--------------------------------|---------|---------|--------|
| <code>XtCPointerWarping</code> | Boolean | TRUE    | SGI    |

Synopsis: Whether the pointer will jump to the default button when the Notice is displayed.

Values: TRUE/"true" - The pointer jumps to the default button.  
FALSE/"false" - The pointer doesn't jump.

***XtNtextArea***

| Class                    | Type   | Default | Access |
|--------------------------|--------|---------|--------|
| <code>XtCTextArea</code> | Widget | NULL    | G      |

Synopsis: The widget ID of the StaticText widget that controls the text area.

This value is available once the NoticeShell widget has been created.

*NoticeShell Widget*

***XtNtextFormat***

| <b>Class</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtCTextFormat | OIStrRep    | OL_SB_STR_REP  | GI            |

Synopsis: The expected data format of all the textual resources of a widget.

Values: OL\_SB\_STR\_REP - Single-byte character representation.  
 OL\_WC\_STR\_REP - Wide character representation.  
 OL\_MB\_STR\_REP - Multibyte character representation.

See “XtNtextFormat” on page 29 for details of initialization and the default value.

***Activation Types***

The following table lists the activation types used by the NoticeShell.

*Table 8-34* NoticeShell Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The NoticeShell widget has no activation types besides the ones in “Common Activation Types” on page 68.

***See Also***

“ControlArea Widget” on page 249 ,  
 “StaticText Widget” on page 600.

## NumericField Widget

### Class

**Class Name:** NumericField  
**Class Pointer:** numericFieldWidgetClass

### Ancestry

Core-Primitive-TextLine-NumericField

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/AbbrevMenu.h>
```

### Description

The NumericField widget is a one-line input field for alphanumeric text. It implements the Numeric TextField object defined in the *OPEN LOOK GUI Functional Specification*. The widget supports the basic datatypes—integers and floats—and provides hooks for more complicated user-defined datatypes. Providing support for any datatype involves:

- Implementing Converters between the datatype and `String`
- Implementing per-key and per-field validation
- Implementing increment and decrement operations

Once the input focus is moved into the widget, keyboard entry is allowed. If the input value exceeds the length of the input field, the ScrollButtons appear. Hidden text can then be scrolled into view by pressing the ScrollButtons. Pressing the buttons continuously scrolls the text repeatedly with a user-adjustable delay. The Increment or Decrement buttons can be used to increment or decrement the input value in an appropriate manner.

### Components

The NumericField contains the following graphical elements:

- Right-justified bold label at the left of the input field.

## NumericField Widget

- Input field
- Input caret (not present in ReadOnly mode)
- 1-point (for Mono) or chiseled underline (not present in ReadOnly mode)
- Optional ScrollButtons
- Increment and Decrement buttons, referred to as *DeltaButtons* hereafter

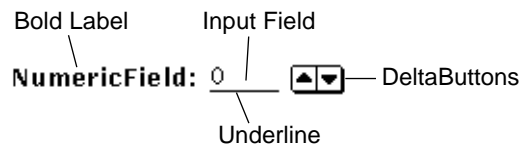


Figure 8-11 NumericField Components

### Keyboard Traversal

The NumericField allows keyboard entry if it is sensitive and it has the keyboard focus. However, in ReadOnly mode the widget is not traversable and it does not receive keyboard focus. Selection is also turned off during ReadOnly mode.

The NumericField responds to the following keyboard navigation keys:

- NEXTFIELD moves to the next traversable widget in the window
- PREVIOUSFIELD moves to the previous traversable widget in the window
- CHARFWD moves the caret forward one character
- CHARBAK moves the caret backward one character
- WORDFWD moves the caret forward one word
- WORDBAK moves the caret back one word
- LINESTART moves the caret to the beginning of the display
- LINEEND moves the caret to the end of the display
- MENUKEY posts the menu associated with the NumericField

The NumericField responds to the following edit keys:

- DELCHARFWD deletes the character to the right of the caret
- DELCHARBAK deletes the character to the left of the caret
- DELWORDFWD deletes the word to the right of the caret
- DELWORDBAK deletes the word to the left of the caret
- DELLINEFWD deletes to the end of the line from the caret
- DELLINEBAK deletes from the beginning of the line to the caret
- DELLINE deletes the line containing the caret
- UNDO undoes the last edit

***Keyboard Mnemonic Display***

The NumericField does not display any mnemonic. If the NumericField is the child of a Caption widget, the Caption can be used to display any mnemonic.

***Keyboard Accelerator Display***

The NumericField does not respond to any keyboard accelerators.

***Display of Text***

The NumericField displays its contents in the font specified by the `XtNfont` resource. If the length of the text exceeds the length of the input field, the widget sets up the left or right scrollbuttons (or both) to indicate this. The text is then visually truncated at the boundaries to show only as many characters as can fit in the input field. The truncation is always at a character boundary. A scrollbutton is present only if characters are hidden in the direction indicated by that buttons. The user can scroll to show the hidden parts of the text by clicking or pressing the scrollbuttons. Clicking SELECT on any scrollbutton will scroll the text one character in the direction indicated by that button. Pressing SELECT on any scrollbutton will repeat the scrolling with a user-adjustable delay between each scroll.

***Caret Position***

As characters are entered from the keyboard, the caret moves to the right until it reaches the right end of the input field. As additional characters are typed, the text jump-scrolls to the left by a specific amount. Note that the NumericField always keeps the cursor visible. Thus, the presence or absence of either of the scrollbuttons is controlled by the current cursor position.

***Selection Of Text***

The user can perform text selection by using the mouse or the keyboard. The widget also provides a set of convenience functions to manipulate the selection programmatically; see “TextLine Functions” on page 708.

***DeltaButtons***

The DeltaButtons can be used to increment or decrement the value in the NumericField by a specified amount. The widget provides default logic for the supported datatypes (`ints` and `floats`) as well as hooks for user-defined datatypes. The DeltaButtons become insensitive when the value equals the maximum or minimum range.

***Coloration***

For 3D and 2D, `XtNfontColor` is used to draw the NumericField's text and `XtNinputFocusColor` is used to draw the active caret.

For 3D, the NumericField's underline and increment/decrement button coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNbackground` and `XtNforeground` are used to render the NumericField's underline and increment/decrement buttons as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

***Known Deficiencies***

The TextLine widget, on which the NumericField is built, currently does not support the *implicit commit* feature; see "Input Method" on page 80. This could be a deficiency in Asian locales.

***Resources***

*Table 8-35* NumericField Core Resources

| <b>Name</b>                       | <b>Type</b>      | <b>Default</b>                   | <b>Access</b> |
|-----------------------------------|------------------|----------------------------------|---------------|
| <code>XtNaccelerators</code>      | AcceleratorTable | NULL                             | SGI           |
| <code>XtNancestorSensitive</code> | Boolean          | TRUE                             | G             |
| <code>XtNbackground</code>        | Pixel            | <code>XtDefaultBackground</code> | SGID          |
| <code>XtNbackgroundPixmap</code>  | Pixmap           | <code>XtUnspecifiedPixmap</code> | SGI           |
| <code>XtNborderColor</code>       | Pixel            | <code>XtDefaultForeground</code> | SGID          |
| <code>XtNborderPixmap</code>      | Pixmap           | <code>XtUnspecifiedPixmap</code> | SGI           |



*NumericField Widget**Table 8-35* NumericField Core Resources (Continued)

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNborderWidth       | Dimension      | 1              | SGI           |
| XtNcolormap          | Colormap       | (parent's)     | SGI           |
| XtNdepth             | int            | (parent's)     | GI            |
| XtNdestroyCallback   | XtCallbackList | NULL           | SGIO          |
| XtNheight            | Dimension      | 0              | SGI           |
| XtNmappedWhenManaged | Boolean        | TRUE           | SGI           |
| XtNscreen            | Screen *       | (parent's)     | G             |
| XtNsensitive         | Boolean        | TRUE           | GIO           |
| XtNtranslations      | XtTranslations | NULL           | SGI           |
| XtNwidth             | Dimension      | 0              | SGI           |
| XtNx                 | Position       | 0              | SGI           |
| XtNy                 | Position       | 0              | SGI           |

*Table 8-36* NumericField Primitive Resources

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|--------------------|----------------|---------------------|---------------|
| XtNaccelerator     | String         | NULL                | SGI           |
| XtNacceleratorText | String         | NULL                | SGI           |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont       | SGI           |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor | Pixel          | Red                 | SGID          |
| XtNmnemonic        | unsigned char  | '\0'                | SGI           |
| XtNreferenceName   | String         | NULL                | GI            |
| XtNreferenceWidget | Widget         | NULL                | GI            |
| XtNscale           | int            | 12                  | SGI           |
| XtNtextFormat      | OlStrRep       | OL_SB_STR_REP       | GI            |
| XtNtraversalOn     | Boolean        | TRUE                | SGI           |
| XtNuserData        | XtPointer      | NULL                | SGI           |

*NumericField Widget*

*Table 8-37 NumericField TextLine Resources<sup>1</sup>*

| <b>Name</b>                     | <b>Type</b>      | <b>Default</b>    | <b>Access</b> |
|---------------------------------|------------------|-------------------|---------------|
| XtNblinkRate                    | int              | 1000              | SGI           |
| XtNcaptionAlignment             | OlDefine         | OL_CENTER         | G             |
| XtNcaptionFont <sup>2</sup>     | OlFont           | OlDefaultBoldFont | SI            |
| XtNcaptionLabel <sup>2</sup>    | OlStr            | NULL              | SGI           |
| XtNcaptionPosition <sup>2</sup> | OlDefine         | OL_LEFT           | G             |
| XtNcaptionSpace <sup>2</sup>    | Dimension        | 4                 | G             |
| XtNcaptionWidth <sup>2</sup>    | Dimension        | 0                 | G             |
| XtNcharsVisible                 | int              | 0                 | GI            |
| XtNcommitCallback               | XtCallbackList   | (special)         | SGIO          |
| XtNcursorPosition               | int              | 0                 | SGI           |
| XtNeditType                     | OlDefine         | OL_TEXT_EDIT      | SGI           |
| XtNimPreeditStyle               | OlImPreeditStyle | OL_NO_PREEDIT     | GI            |
| XtNinitialDelay                 | int              | 500               | SGI           |
| XtNinsertTab                    | Boolean          | FALSE             | SGI           |
| XtNmaximumChars                 | int              | 0                 | GI            |
| XtNmenu                         | Widget           | NULL              | GI            |
| XtNmotionCallback               | XtCallbackList   | NULL              | SGIO          |
| XtNpostModifyCallback           | XtCallbackList   | NULL              | SGIO          |
| XtNpreModifyCallback            | XtCallbackList   | (special)         | SGIO          |
| XtNrepeatRate                   | int              | 100               | SGI           |
| XtNstring                       | OlStr            | NULL              | SGI           |
| XtNupdateDisplay                | Boolean          | TRUE              | SGI           |

1. These resources are defined in the section “TextLine Widget” on page 688.

2. These resources are provided by the TextLine to support captions and are subject to change in a future OLIT release.

*Table 8-38 NumericField Resources*

| <b>Name</b>      | <b>Type</b>     | <b>Default</b>                           | <b>Access</b> |
|------------------|-----------------|------------------------------------------|---------------|
| XtNconvertProc   | OlNFConvertProc | (special)                                | I             |
| XtNdelta         | XtPointer       | 1 for ints; 0.1 for floats;<br>else NULL | SGI           |
| XtNdeltaCallback | XtCallbackList  | NULL                                     | SGIO          |

Table 8-38 NumericField Resources (Continued)

| Name                | Type           | Default      | Access |
|---------------------|----------------|--------------|--------|
| XtNdeltaState       | OlDefine       | OL_ACTIVE    | SGI    |
| XtNmaxValue         | XtPointer      | NULL         | SGI    |
| XtNminValue         | XtPointer      | NULL         | SGI    |
| XtNsizeof           | Cardinal       | (calculated) | GI     |
| XtNtype             | String         | XtRInt       | GI     |
| XtNvalidateCallback | XtCallbackList | NULL         | SGIO   |
| XtNvalue            | XtPointer      | NULL         | SGI    |

The NumericField supports the basic datatypes `int` and `float`. The widget also provides a mechanism for applications to implement their own datatypes. Since the widget can support multiple datatypes, all resources that refer to any data are uniformly represented as pointers to the actual data, which could be any of the above widget-supported types or the application's specific type. Implementing additional datatypes is discussed in "Implementing New Datatypes" on page 455.

### ***XtNconvertProc***

| Class          | Type                | Default   | Access |
|----------------|---------------------|-----------|--------|
| XtCConvertProc | XtROIINFConvertProc | (special) | I      |

Synopsis: The widget-defined `int/float` to `String` converter.

This resource specifies the logic to convert between `String` and the user-defined datatype. This needs to be set only when the application has defined a custom datatype. See "Implementing New Datatypes" on page 455.

### ***XtNdelta***

| Class    | Type      | Default                                  | Access |
|----------|-----------|------------------------------------------|--------|
| XtCDelta | XtPointer | 1 for ints; 0.1 for floats;<br>else NULL | SGI    |

Synopsis: The Increment or Decrement delta to be applied to the data field, when either of the `DeltaButtons` is pressed. The widget applies the delta internally for `ints` and `floats`.

By default, the widget makes internal copies of the data. Setting the `XtNsizeof` resource to zero prevents the widget from doing so. See the section

*NumericField Widget*

on “Implementing New Datatypes” on page 455 for reasons and techniques of doing this.

***XtNdeltaState***

| Class         | Type     | Default   | Access |
|---------------|----------|-----------|--------|
| XtCDeltaState | OlDefine | OL_ACTIVE | SGI    |

Synopsis: The state of the DeltaButtons.

Values: OL\_ACTIVE/"active" - Both buttons are active.  
 OL\_INCR\_INACTIVE/"incr\_inactive" - The increment button is inactive, the decrement button is active.  
 OL\_DECR\_INACTIVE/"decr\_inactive" - The decrement button is inactive, the increment button is active.  
 OL\_ABSENT/"absent" - DeltaButtons are not present.

***XtNdeltaCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when either of the DeltaButtons is pressed. If the buttons are pressed continuously, the callback will be invoked multiple times. The callback is not invoked if the DeltaButtons are not present.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 XEvent *event;
 XtPointer delta;
 XtPointer current_value;
 OlDefine current_delta_state;
 XtPointer new_value;
 OlDefine new_delta_state;
 Boolean update;
} OlNFDeltaCallbackStruct;
```

The fields signify:

*reason*            OL\_REASON\_INCREMENT or OL\_REASON\_DECREMENT  
*event*             A pointer to the corresponding XEvent structure  
*delta*              The value of the XtNdelta resource  
*current\_value*     The current value of the XtNvalue resource

---

*NumericField Widget*

|                            |                                                                                                                                                                                                                             |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>current_delta_state</i> | The current value of <code>XtNdeltaState</code>                                                                                                                                                                             |
| <i>new_value</i>           | The new value of the <code>XtNvalue</code> resource if the widget's default delta logic (if any) is carried out. The application can override the default behavior by changing the contents of this field to any new value. |
| <i>new_delta_state</i>     | The new value of the <code>XtNdeltaState</code> if the widget's default delta logic (if any) is carried out. The application can override the default behavior by changing this field to any new value.                     |
| <i>update</i>              | This resource should be set to TRUE to indicate that the <i>new_value</i> has changed.                                                                                                                                      |

If `XtNtype` is user-defined, then modifying *new\_value* to the new value and setting *update* to TRUE will cause the widget to update `XtNvalue` with the new value, erase the *old\_value*, and display the *new\_value*. If the value of the *new\_delta\_state* field changes, the widget will update the `XtNdeltaState` resource and redisplay the `DeltaButtons` appropriately.

If `XtNtype` is `XtRFloat`/`XtRInt`, the *new\_value* field is computed by the widget by incrementing/decrementing `XtNvalue` by `XtNdelta` and validating it against `XtNmaxValue`/`XtNminValue`. `XtNdeltaState` is also set up appropriately by the widget. The application can override this default logic by modifying *new\_value* and setting *update* to TRUE. This would cause the widget to update `XtNvalue` with the new value, erase the *old\_value*, and display the *new\_value*. Note that no increment/decrement or validation will be done by the widget. If the value of the *new\_delta\_state* field is changed, the widget would update the `XtNdeltaState` resource and redisplay the `DeltaButtons` appropriately. This is also the default behavior (i.e., if no callbacks are registered).

***XtNmaxValue/  
XtNminValue***

| Class                    | Type                   | Default | Access |
|--------------------------|------------------------|---------|--------|
| <code>XtCMaxValue</code> | <code>XtPointer</code> | NULL    | SGI    |
| <code>XtCMinValue</code> | <code>XtPointer</code> | NULL    | SGI    |

Synopsis: The maximum (minimum) value that can be set for the data field.

For datatypes `XtRInt` and `XtRFloat`, the widget does the following if this resource is not NULL:

*NumericField Widget*

- While incrementing (decrementing) the data field, if the value is greater than or equal to `XtNmaxValue` (less than or equal to `XtNminValue`), the increment (decrement) button is desensitized.
- When the value is committed by the user, the widget verifies that the value does not exceed `XtNmaxValue` (is not less than `XtNminValue`).

By default, the widget makes internal copies of the data. Setting the `XtNsizeof` resource to zero prevents the widget from doing so. See the section on “Implementing New Datatypes” on page 455 for reasons and techniques of doing this.

***XtNsizeof***

| Class                  | Type     | Default      | Access |
|------------------------|----------|--------------|--------|
| <code>XtCsizeof</code> | Cardinal | (calculated) | GI     |

Synopsis: The size of the datatype that is used by the application.  
 Values: Size of the datatype or zero.

This value needs to be set only when the application has defined a custom datatype. The special value of zero can be used to indicate that the widget need not make copies of any resources. See the section on “Implementing New Datatypes” on page 455 for reasons and techniques of doing this.

***XtNtype***

| Class                 | Type   | Default             | Access |
|-----------------------|--------|---------------------|--------|
| <code>XtCctype</code> | String | <code>XtRint</code> | GI     |

Synopsis: The type of the data.  
 Values: `XtRint` - Integer  
           `XtRfloat` - Float

The application can also define a custom datatype, as described in “Implementing New Datatypes” on page 455.

***XtNvalidateCallback***

| Class                    | Type                        | Default | Access |
|--------------------------|-----------------------------|---------|--------|
| <code>XtCCallback</code> | <code>XtCallbackList</code> | NULL    | SGIO   |

Synopsis: The validation callback.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 XEvent *event;
 XtPointer value;
 OlDefine delta_state;
 Boolean update;
 Boolean valid;
} OlNFValidateCallbackStruct;
```

This callback is invoked when the user presses the RETURN, NEXTFIELD, or PREVFIELD keys. The fields signify:

|                    |                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i>      | OL_REASON_VALIDATE                                                                                                                                 |
| <i>event</i>       | A pointer to the corresponding XEvent structure                                                                                                    |
| <i>value</i>       | The current value of the XtNvalue resource. The application can make appropriate changes to this field and set the update flag to update XtNvalue. |
| <i>delta_state</i> | The current value of XtNdeltaState. This can be set to a new value to update this resource.                                                        |
| <i>update</i>      | This resource should be set to TRUE to indicate that the value has changed.                                                                        |
| <i>valid</i>       | A field that indicates the validity of the contents.                                                                                               |

If *XtNtype* is *XtRInt* or *XtRFloat*, then the widget will internally validate the *value* against *XtNmaxValue* and *XtNminValue*. If the value is invalid, it sets *valid* to FALSE. The callback can take appropriate action, like beeping, popping up a notice, etc. It can also reset the *value* and *delta\_state* resources to suitable default values by modifying those fields and setting *update* to TRUE. On return from the callback, if *valid* is FALSE, the widget maintains focus within the widget. If *valid* is TRUE, the widget relinquishes focus. If *XtNtype* is user-defined, the widget invokes this callback immediately. Depending on the *valid* flag, the widget will maintain focus or relinquish focus.

### ***XtNvalue***

| Class    | Type      | Default | Access |
|----------|-----------|---------|--------|
| XtCValue | XtPointer | NULL    | SGI    |

Synopsis: The pointer to an *int*, *float*, or the user-defined datatype.

## NumericField Widget

The contents of the pointer is the data to be displayed. The data is converted into a string (depending on the datatype) and used as the `XtNstring` resource of the superclass, the `TextLine` widget.

By default, the `NumericField` widget makes internal copies of the data. Setting the `XtNsizeof` resource to zero prevents the widget from doing so. See the section on “Implementing New Datatypes” on page 455 for reasons and techniques of doing this.

Examples for `XtVaSetValues()`:

```
int i = 10;
XtVaSetValues(w, XtNvalue, &i, NULL);
...
int *ip;
XtVaGetValues(w, XtNvalue, &ip, NULL);
```

In the second example, *ip* now points to data in the widget’s space. Therefore, the application must make copies if it needs to manipulate this data.

### Per-Key and Per-Field Validation

The `NumericField` widget uses the callbacks of its superclass to implement per-key and per-field validation, as described below.

The `NumericField` widget itself does per-key validation for the datatypes `XtRInt` and `XtRFloat`. The widget registers `XtNpreModifyCallbacks` on its superclass (`TextLine`) for ints and floats. The callback for `int` verifies that each keystroke is a valid digit; the callback for `float` ensures that each keystroke is either a digit or a decimal separator and that only one decimal separator occurs in the field. Applications can override this behavior by removing the widget-registered callback and installing their own callback. For user-defined datatypes, the application can register a customized per-key validation callback.

The `NumericField` widget also does per-field validation for the datatypes `XtRInt` and `XtRFloat`, which verifies that the field-value is within the minimum and maximum ranges. The widget registers a callback on the `XtNcommitCallback` of the `TextLine` superclass. This callback verifies the field-value against the minimum and maximum ranges, if they exist. On success, the widget sets the *valid* field of the `OlNFValidateCallbackStruct` to `TRUE` and invokes any `XtNvalidateCallback` callbacks registered. On



failure, the *valid* flag is set to `FALSE` and any `XtNvalidateCallback` callbacks present are invoked.

## *Implementing New Datatypes*

The `NumericField` widget currently supports `int` and `float` as the base datatypes; i.e., the widget internally provides all logic and code necessary to handle these datatypes. The widget also allows applications to define and use their own datatypes. Examples of such user-defined datatypes would include Date, Time, Currency, etc. Implementing a new datatype involves the following steps:

- 1. Register the new datatype with the widget.**
- 2. Implement Converters between the datatype and String.**
- 3. Optionally implement Per-Key and Per-Field Validation routines.**
- 4. Optionally implement Increment and Decrement operations.**

Each step is elaborated below:

1. The application specifies a name for the new datatype. The name should be a printable ASCII String. This name is registered with the widget by setting it up as the `XtNtype` resource. The application should also set the `XtNsizeof` resource to the size in bytes of the new datatype. The widget internally uses this value while making copies of the contents of `XtNvalue`, `XtNdelta`, `XtNminValue`, and `XtNmaxValue`. `XtNsizeof` can be set to the special value of '0' to indicate that the widget need not make copies of the above resources. This will result in the widget accessing these values directly through pointers to the application's dataspace.

Typically, a programmer would want to do this if the datatype is large in size and the application stores all the above resource values in global space or space allocated by `malloc(3)`, so that copying between the application's dataspace and the widget's dataspace is redundant or inefficient. Caution should be used, however, since modifying any of these values also directly affects the widget's state.

2. The `NumericField` displays the contents of `XtNvalue` by converting the contents to a string and setting it up as the `XtNstring` resource of its superclass—the `TextLine`. Similarly, the widget updates `XtNvalue` by retrieving the `XtNstring` resource value of the `TextLine` and converting it

*NumericField Widget*

back to the specified datatype. The widget does these conversions internally for the base datatypes. However, for user-defined datatypes, the application needs to provide this conversion logic. This is done by setting up the `XtNconvertProc` function. The prototype of this function is as follows:

```
Boolean Converter(
 Widget w,
 OlStrRep format,
 XrmQuark from_type,
 XrmQuark to_type,
 XtPointer *value,
 XtPointer *string,
 Cardinal *string_length);
```

|                      |                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------|
| <i>w</i>             | The widget.                                                                                                                |
| <i>format</i>        | The <i>text_format</i> of the widget.                                                                                      |
| <i>from_type</i>     | The source datatype; i.e, the datatype to be converted from. This is a quarkified representation of the datatype name.     |
| <i>to_type</i>       | The destination datatype; i.e., the datatype to be converted to. This is a quarkified representation of the datatype name. |
| <i>value</i>         | A pointer to the data stored.                                                                                              |
| <i>string</i>        | A pointer to the data stored as a string.                                                                                  |
| <i>string_length</i> | The size of the memory where the string is stored.                                                                         |

The Converter should do the following:

- Attempt the type conversion from the source datatype to the destination datatype.
  - Copy the converted data to the appropriate location; i.e., in *string* if the destination datatype is `String`, in *value* if the destination datatype is the user's datatype. Before copying, ensure that the location is a valid memory pointer and that the storage size is sufficient. If not, allocate sufficient memory and then copy the data. Update *\*string\_length* to the length of the string if the destination datatype is `String`.
  - If the conversion failed, issue a warning and return `FALSE`; otherwise, return `TRUE`. The *string* datatype is identified as `XtRString`.
3. Per-Key validation ensures that each key is valid in the current context. The application can verify this by registering an appropriate `XtNpreModifyCallback` on the widget.

*NumericField Widget*

Per-Field validation ensures that the contents of the widget are valid, when the user commits a value (i.e., presses <return> or <Tab>). The application can achieve this by registering an appropriate `XtNvalidateCallback`. The callback should set the *valid* field of the *call\_data* to indicate the validity of the data. The callback can also reset the widget contents by updating *value* and setting *update* to TRUE.

4. The widget should update its contents and the state of its DeltaButtons whenever the buttons are pressed. The widget does this internally for the base datatypes. For user-defined datatypes, the application can implement the necessary logic by registering an appropriate `XtNdeltaCallback`. The callback should update the *new\_value* and *new\_delta\_state* fields in the *call\_data* and set *update* to TRUE, if the contents of the widget need to be changed.

*Activation Types*

The following table lists the activation types used by the NumericField.

*Table 8-39* NumericField Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_CHARBAK             | LEFT             | XtNleftKey           |
| OL_CHARFWD             | RIGHT            | XtNrightKey          |
| OL_COPY                | COPY             | XtNcopyBtn           |
| OL_CUT                 | CUT              | XtNcutBtn            |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_DELCHARBAK          | DELETE BACKWARD  | XtNdelCharBakFwd     |
| OL_DELCHARFWD          | DELETE FORWARD   | XtNdelCharFwdKey     |
| OL_DELLINE             | DELETE LINE      | XtNdelLineKey        |
| OL_DELLINEBAK          | DELLINEBAK       | XtNdelLineBakKey     |
| OL_DELLINEFWD          | DELLINEFWD       | XtNdelLineFwdKey     |
| OL_DELWORDBAK          | DELWORDBAK       | XtNdelWordBakKey     |
| OL_DELWORDFWD          | DELWORDFWD       | XtNdelWordFwdKey     |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_LINEEND             | ROW END          | XtNlineEndKey        |
| OL_LINESTART           | ROW START        | XtNlineStartKey      |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |

Table 8-39 NumericField Activation Types (Continued)

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT     | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP        | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD     | NEXTFIELD     | XtNnextFieldKey     |
| OL_PASTE         | PASTE         | XtNpasteBtn         |
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |
| OL_UNDO          | UNDO          | XtNundoKey          |
| OL_WORDBAK       | JUMP LEFT     | XtNwordBakKey       |
| OL_WORDFWD       | JUMP RIGHT    | XtNwordFwdKey       |

Activation types not described in the following table are described in “Common Activation Types” on page 68.

**OL\_CHARBAK**

The cursor is moved backward by one character. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the `valid` field in the `call_data` to `FALSE`.

**OL\_CHARFWD**

The cursor is moved forward by one character. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the `valid` field in the `call_data` to `FALSE`.

**OL\_COPY**

This activation type copies the current selection from the widget to the CLIPBOARD.

**OL\_CUT**

This activation type copies the current selection from the widget to the CLIPBOARD and also deletes the selected text from the widget. The following callbacks are invoked:

`XtNpreModifyCallback` Invoked before the deletion occurs. The callback can prevent the deletion by setting the `valid` field in the `call_data` to `FALSE`.

---

*NumericField Widget*

|                       |                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNpostModifyCallback | Invoked after the deletion occurs.                                                                                                                                                  |
| XtNmotionCallback     | Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

***OL\_DELCHARBAK***

If there exists a selection in the widget, it is deleted. If there is no selection, the character before the insert point is deleted. The following callbacks are invoked:

|                       |                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNpreModifyCallback  | Invoked before the deletion occurs. The callback can prevent the deletion by setting the <i>valid</i> field in the <i>call_data</i> to FALSE.                                       |
| XtNpostModifyCallback | Invoked after the deletion occurs.                                                                                                                                                  |
| XtNmotionCallback     | Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

***OL\_DELCHARFWD***

If there exists a selection in the widget, it is deleted. If there is no selection, the character after the insert point is deleted. The following callbacks are invoked:

|                       |                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNpreModifyCallback  | Invoked before the deletion occurs. The callback can prevent the deletion by setting the <i>valid</i> field in the <i>call_data</i> to FALSE.                                       |
| XtNpostModifyCallback | Invoked after the deletion occurs.                                                                                                                                                  |
| XtNmotionCallback     | Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

**OL\_DELLINE**

The whole line is deleted. The following callbacks are invoked:

- XtNpreModifyCallback    Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.
- XtNpostModifyCallback    Invoked after the deletion occurs.
- XtNmotionCallback    Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

**OL\_DELLINEBAK**

If there exists a selection in the widget, it is deleted. If there is no selection, the segment of the line before the insert point The following callbacks are invoked:

- XtNpreModifyCallback    Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.
- XtNpostModifyCallback    Invoked after the deletion occurs.
- XtNmotionCallback    Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

**OL\_DELLINEFWD**

If there exists a selection in the widget, it is deleted. If there is no selection, the segment of the line after the insert point is deleted. The following callbacks are invoked:

- XtNpreModifyCallback    Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.
- XtNpostModifyCallback    Invoked after the deletion occurs.
- XtNmotionCallback    Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

**OL\_DELWORDBAK**

If there exists a selection in the widget, it is deleted. If there is no selection, the word before the insert point is deleted. The following callbacks are invoked:

|                       |                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNpreModifyCallback  | Invoked before the deletion occurs. The callback can prevent the deletion by setting the <i>valid</i> field in the <i>call_data</i> to FALSE.                                       |
| XtNpostModifyCallback | Invoked after the deletion occurs.                                                                                                                                                  |
| XtNmotionCallback     | Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

**OL\_DELWORDFWD**

If there exists a selection in the widget, it is deleted. If there is no selection, the word after the insert point is deleted. The following callbacks are invoked:

|                       |                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNpreModifyCallback  | Invoked before the deletion occurs. The callback can prevent the deletion by setting the <i>valid</i> field in the <i>call_data</i> to FALSE.                                       |
| XtNpostModifyCallback | Invoked after the deletion occurs.                                                                                                                                                  |
| XtNmotionCallback     | Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

**OL\_LINESTART**

The cursor is moved to the start of the line. The XtnmotionCallback callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

**OL\_LINEEND**

The cursor is moved to the end of the line. The XtnmotionCallback callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

**OL\_NEXTFIELD**

This activation type invokes the `XtNcommitCallback`. If the callback sets *valid* to TRUE and if `XtNmouseless` is TRUE, the widget transfers focus to the next traversable widget. If the callback sets *valid* to TRUE and if `XtNmouseless` is FALSE, the widget transfers focus and sets the insert point to the next `TextLine` or `TextEdit` widget. If the callback sets *valid* to FALSE, the widget maintains focus and insert point within itself.

**OL\_PASTE**

This activation type inserts the contents of the CLIPBOARD into the widget at the current insert point. The following callbacks are invoked:

- `XtNpreModifyCallback`    Invoked before the insertion occurs. The callback can prevent the insertion by setting the *valid* field in the *call\_data* to FALSE.
- `XtNpostModifyCallback`    Invoked after the insertion occurs.
- `XtNmotionCallback`    Invoked before the cursor position is changed due to the insertion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

**OL\_PREVFIELD**

This activation type invokes the `XtNcommitCallback`. If the callback sets *valid* to TRUE and if `XtNmouseless` is TRUE, the widget transfers focus to the previous traversable widget. If the callback sets *valid* to TRUE and if `XtNmouseless` is FALSE, the widget transfers focus and sets the insert point to the previous `TextLine` or `TextEdit` widget. If the callback sets *valid* to FALSE, the widget maintains focus and insert point within itself.

**OL\_UNDO**

This activation type undoes the last modification to the widget's text buffer. The following callbacks are invoked:

- `XtNpreModifyCallback`    Invoked before any modification occurs. The callback can prevent the modification by setting the *valid* field in the *call\_data* to FALSE.
- `XtNpostModifyCallback`    Invoked after any modification occurs.



---

*NumericField Widget*

`XtNmotionCallback`      Invoked before the cursor position is changed due to the modification. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to `FALSE`.

***OL\_WORDBAK***

The cursor is moved backward by one word. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to `FALSE`.

***OL\_WORDFWD***

The cursor is moved forward by one word. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to `FALSE`.

***See Also***

“TextLine Widget” on page 688.

---

*OblongButton Widget*  
***OblongButton Widget***

***Class***

**Class Name:** OblongButton  
**Class Pointer:** oblongButtonWidgetClass, oblongButtonGadgetClass

***Ancestry***

Core-Primitive-Button-OblongButton

***Required Header Files***

```
#include <Xol/OpenLook>
#include <Xol/OblongButt.h>
```

***Description***

The OblongButton is an action widget that the user can “push” by pressing SELECT on it. When the button is pushed, its border inverts, making it appear as if the button has actually been pressed. It is typically used to initiate one or several application-defined actions.

***Components***

The OblongButton consists of a label surrounded by a rounded, or oblong, border.

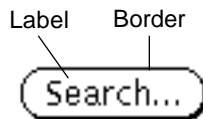


Figure 8-12 **OblongButton Components**

***Busy Indication During Callbacks***

Each OblongButton is associated with an application-defined action implemented as a list of callbacks. To let the user know that an action is still

---

### *OblongButton Widget*

taking place, the `OblongButton` stipples the area inside the border before issuing the callbacks. When the last callback returns, the `OblongButton` restores its original appearance. If the application's action continues to be “busy” after the callbacks return, the application should set the `XtNbusy` resource to `TRUE` before returning from the callbacks, then reset it to `FALSE` when the action is no longer taking place.

The “busy” stipple pattern is designed to show enough dots to gray the button noticeably, while still leaving a text label legible.

### *OblongButtons In Popup Menus*

Entering an oblong button while `MENU` is depressed highlights the button's interior. Releasing `MENU` then restores the original appearance and invokes the action for the button as described above. Leaving the button before releasing `MENU` restores the appearance but does not invoke the action.

### *OblongButtons Not In Popup Menus*

Clicking `SELECT` on an `OblongButton` starts the action associated with the button. Pressing `SELECT`, or moving the pointer into the button while `SELECT` is pressed, highlights the button's interior. Releasing `SELECT` restores the appearance and invokes the action for the button as described above. Moving the pointer off the button before releasing `SELECT` also restores the appearance, but does not invoke the action.

If the `OblongButton` is in a stay-up menu, clicking or pressing `MENU` works the same as `SELECT`. If the `OblongButton` is not in a stay-up (or popup) menu, clicking or pressing `MENU` does not do anything; the event is passed up to an ancestor widget.

### *OblongButton Gadgets*

Correct button behavior is not guaranteed if gadgets are positioned so that they overlap.

Gadgets share some `Core` fields but, since they are not subclasses of `Core`, do not have all `Core` fields. In particular, they do not have a `name` field or a `translation` field, so translations cannot be specified or overridden.

Event Handlers cannot be added to gadgets using `XtAddEventHandler()`.

### *Coloration*

For 3D, *OblongButton* coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.”

`XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1. `XtNfontColor` is used to draw the label.

For 2D, `XtNbackground` and `XtNfontColor` are used to render the *OblongButton* as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, “Controls.”

If the toolkit resource `XtNmouseless` is set to TRUE and the toolkit resource `XtNinputFocusFeedback` is set to `OL_INPUT_FOCUS_COLOR`, then the background of the *OblongButton* will be drawn with the value of `XtNinputFocusColor` when the widget receives input focus. However, if `XtNinputFocusColor` is the same as `XtNbackground`, then the *OblongButton* inverts `XtNfontColor` and `XtNbackground`. Once the input focus leaves the widget, the original coloration is restored.

### *Keyboard Traversal*

The default value of the `XtNtraversalOn` resource is TRUE.

The *OblongButton* widget responds to the following keyboard navigation keys:

- `NEXTFIELD`, `MOVEDOWN`, and `MOVERIGHT` move to the next traversable widget in the window
- `PREVFIELD`, `MOVEUP`, and `MOVELEFT` move to the previous traversable widget in the window
- `NEXTWINDOW` moves to the next window in the application
- `PREVWINDOW` moves to the previous window in the application
- `NEXTAPP` moves to the first window in the next application
- `PREVAPP` moves to the first window in the previous application

The *OblongButton* will respond to the `SELECTKEY` by acting as if the `SELECT` buttons had been clicked.

***Keyboard Mnemonic Display***

The `OblongButton` widget displays its keyboard mnemonic as part of its label. If the mnemonic character is in the label, then that character is highlighted according to the value of the `XtNshowMnemonics` toolkit resource. If the mnemonic character is not in the label, it is displayed to the right of the label in parentheses and highlighted according to the value of the `XtNshowMnemonics` resource.

If label truncation is necessary, the mnemonic displayed in parentheses is truncated as a unit.

***Keyboard Accelerator Display***

The display of keyboard accelerators is controlled by the toolkit resource `XtNshowAccelerators`. When the value of `XtNshowAccelerators` is `OL_DISPLAY`, the `OblongButton` widget displays the keyboard accelerator as part of its label. The string in the `XtNacceleratorText` resource is displayed to the right of the label (or mnemonic) separated by at least one space. The `acceleratorText` is right justified.

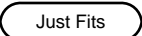

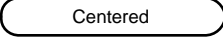


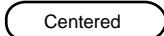
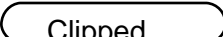
If label truncation is necessary, the accelerator is truncated as a unit. The accelerator is truncated before the mnemonic or the label.

*OblongButton Widget*

**Label Appearance**

The `XtNwidth`, `XtNheight`, `XtNrecomputeSize`, and `XtNlabelJustify` resources interact to produce a truncated, clipped, centered, or left-justified label as shown in the following table.

*Table 8-40 OblongButton Label Appearance*

| XtNwidth           | XtNrecomputeSize | XtNlabelJustify | Result                                                                                |
|--------------------|------------------|-----------------|---------------------------------------------------------------------------------------|
| any value          | TRUE             | any             |    |
| > needed for label | FALSE            | OL_LEFT         |    |
| > needed for label | FALSE            | OL_CENTER       |   |
| < needed for label | FALSE            | any             |  |
| XtNheight          | XtNrecomputeSize | XtNlabelJustify | Result                                                                                |
| any value          | TRUE             | any             |  |
| > needed for label | FALSE            | any             |  |
| < needed for label | FALSE            | any             |  |

When the label is centered or left-justified, the extra space is filled with the background color of the `OblongButton` widget, as determined by the `XtNbackground` and `XtNbackgroundPixmap` resources. When a text label is truncated, the truncation occurs at a character boundary and a solid triangle is inserted to show that part of the label is missing. The triangle requires that more of the label be truncated than would otherwise be necessary. If the width of the button is too small to show even one character with the triangle, only the triangle is shown. If the width is so small that the entire triangle cannot be shown, the triangle is clipped on the right. An image label is simply truncated; no triangle is shown. See also the `XtNlabelTile` resource for how it affects the appearance of a label image.

*Resources*

Table 8-41 OblongButton Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | (calculated)        | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | (calculated)        | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

Table 8-42 OblongButton Primitive Resources

| Name               | Type           | Default             | Access |
|--------------------|----------------|---------------------|--------|
| XtNaccelerator     | String         | NULL                | SGI    |
| XtNacceleratorText | String         | NULL                | SGI    |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO   |
| XtNfont            | OlFont         | XtDefaultFont       | SGID   |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID   |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID   |
| XtNinputFocusColor | Pixel          | Red                 | SGID   |
| XtNmnemonic        | unsigned char  | '\0'                | SGI    |
| XtNreferenceName   | String         | NULL                | GI     |

*OblongButton Widget*

*Table 8-42 OblongButton Primitive Resources (Continued)*

| <b>Name</b>        | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------------|-------------|----------------|---------------|
| XtNreferenceWidget | Widget      | NULL           | GI            |
| XtNscale           | int         | 12             | SGI           |
| XtNtextFormat      | OlStrRep    | OL_SB_STR_REP  | GI            |
| XtNtraversalOn     | Boolean     | TRUE           | SGI           |
| XtNuserData        | XtPointer   | NULL           | SGI           |

*Table 8-43 OblongButton Resources*

| <b>Name</b>      | <b>Type</b>    | <b>Default</b>  | <b>Access</b> |
|------------------|----------------|-----------------|---------------|
| XtNbusy          | Boolean        | FALSE           | SGI           |
| XtNdefault       | Boolean        | FALSE           | SGI           |
| XtNlabel         | OlStr          | (instance name) | SGI           |
| XtNlabelImage    | XImage *       | NULL            | SGI           |
| XtNlabelJustify  | OlDefine       | OL_LEFT         | SGI           |
| XtNlabelTile     | Boolean        | FALSE           | SGI           |
| XtNlabelType     | OlDefine       | OL_STRING       | SGI           |
| XtNrecomputeSize | Boolean        | TRUE            | SGI           |
| XtNselect        | XtCallbackList | NULL            | SGIO          |

***XtNbusy***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCBusy      | Boolean     | FALSE          | SGI           |

**Synopsis:** The “busy” state of the button.

**Values:** TRUE/“true” - The button is stippled as “busy.”. The system will beep if the user attempts to select the button; the attempt is refused and no callbacks are invoked.

FALSE/“false” - The button is not stippled and works normally.



***XtNdefault***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCDefault   | Boolean     | FALSE          | SGI           |

Synopsis: Whether the OblongButton is the default choice in its immediate shell.

Values: TRUE/"true" - If the button is in a menu, an oval ring is drawn around the button to show that the button is the default choice of one or more buttons.

FALSE/"false" - The button is not the default choice.

***XtNlabel***

| <b>Class</b> | <b>Type</b> | <b>Default</b>  | <b>Access</b> |
|--------------|-------------|-----------------|---------------|
| XtCLabel     | OlStr       | (instance name) | SGI           |

Synopsis: The text for the Label.

Values: Any OlStr value valid in the current locale.

This resource is ignored if the XtNlabelType resource has the value OL\_IMAGE.

The OblongButton label is colored using the XtNfontColor resource.

***XtNlabelImage***

| <b>Class</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtCLabelImage | XImage *    | NULL           | SGI           |

Synopsis: The image for the Label.

This resource is ignored unless the XtNlabelType resource has the value OL\_IMAGE. If the image is of type XYBitmap, the image is highlighted when appropriate by reversing the 0 and 1 values of each pixel (that is, by XORing the image data). If the image is of type XYPixmap or ZPixmap, the image is not highlighted, although the space around the image inside the border is.

If the image is smaller than the space available for it inside the border and XtNlabelTile is FALSE, the image is centered vertically and either centered or left-justified horizontally, depending on the value of the XtNlabelJustify resource. If the image is larger than the space available for it, it is clipped so that it does not display outside the border. If the XtNdefault resource is TRUE so that the border is doubled, the space available is that inside the inner line of the border.

*OblongButton Widget*

***XtNlabelJustify***

| Class           | Type     | Default | Access |
|-----------------|----------|---------|--------|
| XtCLabelJustify | OIDefine | OL_LEFT | SGI    |

Synopsis: The justification of the Label within the widget width.

Values: OL\_LEFT/"left" - The Label is left-justified.  
 OL\_CENTER/"center" - The Label is centered.

***XtNlabelTile***

| Class        | Type    | Default | Access |
|--------------|---------|---------|--------|
| XtCLabelTile | Boolean | FALSE   | SGI    |

Synopsis: The tiling of the Label's background.

Values: TRUE/"true" - For an image that is smaller than the subobject's background, the label area is tiled with the image to fill the subobject's background.  
 FALSE/"false" - The label is placed as described by the XtNlabelJustify resource.

This resource augments the XtNlabelImage resource to allow tiling the subobject's background. The XtNlabelTile resource is ignored for text labels.

***XtNlabelType***

| Class        | Type     | Default   | Access |
|--------------|----------|-----------|--------|
| XtCLabelType | OIDefine | OL_STRING | SGI    |

Synopsis: The form that the Label takes.

Values: OL\_STRING/"string" - The label is text.  
 OL\_IMAGE/"image" - The label is an image.  
 OL\_POPUP/"popup" - The label is text followed by an ellipsis.

***XtNrecomputeSize***

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCRecomputeSize | Boolean | TRUE    | SGI    |

Synopsis: Whether the OblongButton widget should calculate its size.

Values: TRUE/"true" - The OblongButton widget will do normal size calculations that may cause its geometry to change, and automatically set the XtNheight and XtNwidth resources.  
 FALSE/"false" - The OblongButton widget will leave its size alone; this may cause truncation of the visible image being shown

*OblongButton Widget*

by the `OblongButton` widget if the fixed size is too small, or may cause padding if the fixed size is too large. The location of the padding is determined by the `XtNlabelJustify` resource.

***XtNselect***

| <b>Class</b> | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|--------------|----------------|----------------|---------------|
| XtCCallback  | XtCallbackList | NULL           | SGIO          |

Synopsis: The callback list invoked when the widget is selected.

***Activation Types***

The following table lists the activation types used by the `OblongButton`.

*Table 8-44* OblongButton Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MENU                | MENU             | XtNmenuBtn           |
| OL_MENUDEFAULT         | MENUDEFAULT      | XtNmenuDefaultBtn    |
| OL_MENUDEFAULTKEY      | MENUDEFAULT      | XtNmenuDefaultKey    |
| OL_MENUKEY             | MENU             | XtNmenuKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_SELECT              | SELECT           | XtNselectBtn         |
| OL_SELECTKEY           | SELECT           | XtNselectKey         |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

---

## OblongButton Widget

### ***OL\_MENU/ OL\_MENUKEY***

The OblongButton only will respond to the OL\_MENU and OL\_MENUKEY activation types if it is a descendant of a Menu widget. When this is the case, the OL\_MENU and OL\_MENUKEY will behave as the OL\_SELECT and OL\_SELECTKEY, respectively.

### ***OL\_MENUDEFAULT/ OL\_MENUDEFAULTKEY***

The OL\_MENUDEFAULT and OL\_MENUDEFAULTKEY activation types apply only to OblongButtons that are descendants of a Menu. These activation types set the OblongButton `XtNdefault` resource to TRUE, and change the display of the widget according to the *OPEN LOOK GUI Functional Specification* section “Changing Menu Defaults” in Chapter 15.

### ***OL\_SELECT/ OL\_SELECTKEY***

The activation of the OblongButton widget with the SELECT button or key will cause the `XtNselect` callback to be called.

## *See Also*

“Exclusives Widget” on page 277  
“FlatExclusives Widget” on page 337  
“FlatNonexclusives Widget” on page 347  
“MenuButton Widget” on page 403  
“Nonexclusives Widget” on page 428  
“RectButton Widget” on page 489.

## PopupWindowShell Widget

### Class

**Class Name:** PopupWindowShell  
**Class Pointer:** popupWindowShellWidgetClass

### Ancestry

Core-Composite-Shell-WMShell-VendorShell-TransientShell-  
 PopupWindowShell

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/PopupWindo.h>
```

### Description

The PopupWindowShell widget is used to implement the OPEN LOOK property window, managing its creation, and providing a simple interface for populating the window with controls. However, the PopupWindowShell has no innate semantics to relate the controls to a selected object; this must be handled by the application. For example, the application must dim all the controls if an object selected by the user is incompatible with a displayed property window.

The decoration of the PopupWindowShell by the Window Manager is controlled by the resources listed in the following table.

*Table 8-45* PopupWindowShell Default Window Decorations

| Resource         | Type     | Default |
|------------------|----------|---------|
| XtNmenuButton    | Boolean  | FALSE   |
| XtNpushpin       | OlDefine | OL_OUT  |
| XtNresizeCorners | Boolean  | FALSE   |
| XtNwindowHeader  | Boolean  | TRUE    |

Components

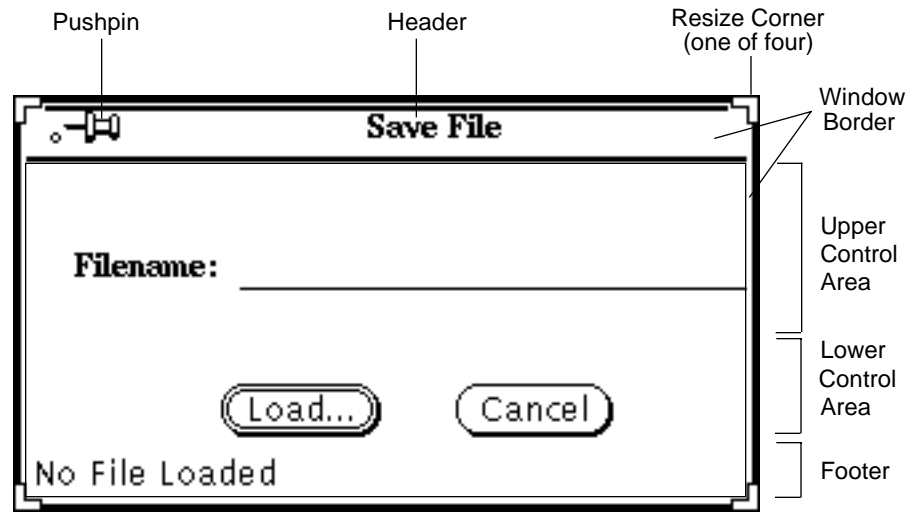


Figure 8-13 PopupWindowShell Components

The PopupWindowShell has the following parts, as shown in the figure.

- Upper control area
- Lower control area
- Window border
- Popup window menu
- Settings menu (conditional)
- Apply button (conditional)
- Reset button (conditional)
- Reset to Factory button (conditional)
- Set Defaults button (conditional)
- Header
- Window mark
- Pushpin (optional)
- Resize corners (optional)
- Footer (optional)

The window border, popup window menu, header, window mark, and pushpin provide the user with window management controls over the PopupWindowShell widget. The Apply, Reset, Reset to Factory, and Set

Defaults Buttons are automatically created by adding an appropriate callback to help create a standard layout of a property window.

### Subwidgets

The PopupWindowShell contains three subwidgets, a FooterPanel and two ControlArea widgets, provided automatically, and accessible through the following resources:

- XtNfooterPanel
- XtNlowerControlArea
- XtNupperControlArea

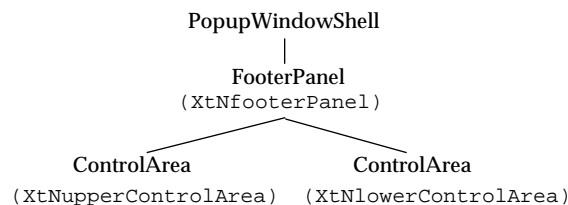


Figure 8-14 PopupWindowShell Subwidgets

### Automatic Addition of Buttons, Settings Menu

To aid in the creation of a property window, the PopupWindowShell has several callbacks typically used in such a popup window, for applying, resetting, etc. For each of these callbacks that the application sets in the argument list used for creation of the PopupWindow, the PopupWindowShell widget automatically creates a button in the lower ControlArea, and the same button in the Settings Menu. For example, if the application specifies a callback for XtNapply, the popup displays an Apply button. If none of the callbacks is defined, no buttons are automatically created and no Settings Menu is created.

If the application is building a command window, it has to create whatever buttons and menus are needed.

### Subclass Shell Widget

The PopupWindowShell widget is a subclass of Shell, so the XtCreatePopupShell() routine is used to create a popup window instead of the normal XtCreateWidget().

---

## *PopupWindowShell Widget*

### ***Popping the Window Up and Down***

The application controls when the `PopupWindowShell` widget is to be displayed or popped up with the `XtPopup()` routine.

The application also has the responsibility for raising a mapped popup window to the front if the user attempts to pop it up and it is already up. This can be accomplished using the `XRaiseWindow()` function.

However, the application cannot control when the `PopupWindowShell` widget is to be popped down, since the user may have pinned it up with the intent that it stays up until he or she dismisses it. The widget itself detects when to pop down: the user clicks `SELECT` on an `OblongButton` widget in the lower `ControlArea`, or the user dismisses the popup window using the `Popup Window Menu` or `pushpin`.

### ***Control Areas***

The upper and lower `ControlAreas` are handled by separate widget interfaces. The application needs to obtain the individual widget IDs for the control areas (*upper\_controlarea* and *lower\_controlarea*) and footer container (*footerarea*) from the `PopupWindowShell` widget.

The two `ControlAreas` and the `Footer` abut so that there is no space between them. An application can control the distance between the controls in the `Control Areas` by setting margins in each area.

If the `PopupWindowShell` widget automatically creates the `Apply`, `Reset`, `Reset to Factory`, or `Set Defaults` Buttons, it puts them in that order in the lower `ControlArea`. No space is left for a missing button. These buttons will also appear before any buttons added to the lower `ControlArea` by the application.



### Coloration

The following diagram illustrates the resources that affect PopupWindowShell coloration.

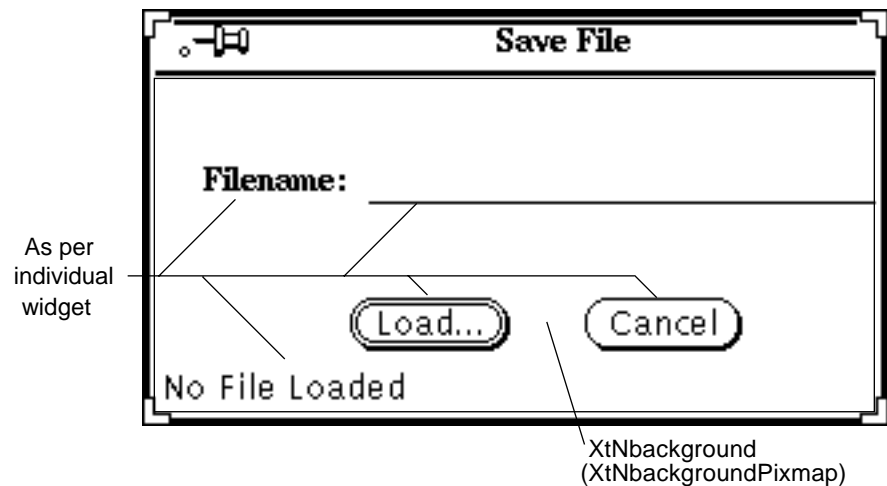


Figure 8-15 PopupWindowShell Coloration

### Keyboard Traversal

The PopupWindowShell widget has a number of components that the user can traverse between. The buttons in the lower ControlArea and in the Settings menu have the following mnemonics:

Table 8-46 PopupWindowShell Traversable Components

| Resource        | Button Name      | Mnemonic |
|-----------------|------------------|----------|
| XtNapply        | Apply            | A        |
| XtNreset        | Reset            | R        |
| XtNresetFactory | Reset to Factory | F        |
| XtNsetDefaults  | Set Defaults     | S        |

These mnemonics will be displayed in the button labels according to the value of the toolkit resource `XtNshowMnemonics` (see page 16). The buttons are created with `XtNtraversalOn` set to TRUE.

*PopupWindowShell Widget*

The TOGGLEPUSHPIN key changes the state of the pushpin in the window header. If the pushpin is in, TOGGLEPUSHPIN will pull the pin out and pop down the window. If the pushpin is out, TOGGLEPUSHPIN will stick the pin in.

*Resources*

*Table 8-47* PopupWindowShell Core Resources

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | GI            |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | 0                   | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 8-48* PopupWindowShell Composite Resources

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

Table 8-49 PopupWindowShell Shell Resources

| Name                    | Type                   | Default    | Access |
|-------------------------|------------------------|------------|--------|
| XtNallowShellResize     | Boolean                | TRUE       | SGI    |
| XtNcreatePopupChildProc | XtCreatePopupChildProc | NULL       | SGI    |
| XtNgeometry             | String                 | NULL       | GI     |
| XtNoverrideRedirect     | Boolean                | FALSE      | SGI    |
| XtNpopdownCallback      | XtCallbackList         | NULL       | SGIO   |
| XtNpopupCallback        | XtCallbackList         | NULL       | SGIO   |
| XtNsaveUnder            | Boolean                | FALSE      | SGI    |
| XtNvisual               | Visual *               | (parent's) | GIO    |

Table 8-50 PopupWindowShell WMShell Resources

| Name            | Type         | Default               | Access |
|-----------------|--------------|-----------------------|--------|
| XtNbaseHeight   | int          | XtUnspecifiedShellInt | SGI    |
| XtNbaseWidth    | int          | XtUnspecifiedShellInt | SGI    |
| XtNheightInc    | int          | XtUnspecifiedShellInt | SGI    |
| XtNiconMask     | Pixmap       | NULL                  | SGI    |
| XtNiconPixmap   | Pixmap       | NULL                  | SGI    |
| XtNiconWindow   | Window       | NULL                  | SGI    |
| XtNiconX        | int          | XtUnspecifiedShellInt | SGI    |
| XtNiconY        | int          | XtUnspecifiedShellInt | SGI    |
| XtNinitialState | InitialState | NormalState           | SGI    |
| XtNinput        | Bool         | FALSE                 | G      |
| XtNmaxAspectX   | int          | XtUnspecifiedShellInt | SGI    |
| XtNmaxAspectY   | int          | XtUnspecifiedShellInt | SGI    |
| XtNmaxHeight    | int          | OL_IGNORE             | SGI    |
| XtNmaxWidth     | int          | OL_IGNORE             | SGI    |
| XtNminAspectX   | int          | XtUnspecifiedShellInt | SGI    |
| XtNminAspectY   | int          | XtUnspecifiedShellInt | SGI    |
| XtNminHeight    | int          | OL_IGNORE             | SGI    |
| XtNminWidth     | int          | OL_IGNORE             | SGI    |

*PopupWindowShell Widget*

*Table 8-50* PopupWindowShell WMShell Resources (Continued)

| <b>Name</b>      | <b>Type</b> | <b>Default</b>        | <b>Accesses</b> |
|------------------|-------------|-----------------------|-----------------|
| XtNtitle         | String      | NULL                  | SGI             |
| XtNtitleEncoding | Atom        | XA_STRING             | SGI             |
| XtNtransient     | Boolean     | TRUE                  | SGI             |
| XtNwaitForWm     | Boolean     | TRUE                  | SGI             |
| XtNwidthInc      | int         | XtUnspecifiedShellInt | SGI             |
| XtNwindowGroup   | Window      | XtUnspecifiedWindow   | SGI             |
| XtNwinGravity    | int         | XtUnspecifiedShellInt | SGI             |
| XtNwmTimeout     | int         | 5000 (msec)           | SGI             |

*Table 8-51* PopupWindowShell VendorShell Resources

| <b>Name</b>             | <b>Type</b>    | <b>Default</b>                            | <b>Access</b> |
|-------------------------|----------------|-------------------------------------------|---------------|
| XtNbusy                 | Boolean        | FALSE                                     | SGI           |
| XtNconsumeEvent         | XtCallbackList | NULL                                      | SGIO          |
| XtNdefaultImName        | String         | NULL                                      | SGI           |
| XtNfooterPresent        | Boolean        | FALSE                                     | SGI           |
| XtNfocusWidget          | Widget         | (see description)                         | SGI           |
| XtNimFontSet            | OIFont         | XtDefaultFontSet                          | SGI           |
| XtNimStatusStyle        | OImStatusStyle | OL_NO_STATUS                              | GI            |
| XtNleftFooterString     | OIStr          | NULL                                      | SGI           |
| XtNleftFooterVisible    | Boolean        | TRUE                                      | SGI           |
| XtNmenuButton           | Boolean        | (see description)                         | GI            |
| XtNmenuType             | OIDefine       | (see description)                         | SGI           |
| XtNpushpin              | OIDefine       | (see description)                         | SGI           |
| XtNresizeCorners        | Boolean        | (see description)                         | SGI           |
| XtNrightFooterString    | OIStr          | NULL                                      | SGI           |
| XtNrightFooterVisible   | Boolean        | TRUE                                      | SGI           |
| XtNshellTitle           | OIStr          | NULL                                      | SGI           |
| XtNuserData             | XtPointer      | NULL                                      | SGI           |
| XtNwindowHeader         | Boolean        | (see description)                         | GI            |
| XtNwmProtocol           | XtCallbackList | NULL                                      | SGIO          |
| XtNwmProtocolInterested | int            | OL_WM_DELETE_WINDOW<br>  OL_WM_TAKE_FOCUS | I             |

*PopupWindowShell Widget**Table 8-52* PopupWindowShell TransientShell Resources

| <b>Name</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-----------------|-------------|----------------|---------------|
| XtNtransientFor | Widget      | NULL           | SGI           |

*Table 8-53* PopupWindowShell Resources

| <b>Name</b>             | <b>Type</b>    | <b>Default</b>     | <b>Access</b> |
|-------------------------|----------------|--------------------|---------------|
| XtNapply                | XtCallbackList | NULL               | I             |
| XtNapplyLabel           | String         | “Apply”            | GI            |
| XtNapplyMnemonic        | unsigned char  | ‘A’                | GI            |
| XtNfooterPanel          | Widget         | (none)             | G             |
| XtNlowerControlArea     | Widget         | (none)             | G             |
| XtNmenuTitle            | String         | “Settings”         | GI            |
| XtNpointerWarping       | Boolean        | TRUE               | SGI           |
| XtNreset                | XtCallbackList | NULL               | I             |
| XtNresetFactory         | XtCallbackList | NULL               | I             |
| XtNresetFactoryLabel    | String         | “Reset To Factory” | GI            |
| XtNresetFactoryMnemonic | unsigned char  | ‘F’                | GI            |
| XtNresetLabel           | String         | “Reset”            | GI            |
| XtNresetMnemonic        | unsigned char  | ‘R’                | GI            |
| XtNsetDefaults          | XtCallbackList | NULL               | I             |
| XtNsetDefaultsLabel     | String         | “Set Defaults”     | GI            |
| XtNsetDefaultsMnemonic  | unsigned char  | ‘S’                | GI            |
| XtNupperControlArea     | Widget         | (none)             | G             |
| XtNverify               | XtCallbackList | NULL               | I             |

*PopupWindowShell Widget*

The following table lists resources passed to the ControlArea subwidget maintained by the PopupWindowShell; they become resources of the ControlArea subwidget and can be set and read just like any other PopupWindowShell resources.

*Table 8-54* PopupWindowShell ControlArea Subwidget Resources<sup>1</sup>

| <b>Name</b>      | <b>Type</b> | <b>Default</b>                                                    | <b>Access</b> |
|------------------|-------------|-------------------------------------------------------------------|---------------|
| XtNalignCaptions | Boolean     | TRUE for upper Control Area;<br>FALSE for lower                   | I             |
| XtNcenter        | Boolean     | FALSE                                                             | I             |
| XtNhPad          | Dimension   | 0                                                                 | I             |
| XtNhSpace        | Dimension   | 0                                                                 | I             |
| XtNlayoutType    | OlDefine    | OL_FIXEDCOLS for upper<br>Control Area;<br>OL_FIXEDROWS for lower | I             |
| XtNmeasure       | int         | 1                                                                 | I             |
| XtNsameSize      | OlDefine    | OL_COLUMNS                                                        | I             |
| XtNvPad          | Dimension   | 0                                                                 | I             |
| XtNvSpace        | Dimension   | 0                                                                 | I             |

1. These subwidget resource are defined in the section “ControlArea Widget” on page 249.

***XtNapply/  
XtNreset/  
XtNresetFactory/  
XtNsetDefaults***

| <b>Class</b> | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|--------------|----------------|----------------|---------------|
| XtCCallback  | XtCallbackList | NULL           | I             |

**Synopsis:** The callback lists invoked for buttons in the lower ControlArea and in the Settings Menu—the Apply, Reset, Reset to Factory, and Set Defaults buttons, respectively.

Typically, an application defines one of these resources only when it is using the PopupWindowShell widget for a property window. For each resource with a defined callback, a unique button is added in the lower ControlArea; conversely, where a resource has no callback defined by the application, no button is shown. The callbacks must be set at initialization time in order for the buttons to be created. The labels for these buttons are listed below, in the order they appear in the lower ControlArea.

*PopupWindowShell Widget*

No space is left for a missing button. In general, the callback list for one of these resources is issued when the user activates the button associated with the resource. After the callbacks are issued, the `PopupWindowShell` widget will attempt to pop itself down, first checking with the application that this may be done by issuing the `XtNverify` callbacks, then checking the state of the pushpin.

***XtNapplyLabel/  
XtNresetLabel/  
XtNresetFactoryLabel/  
XtNsetDefaultsLabel***

| Class                             | Type   | Default            | Access |
|-----------------------------------|--------|--------------------|--------|
| <code>XtCApplyLabel</code>        | String | "Apply"            | GI     |
| <code>XtCResetLabel</code>        | String | "Reset"            | GI     |
| <code>XtCResetFactoryLabel</code> | String | "Reset to Factory" | GI     |
| <code>XtCSetDefaultsLabel</code>  | String | "Set Defaults"     | GI     |

Synopsis: The labels for the Apply, Reset, Reset to Factory, and Set Defaults buttons, respectively.

***XtNapplyMnemonic/  
XtNresetMnemonic/  
XtNresetFactoryMnemonic/  
XtNsetDefaultsMnemonic***

| Class                                | Type          | Default | Access |
|--------------------------------------|---------------|---------|--------|
| <code>XtCApplyMnemonic</code>        | unsigned char | 'A'     | GI     |
| <code>XtCResetMnemonic</code>        | unsigned char | 'R'     | GI     |
| <code>XtCResetFactoryMnemonic</code> | unsigned char | 'F'     | GI     |
| <code>XtCSetDefaultsMnemonic</code>  | unsigned char | 'S'     | GI     |

Synopsis: The mnemonics for the Apply, Reset, Reset to Factory, and Set Defaults buttons, respectively.

Values: Any ASCII character.

***XtNfooterPanel***

| Class                       | Type   | Default | Access |
|-----------------------------|--------|---------|--------|
| <code>XtCFooterPanel</code> | Widget | (none)  | G      |

Synopsis: The widget ID of the FooterPanel composite child widget that handles the footer.

*PopupWindowShell Widget*

The value of this resource is available once the `PopupWindowShell` widget has been created. If the application wants a footer, it can add one to the composite identified by this resource.

***XtNlowerControlArea/  
XtNupperControlArea***

| Class               | Type   | Default | Access |
|---------------------|--------|---------|--------|
| XtCLowerControlArea | Widget | (none)  | G      |
| XtCUpperControlArea | Widget | (none)  | G      |

Synopsis: The widget IDs of the `ControlArea` composite child widgets that handle the lower and upper `ControlAreas`, respectively.

The application can use each widget ID to populate the `PopupWindowShell` with controls. These widget IDs are available once the `PopupWindowShell` widget has been created. Any widgets of the class `OblongButton` added to the lower `ControlArea` are assumed to be window disposition controls; that is, when the user activates one of them the `PopupWindowShell` widget should pop itself down, if allowed by the application and the state of the pushpin.

***XtNmenuItem***

| Class       | Type   | Default    | Access |
|-------------|--------|------------|--------|
| XtCMenuItem | String | "Settings" | GI     |

Synopsis: The title for the conditional `Settings Menu`.

***XtNpointerWarping***

| Class             | Type    | Default | Access |
|-------------------|---------|---------|--------|
| XtCPointerWarping | Boolean | TRUE    | SGI    |

Synopsis: Whether the pointer will jump to the focus widget when the `PopupWindowShell` is popped up.

Values: TRUE/"true" - The pointer jumps to the focus widget (see "XtNfocusWidget" on page 44) when the widget is popped up.  
FALSE/"false" - The pointer doesn't jump.

***XtNverify***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | I      |

Synopsis: The callback list invoked when popping down.



*PopupWindowShell Widget*

The *call\_data* parameter to the callback is a pointer to a variable of type `Boolean`. It is initially set to `TRUE`, and the application should set a value that reflects whether the pop-down is allowed. Typically, the application will use this to prevent a pop-down so that an error message can be displayed. Since more than one callback routine may be registered for this resource, each callback routine can first check the value pointed to by the *call\_data* parameter to see if a previous callback in the list has already rejected the pop-down attempt. If one has, the subsequent callback need not continue evaluating whether a pop-down is allowed. If the value is still `TRUE` after the last callback returns, the pop-down continues. Since these callbacks are issued before the `PopupWindowShell` checks the state of the pushpin, the application should not assume that the pop-down will occur even though it has allowed it.

*Activation Types*

The following table lists the activation types used by the `PopupWindowShell`.

*Table 8-55* `PopupWindowShell` Activation Types

| <b>Activation Type</b>        | <b>Semantics</b>           | <b>Resource Name</b>             |
|-------------------------------|----------------------------|----------------------------------|
| <code>OL_CANCEL</code>        | <code>CANCEL</code>        | <code>XtNcancelKey</code>        |
| <code>OL_DEFAULTACTION</code> | <code>DEFAULTACTION</code> | <code>XtNdefaultActionKey</code> |
| <code>OL_HELP</code>          | <code>HELP</code>          | <code>XtNhelpKey</code>          |
| <code>OL_MOVEDOWN</code>      | <code>MOVEDOWN</code>      | <code>XtNdownKey</code>          |
| <code>OL_MOVELEFT</code>      | <code>MOVELEFT</code>      | <code>XtNleftKey</code>          |
| <code>OL_MOVERIGHT</code>     | <code>MOVERIGHT</code>     | <code>XtNrightKey</code>         |
| <code>OL_MOVEUP</code>        | <code>MOVEUP</code>        | <code>XtNupKey</code>            |
| <code>OL_NEXTFIELD</code>     | <code>NEXTFIELD</code>     | <code>XtNnextFieldKey</code>     |
| <code>OL_PREVFIELD</code>     | <code>PREVFIELD</code>     | <code>XtNprevFieldKey</code>     |
| <code>OL_TOGGLEPUSHPIN</code> | <code>TOGGLEPUSHPIN</code> | <code>XtNtogglePushpinKey</code> |

The `PopupWindowShell` widget has no activation types besides the ones in “Common Activation Types” on page 68.

*See Also*

“ControlArea Widget” on page 249,  
“FooterPanel Widget” on page 381.

## ≡ 8

---

### *PopupWindowShell Widget*

### *RectButton Widget*

#### *Class*

*Class Name:* RectButton  
*Class Pointer:* rectButtonWidgetClass

#### *Ancestry*

Core-Primitive-Button-RectButton

#### *Required Header Files*

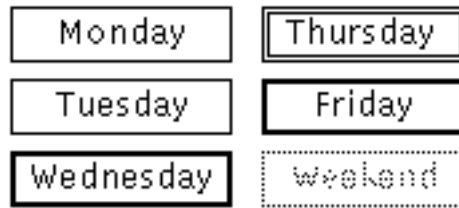
```
#include <Xol/OpenLook>
#include <Xol/RectButton.h>
```

#### *Description*

The RectButton is a toggle button that can be either *set* or *unset* and is designed to be a choice item in a one-of-many or several-of-many choice object. When the user presses SELECT on the button, its state will toggle. The state of the button can also be set programmatically.

**Components**

The figure below shows several OPEN LOOK compliant buttons, in normal, default, and current states (two versions).



*Figure 9-1* **RectButton Rectangular Buttons**

The RectButton widget consists of a Label surrounded by a rectangular border. The border can change to reflect that the button may be a default of several buttons (double border), or represents a current state of an object (thick border), or represents a current state of one of several objects with different states (dimmed border).

The RectButton widget is not intended to be used alone, but rather as a child of either an Exclusives composite widget (to implement one-of-many choice selection—see “Exclusives Widget” on page 277) or a Nonexclusives composite widget (to implement several-of-many choice selection—see “Nonexclusives Widget” on page 428). If the RectButton is created as a child of a different composite widget, proper behavior is not guaranteed.

The following descriptions of using the RectButton widget in a menu assume that the RectButton widget is a child of an Exclusives or Nonexclusives widget, which is a descendant of a MenuShell widget.

**Popup Menu RectButtons**

Entering a RectButton while MENU is depressed changes the appearance of the button from unset to set state or vice versa, to reflect the state the button would be in if MENU were released. Releasing MENU toggles the state associated with the button. Leaving the button before releasing MENU restores the original state appearance and does not toggle the button.

### ***Non-Popup Menu RectButtons***

Clicking SELECT on a RectButton toggles the state associated with it. Pressing SELECT, or moving the pointer into the button while SELECT is pressed, changes the border from unset to set state or vice versa, to reflect the state the button would be in if SELECT were released. Releasing SELECT toggles the state. Moving the pointer off the button before releasing SELECT restores the state appearance and does not toggle the button.

If the button is in a stay-up menu, clicking or pressing MENU works the same as SELECT. If the button is not in a stay-up (or popup) menu, clicking or pressing MENU does not do anything; the event is passed up to an ancestor widget.

### ***Coloration***

For 3D, RectButton coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.” XtNbackground is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1. XtNfontColor is used to draw the label.

For 2D, XtNbackground and XtNfontColor are used to render the RectButton as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, “Controls.”

If the toolkit resource XtNmouseless is set to TRUE and the toolkit resource XtNinputFocusFeedback is set to OL\_INPUT\_FOCUS\_COLOR, then the background of the RectButton will be drawn with the value of XtNinputFocusColor when the widget receives input focus. However, if XtNinputFocusColor is the same as XtNbackground, then the RectButton inverts XtNfontColor and XtNbackground. Once the input focus leaves the widget, the original coloration is restored.

### ***Keyboard Traversal***

The default value of the XtNtraversalOn resource is TRUE.

The RectButton widget responds to the following keyboard navigation keys:

- NEXTFIELD moves to the next traversable widget in the window
- PREVIOUSFIELD moves to the previous traversable widget in the window

## *RectButton Widget*

- MOVEUP moves to the RectButton above the current widget in the Nonexclusives or Exclusives composite
- MOVEDOWN moves to the RectButton below the current widget in the Nonexclusives or Exclusives composite
- LEFT moves to the RectButton to the left of the current widget in the Nonexclusives or Exclusives composite
- MOVERIGHT moves to the RectButton to the right of the current widget in the Nonexclusives or Exclusives composite
- NEXTWINDOW moves to the next window in the application
- PREVWINDOW moves to the previous window in the application
- NEXTAPP moves to the first window in the next application
- PREVAPP moves to the first window in the previous application

The RectButton will respond to the SELECTKEY by acting as if the SELECT buttons had been clicked.

### ***Keyboard Mnemonic Display***

The RectButton widget displays its mnemonic accelerator as part of its label. If the mnemonic character is in the label, then that character is highlighted according to the value of the `XtNshowMnemonics` toolkit resource. If the mnemonic character is not in the label, it is displayed to the right of the label in parenthesis and highlighted according to the value of the `XtNshowMnemonics` resource.

If label truncation is necessary, the mnemonic displayed in parenthesis is truncated as a unit.

### ***Keyboard Accelerator Display***

The display of keyboard accelerators is controlled by the toolkit resource `XtNshowAccelerators`. When the value of `XtNshowAccelerators` is `OL_DISPLAY`, the RectButton widget displays the keyboard accelerator as part of its label. The string in the `XtNacceleratorText` resource is displayed to the right of the label (or mnemonic) separated by at least one space. The `acceleratorText` is right justified.

If label truncation is necessary, the accelerator is truncated as a unit. The accelerator is truncated before the mnemonic or the label.

## Resources

Table 9-1 RectButton Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | (calculated)        | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | (calculated)        | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

Table 9-2 RectButton Primitive Resources

| Name               | Type           | Default             | Access |
|--------------------|----------------|---------------------|--------|
| XtNaccelerator     | String         | NULL                | SGI    |
| XtNacceleratorText | String         | NULL                | SGI    |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO   |
| XtNfont            | OlFont         | XtDefaultFont       | SGID   |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID   |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID   |
| XtNinputFocusColor | Pixel          | Red                 | SGID   |
| XtNmnemonic        | unsigned char  | '\0'                | SGI    |
| XtNreferenceName   | String         | NULL                | GI     |

*RectButton Widget*

*Table 9-2 RectButton Primitive Resources (Continued)*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNscale             | int            | 12             | SGI           |
| XtNtextFormat        | OlStrRep       | OL_SB_STR_REP  | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 9-3 RectButton Resources*

| <b>Name</b>      | <b>Type</b>    | <b>Default</b>  | <b>Access</b> |
|------------------|----------------|-----------------|---------------|
| XtNdefault       | Boolean        | FALSE           | SGI           |
| XtNdim           | Boolean        | FALSE           | SGI           |
| XtNlabel         | OlStr          | (instance name) | SGI           |
| XtNlabelImage    | XImage *       | NULL            | SGI           |
| XtNlabelJustify  | OlDefine       | OL_LEFT         | SGI           |
| XtNlabelTile     | Boolean        | FALSE           | SGI           |
| XtNlabelType     | OlDefine       | OL_STRING       | SGI           |
| XtNrecomputeSize | Boolean        | TRUE            | SGI           |
| XtNselect        | XtCallbackList | NULL            | SGIO          |
| XtNset           | Boolean        | FALSE           | SGI           |
| XtNunselect      | XtCallbackList | NULL            | SGIO          |

***XtNdefault***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCDefault   | Boolean     | FALSE          | SGI           |

**Synopsis:** Whether the RectButton is the default choice in its immediate shell.

**Values:** TRUE/"true" - This button is the default control and it will indicate as such by displaying a border doubled to two lines.  
 FALSE/"false" - The button is not the default choice.



**XtNdim**

| Class  | Type    | Default | Access |
|--------|---------|---------|--------|
| XtCDim | Boolean | FALSE   | SGI    |

Synopsis: Whether the border of the RectButton visually reflects the state of associated objects.

Values: TRUE/"true" - The border is dimmed to show that the RectButton represents the state of one or more of several objects that, as a group, are in different states.  
FALSE/"false" - The border is not dimmed.

**XtNlabel**

| Class    | Type  | Default         | Access |
|----------|-------|-----------------|--------|
| XtCLabel | OlStr | (instance name) | SGI    |

Synopsis: The text for the Label.

Values: Any OlStr value valid in the current locale.

This resource is ignored if the XtNlabelType resource has the value OL\_IMAGE.

The RectButton label is colored using the XtNfontColor resource.

**XtNlabelImage**

| Class         | Type     | Default | Access |
|---------------|----------|---------|--------|
| XtCLabelImage | XImage * | NULL    | SGI    |

Synopsis: The image for the Label.

This resource is ignored unless the XtNlabelType resource has the value OL\_IMAGE. If the image is of type XYBitmap, the image is highlighted when appropriate by reversing the 0 and 1 values of each pixel (that is, by XORing the image data). If the image is of type XYPixmap or ZPixmap, the image is not highlighted, although the space around the image inside the border is highlighted.

If the image is smaller than the space available for it inside the border and XtNlabelTile is FALSE, the image is centered vertically and either centered or left-justified horizontally, depending on the value of the XtNlabelJustify resource. If the image is larger than the space available for it, it is clipped so that it does not display outside the border. If the XtNdefault resource is TRUE

*RectButton Widget*

so that the border is doubled, the space available is that inside the inner line of the border.

***XtNlabelJustify***

| Class           | Type     | Default | Access |
|-----------------|----------|---------|--------|
| XtCLabelJustify | OlDefine | OL_LEFT | SGI    |

Synopsis: The justification of the Label within the widget width.

Values: OL\_LEFT/"left" - The Label is left-justified.  
 OL\_RIGHT/"right" - The Label is right-justified.  
 OL\_CENTER/"center" - The Label is centered.

***XtNlabelTile***

| Class        | Type    | Default | Access |
|--------------|---------|---------|--------|
| XtCLabelTile | Boolean | FALSE   | SGI    |

Synopsis: The tiling of the Label's background.

Values: TRUE/"true" - For an image that is smaller than the subobject's background, the label area is tiled with the image to fill the subobject's background.  
 FALSE/"false" - The label is placed as described by the XtNlabelJustify resource.

This resource augments the XtNlabelImage resource to allow tiling the subobject's background. The XtNlabelTile resource is ignored for text labels.

***XtNlabelType***

| Class        | Type     | Default   | Access |
|--------------|----------|-----------|--------|
| XtCLabelType | OlDefine | OL_STRING | SGI    |

Synopsis: The form that the Label takes.

Values: OL\_STRING/"string" - The label is text.  
 OL\_IMAGE/"image" - The label is an image.

***XtNrecomputeSize***

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCRecomputeSize | Boolean | TRUE    | SGI    |

Synopsis: The resize policy of the widget.

*RectButton Widget*

**Values:** TRUE/"true" - The RectButton widget will do normal size calculations that may cause its geometry to change, and automatically set the XtNheight and XtNwidth resources.  
 FALSE/"false" - The RectButton widget will leave its size alone; this may cause truncation of the visible image being shown by the RectButton widget if the fixed size is too small, or may cause padding if the fixed size is too large. The location of the padding is determined by the XtNlabelJustify resource.

***XtNselect***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

**Synopsis:** The callback list invoked when the widget is selected.

***XtNset***

| Class  | Type    | Default | Access |
|--------|---------|---------|--------|
| XtCSet | Boolean | FALSE   | SGI    |

**Synopsis:** The current state of the button.

**Values:** TRUE/"true" - The button is set (if the toolkit resource XtNthreeD is TRUE, the button will appear pressed-in; otherwise, the border is thickened to indicate the set state).  
 FALSE/"false" - The button is unset.

Simply setting XtNset to TRUE with a call to XtSetValues() does not issue the XtNselect callbacks.

***XtNunselect***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

**Synopsis:** The callback list invoked when the widget is toggled into "unset" mode.


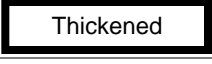

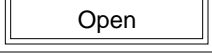
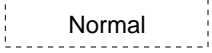
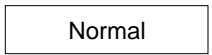
When the RectButton is toggled into "unset" mode by the user to make XtNset become FALSE, the callbacks specified in this resource are activated.

Simply setting XtNset to FALSE with a call to XtSetValues() does not issue the XtNunselect callbacks.

**Border Resource Interactions**

The `XtNdim`, `XtNdefault`, and `XtNset` resources can be set independently; however, all these states cannot be reflected in the visual appearance of the `RectButton`, as shown in the following table.

*Table 9-4* `RectButton` Borders

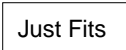
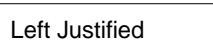
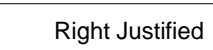
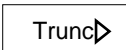

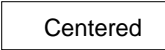

| XtNset | XtNdefault | XtNdim | Border appearance                                                                    |
|--------|------------|--------|--------------------------------------------------------------------------------------|
| TRUE   | TRUE/FALSE | TRUE   |    |
| TRUE   | TRUE/FALSE | FALSE  |    |
| TRUE   | TRUE       | TRUE   |   |
| FALSE  | TRUE       | FALSE  |  |
| FALSE  | TRUE       | TRUE   |  |
| FALSE  | FALSE      | FALSE  |  |

**Label Resource Interactions**

The `XtNwidth`, `XtNheight`, `XtNrecomputeSize`, and `XtNlabelJustify` resources interact to produce a truncated, clipped, centered, or left-justified label as shown in the following table.

When the label is centered or left-justified, the extra space is filled with the background color of the `RectButton` widget, as determined by the `XtNbackground` and `XtNbackgroundPixmap` resources. When the label is truncated, a solid-black triangle is inserted to show that part of the label is missing. The triangle requires that more of the label be truncated than would otherwise be necessary. If the width of the button is too small to show even one character with the triangle, only the triangle is shown. If the width is so small that the entire triangle cannot be shown, the triangle is clipped on the right. See the `XtNlabelTile` resource for how it affects the appearance of a label.

Table 9-5 RectButton Label Appearance

| XtNwidth           | XtNrecomputeSize  | XtNlabelJustify | Result                                                                                |
|--------------------|-------------------|-----------------|---------------------------------------------------------------------------------------|
| any value          | TRUE              | any             |    |
| > needed for label | FALSE             | OL_LEFT         |    |
| > needed for label | FALSE             | OL_CENTER       |    |
| < needed for label | FALSE             | any             |    |
| XtNheight          | XtNrecomputerSize | XtNlabelJustify | Result                                                                                |
| any value          | TRUE              | any             |  |
| > needed for label | FALSE             | any             |  |
| < needed for label | FALSE             | any             |  |

## Activation Types

The following table lists the activation types used by the RectButton.

Table 9-6 RectButton Activation Types

| Activation Type   | Semantics     | Resource Name       |
|-------------------|---------------|---------------------|
| OL_CANCEL         | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION  | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP           | HELP          | XtNhelpKey          |
| OL_MENU           | MENU          | XtNmenuBtn          |
| OL_MENUDEFAULT    | MENUDEFAULT   | XtNmenuDefaultBtn   |
| OL_MENUDEFAULTKEY | MENUDEFAULT   | XtNmenuDefaultKey   |
| OL_MENUKEY        | MENU          | XtNmenuKey          |
| OL_MOVEDOWN       | DOWN          | XtNdownKey          |
| OL_MOVEDOWN       | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT       | LEFT          | XtNleftKey          |
| OL_MOVELEFT       | MOVELEFT      | XtNleftKey          |
| OL_MOVERIGHT      | MOVERIGHT     | XtNrightKey         |

*RectButton Widget*

*Table 9-6 RectButton Activation Types (Continued)*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_MOVERIGHT           | RIGHT            | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_MOVEUP              | UP               | XtNupKey             |
| OL_MULTIDOWN           | JUMP DOWN        | XtNmultiDownKey      |
| OL_MULTILEFT           | JUMP LEFT        | XtNmultiLeftKey      |
| OL_MULTIRIGHT          | JUMP RIGHT       | XtNmultiRightKey     |
| OL_MULTIUP             | JUMP UP          | XtNmultiUpKey        |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_NEXTWINDOW          | NEXTWINDOW       | XtNnextWinKey        |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_PREVWINDOW          | PREVWINDOW       | XtNprevWinKey        |
| OL_SELECT              | SELECT           | XtNselectBtn         |
| OL_SELECTKEY           | SELECT           | XtNselectKey         |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

***OL\_MENU/  
OL\_MENUKEY***

The RectButton only will respond to the OL\_MENU and OL\_MENUKEY activation types if it is a descendant of a Menu widget. When this is the case, the OL\_MENU and OL\_MENUKEY will behave as the OL\_SELECT and OL\_SELECTKEY, respectively.

***OL\_MENUDEFAULT/  
OL\_MENUDEFAULTKEY***

The OL\_MENUDEFAULT and OL\_MENUDEFAULTKEY activation types only apply to RectButtons that are descendants of a Menu. These activation types set the RectButton XtNdefault resource to TRUE, and change the display of the widget according to the *OPEN LOOK GUI Functional Specification* section “Changing Menu Defaults” in Chapter 15.

***OL\_SELECT/  
OL\_SELECTKEY***

The activation of the *RectButton* widget with the *SELECT* button or key will depend on the parent of the *RectButton*: either *Exclusives* or *Nonexclusives*.

The activation of an *Exclusive RectButton* is described in the *OPEN LOOK GUI Functional Specification* section “*Exclusive Settings*” in Chapter 4 and in “*Using Menus*” in Chapter 15. When the user selects a *RectButton* in an *exclusive* setting, the *XtNset* resource will be set to *TRUE* and the *XtNselect* callback will be called. In addition, the *RectButton* in the *exclusive* that was previously set will have the *XtNset* resource changed to *FALSE* and the *XtNunselect* callback will be called.

The activation of a *Nonexclusive RectButton* is described in the *OPEN LOOK GUI Functional Specification* section “*Nonexclusive Settings*” in Chapter 4 and in “*Using Menus*” in Chapter 15. When the user selects a *RectButton* in a *nonexclusive* setting, the state of the *XtNset* resource is reversed. When the *XtNset* resource goes to *FALSE*, the *XtNunselect* callback is called; when the *XtNset* resource goes to *TRUE*, the *XtNselect* callback will be called.

***See Also***

“*Exclusives Widget*” on page 277,  
“*FlatExclusives Widget*” on page 337,  
“*FlatNonexclusives Widget*” on page 347,  
“*MenuButton Widget*” on page 403,  
“*Nonexclusives Widget*” on page 428,  
“*OblongButton Widget*” on page 464.

---

## RubberTile Widget

### RubberTile Widget

#### Class

*Class Name:* RubberTile  
*Class Pointer:* rubberTileWidgetClass

#### Ancestry

Core-Composite-Constraint-Manager-RubberTile

#### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/RubberTile.h>
```

#### Description

The RubberTile is a constraint widget that allows an application to lay out its children either vertically or horizontally, and then assign relative weights to each child so that it absorbs a certain percentage of size changes. If the RubberTile is set with a vertical orientation, then the children will be laid out vertically in column, each spanning the width of the RubberTile. If the RubberTile is laid out horizontally, then the children will be laid out horizontally in a row, each ones height spanning the height of the RubberTile. The RubberTile resizes its children according to a weight assigned to each child. If there are three children with weights of 1, 2, and 3, they resize to get 1/6th, 1/3rd, and 1/2 of the available space respectively.

This widget is very useful when laying out panes in a window. For example, if an application requires three panes laid out vertically, and the top pane is not to absorb any height changes, but the two lower panes are to each absorb half of the height changes, then the application can do the following: create a RubberTile with `XtNOrientation` set to `OL_VERTICAL`, then create each child, assigning each the appropriate `XtNweight` constraint resource value:

```
top pane -> XtNweight = 0
middle pane -> XtNweight = 1
bottom pane -> XtNweight = 1
```



*RubberTile Widget*

top pane's size change percentage =  $0 / (0 + 1 + 1) = 0\%$   
 middle pane's size change percentage =  $1 / (0 + 1 + 1) = 50\%$   
 bottom pane's size change percentage =  $1 / (0 + 1 + 1) = 50\%$

RubberTiles do not enforce any weighting on their children when the widget is created, only when it is resized. During creation, for a horizontal layout, the initial width of the RubberTile widget will be the sum of the width of the children and the initial height will be that of the tallest child. For a vertical layout, the initial height of the RubberTile widget will be the sum of the individual heights and the initial width will be that of the widest child.

An individual RubberTile only supports a single dimensional array, vertical or horizontal. However, an application can produce a two dimensional effect with a matrix of RubberTile widgets. For example, to create the effect of a  $2 \times 4$  matrix, two top level RubberTiles could manage four RubberTiles each.

*Coloration*

The diagram illustrates the resources that affect RubberTile coloration.

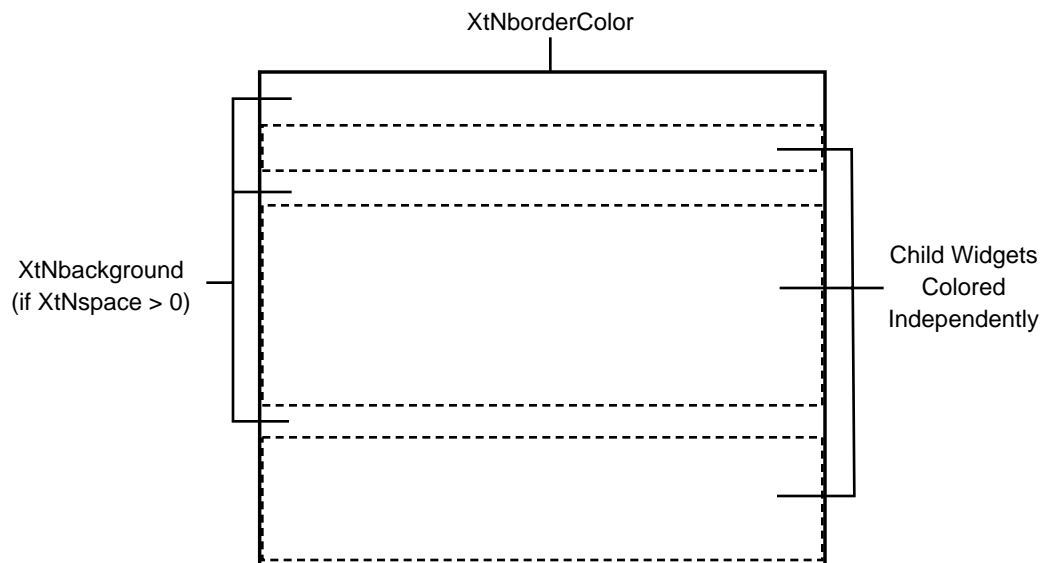


Figure 9-2 RubberTile Coloration

*RubberTile Widget*

*Resources*

*Table 9-7 RubberTile Core Resources*

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | 0                   | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 9-8 RubberTile Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 9-9 RubberTile Manager Resources*

| <b>Name</b>        | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|--------------------|----------------|----------------|---------------|
| XtNconsumeEvent    | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor | Pixel          | Red            | SGID          |

Table 9-9 RubberTile Manager Resources (Continued)

| Name                 | Type           | Default | Access |
|----------------------|----------------|---------|--------|
| XtNreferenceName     | String         | NULL    | GI     |
| XtNreferenceWidget   | Widget         | NULL    | GI     |
| XtNtraversalOn       | Boolean        | TRUE    | SGI    |
| XtNunrealizeCallback | XtCallbackList | NULL    | SGIO   |
| XtNuserData          | XtPointer      | NULL    | SGI    |

Table 9-10 RubberTile Resources

| Name           | Type     | Default     | Access |
|----------------|----------|-------------|--------|
| XtNorientation | OlDefine | OL_VERTICAL | SGI    |

Each child widget attached to the RubberTile composite widget is constrained by the resources in the following table. These resources become resources for each child widget and can be set and read just like any other resources defined for the child.

Table 9-11 RubberTile Constraint Resources

| Name         | Type      | Default | Access |
|--------------|-----------|---------|--------|
| XtNrefName   | String    | NULL    | SGI    |
| XtNrefWidget | XtPointer | NULL    | SGI    |
| XtNspace     | Dimension | 0       | SGI    |
| XtNweight    | Dimension | 1       | SGI    |

**XtNorientation**

| Class          | Type     | Default     | Access |
|----------------|----------|-------------|--------|
| XtCOrientation | OlDefine | OL_VERTICAL | SGI    |

Synopsis: The orientation of the widget.

Values: OL\_VERTICAL/"vertical" - Arrange the child widgets as a single vertical column.  
 OL\_HORIZONTAL/"horizontal" - Arrange the child widgets as a single horizontal row.

*RubberTile Widget*

***XtNrefName***

| Class      | Type   | Default | Access |
|------------|--------|---------|--------|
| XtCRefName | String | NULL    | SGI    |

Synopsis: The reference widget by name.

Values: The name of the child of the RubberTile.

When a child is positioned by the RubberTile widget, it is placed geometrically after (below or right of) the widget referenced by *XtNrefName* or *XtNrefWidget*.

The reference named by this resource will be resolved no sooner than when the children are managed by the RubberTile widget, so that a client can use forward referencing of children.

***XtNrefWidget***

| Class        | Type      | Default | Access |
|--------------|-----------|---------|--------|
| XtCRefWidget | XtPointer | NULL    | SGI    |

Synopsis: The reference widget by widget ID.

Values: The ID of an existing child of the RubberTile.

If both *XtNrefName* and *XtNrefWidget* are given, they must agree.

***XtNspace***

| Class    | Type      | Default | Access |
|----------|-----------|---------|--------|
| XtCSpace | Dimension | 0       | SGI    |

Synopsis: The amount of space between the child and its reference widget (the widget identified by *XtNrefName* or *XtNrefWidget*).

Values:  $0 \leq XtNspace$

**Note** - The *XtNspace* constraint resource conflicts with the Caption widget resource of the same name. Thus, setting this resource for a Caption widget that is a child of a RubberTile will affect both of the widgets.

***XtNweight***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCWeight    | Dimension   | 1              | SGI           |

Synopsis: The weight that dictates how much of a resize is applied to the child.

Values:  $0 \leq \text{XtNweight}$

***Activation Types***

The following table lists the activation types used by the RubberTile.

*Table 9-12 RubberTile Activation Types*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The RubberTile widget has no activation types besides the ones in “Common Activation Types” on page 68.

***See Also***

“Form Widget” on page 385.

---

*Scrollbar Widget**Scrollbar Widget**Class*

*Class Name:* Scrollbar  
*Class Pointer:* scrollbarWidgetClass

*Ancestry*

Core-Primitive-Scrollbar

*Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/Scrollbar.h>
```

*Description*

The Scrollbar widget is similar to the slider widget, but provides additional features. It is typically used with a pane when the pane's contents exceed its size. The user can adjust the pane's view of the contents by manipulating the Scrollbar.

*Components*

Each full scrollbar has the following parts:

- Cable, indicating the extent of scrolling
- Anchors, top (left) and bottom (right), located at both ends of the cable. They are used to move the view to the corresponding extreme of the item or list of items being viewed.
- Elevator, which slides along the length of the cable, containing
  - Arrows, up (left) and down (right), used to move the view in the direction of the arrow by one unit.
  - Drag area, for moving the view by tracking the position of the mouse pointer relative to the scrollbar.

- Proportion indicator, which moves along with the elevator to indicate the size of the view and its position relative to the entire item or list of items being viewed.
- Page indicator (optional), located next to the drag area, which indicates the page number of the content being viewed. The page indicator will be displayed only when the SELECT button is pressed in the drag area.
- Scrollbar menu.

Because a scrollbar can be seen and used horizontally as well as vertically, the top anchor and bottom anchor have the aliases left anchor and right anchor, respectively.

Each scrollbar is associated with a content, as defined by the application. The content is composed of units (e.g. lines of text) that are visible in a viewing area. For a scrollbar to be useful, the content typically has more units than can fit in the viewing area. Hence, “scrolling” the content brings units into view as other units move out of view. The amount of the Content that is visible at one time is called a *pane* in the descriptions below.

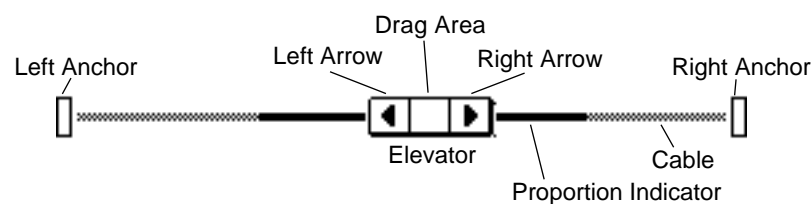


Figure 9-3 Scrollbar Horizontal Orientation

Figure 9-4

### Subwidgets

The Scrollbar contains one subwidget: a *MenuShell* created automatically, and accessible through the `XtNmenuPane` resource.

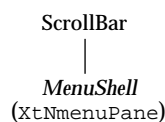


Figure 9-5 Scrollbar Subwidgets

*Scrollbar Widget*

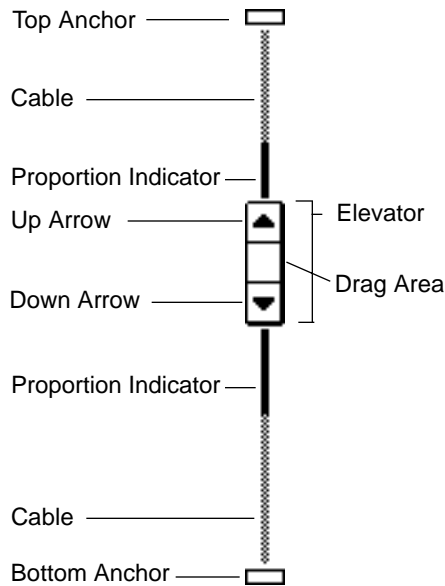


Figure 9-6 Scrollbar Vertical Orientation

**Abbreviated Scrollbar**

The Scrollbar widget responds to a parent’s request to resize smaller by shortening the cable (and proportion indicator), but leaving the other elements full-sized. The Scrollbar widget will eliminate the cable entirely, if necessary, to meet a resize request. These abbreviated scrollbars are shown in the following figure. If necessary, the Scrollbar widget will eliminate the anchors (in addition to the cable and drag area) to meet a resize request to form a minimum Scrollbar.



Figure 9-7 Abbreviated Scrollbars



***Elevator Motion***

As visual feedback to the user, the elevator moves up and down (or left and right) along the line of the cable as the content scrolls or changes panes.

The range of motion of the elevator is not necessarily the full distance between the anchors. The application decides how far the elevator can be moved by evaluating each attempt to move it.

The user manipulates the scrollbar by pressing or clicking SELECT. The action performed depends on the position of the pointer and whether the application is willing to scroll the content.

***Scrolling One Unit***

Clicking SELECT on one of the arrows moves the elevator in the direction of the arrow, moves the pointer to stay on the arrow, and changes the content to move one unit out of view and another unit into view, such that the view scrolls in the opposite direction of the elevator motion.

If the application cannot scroll at this time, the elevator and pointer do not move and the view does not change.

Pressing SELECT on an arrow repeats the action described above. When SELECT is clicked or pressed, the arrow highlights while the scrolling action takes place. The highlighting stays until SELECT is released.

When the elevator has reached the end of the cable, the arrow in that direction is made inactive.

***Scrolling Several Units***

Dragging SELECT on the drag area moves the elevator along the cable to track the component of the pointer motion parallel to the cable. The content scrolls in the opposite direction, bringing one or more units into view as other units move out of view.

If granularity is enforced and the elevator is moved to a position that represents a non-integral number of units, the closest integral number of units is considered instead. If granularity is not enforced, the elevator is moved by the non-integral number of units. The `XtNsliderMoved` callback allows the application to enforce granularity.

When the application reaches the limit that it can scroll, the view no longer changes and the elevator stops moving. While dragging SELECT, the drag area highlights. The pointer is constrained to stay within the Drag Area as the elevator moves.

### ***Scrolling Limits***

Clicking SELECT on one of the anchors causes the view of the Content to change to the corresponding pane, and moves the elevator to the limit in the direction of the anchor. If the elevator is already at the limit, nothing happens.

Clicking SELECT on an anchor highlights the anchor while the scrolling action takes place.

### ***Scrolling a Pane***

Clicking SELECT on the cable above/left-of or below/right-of the elevator causes the view of the Content to change to the previous or next pane, respectively. The pointer is moved along the direction of the elevator travel to keep it off the elevator.

If only a partial pane remains before the limit of the Content is reached, the effect is as if the user clicked SELECT on the corresponding anchor. If the application cannot move to another pane, the view does not change, the elevator and pointer do not move.

Pressing SELECT on the Cable repeats the action described above.

### ***Elevator at Limits***

The application calibrates the scrollbar so that the position of the elevator on the scrollbar is in units useful to the application. In general, these units will not be pixels or points.

If the scrollbar is close enough to an anchor, the separation in application units may be zero pixels, because of the discrete nature of pixels. Here, the elevator is kept away from the anchor so that two points of the cable length are visible. The elevator is placed at the limit of motion only when the user explicitly moves the elevator to an anchor by clicking SELECT on the anchor, or drags the elevator until it reaches the limit.

***Indicating View Proportion***

The proportion indicator gives a gross measure of what part of the content is in view. Its size relative to the length of the cable is the same as the size of the pane relative to the size of the content. However, the scrollbar widget does not maintain this relation but relies on the application to provide the length of the proportion indicator.

The proportion indicator moves with the elevator such that both reach the limits together. When the content is scrolled to the beginning, the proportion indicator and the elevator align at the left or top end of the scrollbar. When the content is scrolled to the end, the proportion indicator and the elevator align at the right or bottom end of the scrollbar. For intermediate positions, the elevator is positioned proportionally between the ends of the proportion indicator. Thus, as the content is scrolled at a constant rate (e.g., by dragging SELECT), the elevator creeps from one end of the Proportion Indicator to the other at a constant rate.

***Scrollbar Menu***

The scrollbar menu (not shown in the figures) pops up when the user presses MENU anywhere over the Scrollbar widget. The menu has three default choices depending on the Scrollbar orientation.

|                  |                                                                                                                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Here to top/left | Scrolls the content so that the unit next to the pointer is placed at the top or left of the viewing area.                                                                                       |
| Top/left to here | Scrolls the content so that the unit at the top or left of the viewing area is placed next to the pointer.                                                                                       |
| Previous         | Scrolls the content to restore the previous view. The Scrollbar widget remembers only the last two scroll positions, so repeated access of this choice alternates the content between two views. |

An application can add choices to this menu, using the same technique for populating other menus (see “MenuShell Widget” on page 414). The ID of the Menu widget is available as a resource of the Scrollbar.

### ***Coloration***

For 3D and 2D, the area surrounding the Scrollbar is drawn with the parent's `XtNbackground`. `XtNforeground` is used to draw the optional page number indicator value.

For 3D, the Scrollbar coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNbackground` and `XtNforeground` are used to render the Scrollbar as described by the *OPEN LOOK GUI Functional Specification*, Chapter 7, "Scrolling."

### ***Keyboard Traversal***

The Scrollbar's default value of the `XtNtraversalOn` resource is FALSE. If the application sets it to TRUE, undefined results occur.

### ***Scrollbar Menu***

The default choices in the Scrollbar Menu are created with `XtNtraversalOn` set to TRUE and `XtNmnemonic` set to the first character of their label.

When the Scrollbar menu is posted via keyboard traversal, the "Here to top" and "Top to here" buttons are not sensitive. These buttons depend on the position of the pointer when the menu is posted, and so they are not applicable when the menu is posted from the keyboard.

### ***Keyboard Mnemonic Display***

The Scrollbar does not display the mnemonic accelerator. If the Scrollbar is the child of a Caption widget, the Caption widget can be used to display the mnemonic for the Scrollbar.

### ***Keyboard Accelerator Display***

The Scrollbar does not display or respond to a keyboard accelerator because clicking the SELECT button on a Scrollbar activates depending on the pointer position.

*Resources**Table 9-13 Scrollbar Core Resources*

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | 0                   | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 9-14 Scrollbar Primitive Resources*

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|--------------------|----------------|---------------------|---------------|
| XtNaccelerator     | String         | NULL                | SGI           |
| XtNacceleratorText | String         | NULL                | SGI           |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID          |
| XtNinputFocusColor | Pixel          | Red                 | SGID          |
| XtNmnemonic        | unsigned char  | '\0'                | SGI           |
| XtNreferenceName   | String         | NULL                | GI            |

Scrollbar Widget

Table 9-14 Scrollbar Primitive Resources (Continued)

| Name               | Type      | Default       | Access |
|--------------------|-----------|---------------|--------|
| XtNreferenceWidget | Widget    | NULL          | GI     |
| XtNscale           | int       | 12            | SGI    |
| XtNtextFormat      | OlStrRep  | OL_SB_STR_REP | GI     |
| XtNtraversalOn     | Boolean   | FALSE         | SGI    |
| XtNuserData        | XtPointer | NULL          | SGI    |

Table 9-15 Scrollbar Resources

| Name                  | Type           | Default                       | Access |
|-----------------------|----------------|-------------------------------|--------|
| XtNcurrentPage        | int            | 1                             | SGI    |
| XtNdragCBType         | OlDefine       | OL_CONTINUOUS                 | SGI    |
| XtNgranularity        | int            | 1                             | SGI    |
| XtNhereToLeftLabel    | OlStr          | "Here To Left"                | GI     |
| XtNhereToLeftMnemonic | unsigned char  | '\0'                          | GI     |
| XtNhereToTopLabel     | OlStr          | "Here To Top"                 | GI     |
| XtNhereToTopMnemonic  | unsigned char  | '\0'                          | GI     |
| XtNinitialDelay       | int            | 500                           | SGI    |
| XtNleftToHereLabel    | OlStr          | "Left To Here"                | GI     |
| XtNleftToHereMnemonic | unsigned char  | '\0'                          | GI     |
| XtNmenuPane           | Widget         | NULL                          | G      |
| XtNmenuTitle          | OlStr          | "Scrollbar"                   | GI     |
| XtNorientation        | OlDefine       | OL_VERTICAL                   | GI     |
| XtNpointerWarping     | Boolean        | TRUE                          | SGI    |
| XtNpreviousLabel      | OlStr          | "Previous"                    | GI     |
| XtNpreviousMnemonic   | unsigned char  | '\0'                          | GI     |
| XtNproportionLength   | int            | (XtNsliderMax - XtNsliderMin) | SGI    |
| XtNrepeatRate         | int            | 100                           | SGI    |
| XtNshowPage           | OlDefine       | OL_NONE                       | SGI    |
| XtNsliderMax          | int            | 100                           | SGI    |
| XtNsliderMin          | int            | 0                             | SGI    |
| XtNsliderMoved        | XtCallbackList | NULL                          | SGIO   |
| XtNsliderValue        | int            | 0                             | SGI    |
| XtNstopPosition       | OlDefine       | OL_ALL                        | SGI    |

Table 9-15 Scrollbar Resources (Continued)

| Name                 | Type          | Default       | Access |
|----------------------|---------------|---------------|--------|
| XtNtopToHereLabel    | OlStr         | "Top To Here" | GI     |
| XtNtopToHereMnemonic | unsigned char | '\0'          | GI     |
| XtNuseSetValCallback | Boolean       | FALSE         | SGI    |

The following table lists resources passed to the Menu subwidget maintained by the Scrollbar. They become resources of the Menu subwidget and can be set and read just like any other resources for the Scrollbar.

Table 9-16 Scrollbar Subwidget Resources<sup>1</sup>

| Name              | Type      | Default      | Access |
|-------------------|-----------|--------------|--------|
| XtNcenter         | Boolean   | TRUE         | I      |
| XtNhPad           | Dimension | 4            | I      |
| XtNhSpace         | Dimension | 4            | I      |
| XtNlayoutType     | OlDefine  | OL_FIXEDROWS | I      |
| XtNmeasure        | int       | 1            | I      |
| XtNpushpin        | OlDefine  | OL_NONE      | I      |
| XtNpushpinDefault | Boolean   | FALSE        | I      |
| XtNsameSize       | OlDefine  | OL_COLUMNS   | I      |
| XtNvPad           | Dimension | 4            | I      |
| XtNvSpace         | Dimension | 4            | I      |

1. These subwidget resources are described in the sections "ControlArea Widget" on page 249 and "MenuShell Widget" on page 414.

### ***XtNcurrentPage***

| Class          | Type | Default | Access |
|----------------|------|---------|--------|
| XtCCurrentPage | int  | 1       | SGI    |

**Synopsis:** The page number displayed when the `XtNshowPage` resource is set to `OL_RIGHT` or `OL_LEFT`.

**Values:**  $1 \leq \text{XtNcurrentPage}$

If `XtNshowPage` is set via `XtSetValues()`, or if `XtNshowPage` is already set and later `XtNsliderValue` is changed via `SetValues()`, it is the responsibility of the application to update `XtNcurrentPage` appropriately.

*Scrollbar Widget*

***XtNdragCBType***

| Class         | Type     | Default       | Access |
|---------------|----------|---------------|--------|
| XtCDragCBType | OlDefine | OL_CONTINUOUS | SGI    |

Synopsis: The frequency of issuing XtNsliderMoved callbacks during a drag operation.

Values: OL\_CONTINUOUS/"continuous" - Issue callbacks continuously.  
 OL\_GRANULARITY/"granularity" - Issue callbacks only when the drag box crosses any granularity positions.  
 OL\_RELEASE/"release" - Issue callbacks only once when the SELECT button is released.

***XtNgranularity***

| Class          | Type | Default | Access |
|----------------|------|---------|--------|
| XtCGranularity | int  | 1       | SGI    |

Synopsis: The distance the elevator attempts to move when clicking or pressing SELECT on an arrow.

Values:  $1 \leq \text{XtNgranularity} \leq \text{XtNsliderMax} - \text{XtNsliderMin}$

Normally, the drag operation does not honor granularity unless enforcement is set in the XtNsliderMoved callback procedure.

***XtNhereToLeftLabel/  
 XtNhereToTopLabel/  
 XtNleftToHereLabel/  
 XtNpreviousLabel/  
 XtNtopToHereLabel***

| Class              | Type  | Default        | Access |
|--------------------|-------|----------------|--------|
| XtCHereToLeftLabel | OlStr | "Here to Left" | GI     |
| XtCHereToTopLabel  | OlStr | "Here to Top"  | GI     |
| XtCLeffToHereLabel | OlStr | "Left to Here" | GI     |
| XtCPreviousLabel   | OlStr | "Previous"     | GI     |
| XtCTopToHereLabel  | OlStr | "Top to Here"  | GI     |

Synopsis: The labels for the various menu buttons.

Values: Any OlStr values valid in the current locale.



***XtNhereToLeftMnemonic/  
XtNhereToTopMnemonic/  
XtNleftToHereMnemonic/  
XtNpreviousMnemonic/  
XtNtopToHereMnemonic***

| Class                 | Type          | Default | Access |
|-----------------------|---------------|---------|--------|
| XtCHereToLeftMnemonic | unsigned char | '\0'    | GI     |
| XtCHereToTopMnemonic  | unsigned char | '\0'    | GI     |
| XtCLeftToHereLabel    | unsigned char | '\0'    | GI     |
| XtCPreviousLabel      | unsigned char | '\0'    | GI     |
| XtCTopToHereLabel     | unsigned char | '\0'    | GI     |

Synopsis: The mnemonics for the various menu buttons.

Values: Any ASCII character.

***XtNinitialDelay***

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCInitialDelay | int  | 500     | GI     |

Synopsis: The time, in milliseconds, before the first action occurs when SELECT is pressed on the cables or arrows.

Values:  $0 < XtNinitialDelay$

***XtNmenuPane***

| Class       | Type   | Default | Access |
|-------------|--------|---------|--------|
| XtCMenuPane | Widget | NULL    | G      |

Synopsis: The widget where the scrollbar menu items are attached.

Values: The ID of the menu widget associated with the scrollbar.

This value is available once the Scrollbar has been created.

The default scrollbar menu contains the following items:

- Here To Top (Here To Left)
- Top To Here (Left To Here)
- Previous

The application must not remove these items from the menu.

*Scrollbar Widget*

The scrollbar menu can be enhanced by adding new menu items to this widget. Items can be added just as they are added to the menupane for a MenuShell or MenuButton widget.

***XtNmenuTitle***

| Class        | Type  | Default     | Access |
|--------------|-------|-------------|--------|
| XtCMenuTitle | OlStr | "Scrollbar" | GI     |

Synopsis: The title of the Scrollbar menu.

Values: Any OlStr value valid in the current locale.

***XtNorientation***

| Class          | Type     | Default     | Access |
|----------------|----------|-------------|--------|
| XtCOrientation | OlDefine | OL_VERTICAL | GI     |

Synopsis: The direction of the visual presentation of the widget.

Values: OL\_HORIZONTAL/"horizontal" - Define a horizontal scrollbar.  
 OL\_VERTICAL/"vertical" - Define a vertical scrollbar.

This resource cannot be changed via XtSetValues().

***XtNpointerWarping***

| Class             | Type    | Default | Access |
|-------------------|---------|---------|--------|
| XtCPointerWarping | Boolean | TRUE    | SGI    |

Synopsis: Whether the pointer will move along with the elevator when SELECT is pressed on one of the arrows.

Values: TRUE/"true" - The pointer moves with the elevator.  
 FALSE/"false" - The pointer doesn't move with the elevator.

***XtNproportionLength***

| Class               | Type | Default                       | Access |
|---------------------|------|-------------------------------|--------|
| XtCProportionLength | int  | (XtNsliderMax - XtNsliderMin) | SGI    |

Synopsis: The size of the proportion indicator.

Values:  $1 \leq \text{XtNproportionLength} \leq (\text{XtNsliderMax} - \text{XtNsliderMin})$

The application uses the XtNsliderMax and XtNsliderMin resources to calibrate the scrollbar, making its overall length correspond to the overall length of the content, and uses the XtNproportionLength resource to indicate how much of the content is visible. While this resource gives the

*Scrollbar Widget*

overall length of the proportion indicator, the elevator always covers part of it. If the elevator would completely hide the proportion indicator, 3-point sections of it are shown above and below (or left of and right of) the elevator. If the elevator is too close to an anchor to show all of a 3-point section, as much as possible of the section is shown on that side (this may be a zero-length section).

***XtNrepeatRate***

| Class         | Type | Default | Access |
|---------------|------|---------|--------|
| XtCRepeatRate | int  | 100     | SGI    |

Synopsis: The time in milliseconds between repeated actions when SELECT is pressed on the cables or arrows.

Values:  $0 \leq \text{XtNrepeatRate}$

***XtNshowPage***

| Class       | Type     | Default | Access |
|-------------|----------|---------|--------|
| XtCShowPage | OlDefine | OL_NONE | SGI    |

Synopsis: The position of the page indicator.

Values: OL\_LEFT/"left" - The page indicator is to the left of the drag box.

OL\_RIGHT/"right" - The page indicator is to the right of the drag box.

OL\_NONE/"none" - The page indicator is not displayed.

The page indicator is displayed only if `XtNOrientation` is `OL_VERTICAL`. The page indicator is popped up when the scrollbar is dragged by its drag area. While dragging, the page indicator is updated constantly. If `XtNshowPage` changes from `OL_NONE` to any other value, a popup window for the page indicator is created. If the value changes to `OL_NONE`, the popup window is destroyed.

***XtNsliderMax/  
XtNsliderMin***

| Class        | Type | Default | Access |
|--------------|------|---------|--------|
| XtCSliderMax | int  | 100     | SGI    |
| XtCSliderMin | int  | 0       | SGI    |

Synopsis: The calibration of the scrollbar.

Values:  $\text{XtNsliderMin} < \text{XtNsliderMax}$

*Scrollbar Widget*

An application should set the values of these resources to correspond to the range of the content, and should set the value of the `XtNproportionLength` resource to the length of the view into the content. This calibrates the scrollbar.

The Scrollbar uses the calibration to convert the pixel location of the elevator into a value in the range represented by the length of the Content.

The explanation for this range relation follows: First, an application calibrates the scrollbar as described above, so that `XtNsliderMin` and `XtNsliderMax` span the length of the Content and `XtNproportionLength` gives the length of the view of the Content. That is,

```
XtNsliderMin = start of Content
XtNsliderMax = length of Content
XtNproportionLength = length of Pane
```

Consider that the Elevator tracks a fixed position in the view; the position is arbitrary, but remains the same as the view is scrolled over the Content. This can be the first line in the view.

When the view is at the top of the content, the elevator is at the top of the scrollbar and the calibrated position of the first line is `XtNsliderMin`. However, when the view is at the bottom of the content, the elevator is at the bottom of the scrollbar and the calibrated position of the first line is `XtNsliderMax - XtNproportionLength`.

***XtNsliderMoved***

| Class          | Type           | Default | Access |
|----------------|----------------|---------|--------|
| XtCSliderMoved | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback lists used when the scrollbar is manipulated.

The Scrollbar widget passes the final location of the Elevator, as an integer between `XtNsliderMin` and `XtNsliderMax` inclusive, in a structure pointed to by the *call\_data* parameter:

```
typedef struct _OlScrollbarVerify {
 int new_location;
 int new_page;
 Boolean ok;
 int slidermin;
 int slidermax;
```

```
 int delta;
 Boolean more_cb_pending;
} OlScrollbarVerify;
```

|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>new_location</i>    | When the <code>XtNsliderMoved</code> callbacks are made, the <i>new_location</i> member gives the position of the attempted scroll. This will be the new value of the <code>XtNsliderValue</code> resource if the scroll attempt is successful; however, the <code>XtNsliderValue</code> resource is not updated until after the callbacks return.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <i>new_page</i>        | Used to set the page number. To see the page number, the application must set <code>XtNshowPage</code> to <code>OL_LEFT</code> or <code>OL_RIGHT</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <i>ok</i>              | Initially set to <code>TRUE</code> ; the application sets the value to indicate whether the scroll attempt is allowed. Since more than one callback routine may be registered for these resources, each callback routine can first check the <i>ok</i> member to see if a previous callback routine in the list has already rejected the scroll attempt. The Scrollbar will complete the scroll attempt only if, after the last callback has returned, the <i>ok</i> member is still <code>TRUE</code> .<br><br>If <i>ok</i> is <code>FALSE</code> after the last callback returns, the Scrollbar restores the Elevator to the position it was in before the user attempted to move it. This is required only when the Elevator has been dragged. The Scrollbar does not move the Elevator for other scrollbar manipulations until the scroll attempt has been verified. |
| <i>slidermin</i>       | The same value as in the <code>XtNsliderMin</code> resource.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>slidermax</i>       | The same value as in the <code>XtNsliderMax</code> resource.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>delta</i>           | The distance between the new scroll position and the old, as a signed value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>more_cb_pending</i> | Specifies if there are more callbacks pending.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

A callback can change the *new\_location* value to reflect a partial scroll. For example, if the scrolling granularity causes a scroll attempt past the end of an application's partially full buffer, the application should adjust *new\_location* to a value representing the end of the buffer. The adjusted value must lie between the values present before the attempted scroll and the new values given in the `OlScrollbarVerify` structure.

The `XtNsliderMoved` callbacks are issued when the elevator position has been conditionally changed by the user

*Scrollbar Widget*

- clicking or pressing SELECT on the up/left or down/right arrow buttons;
- moving the Elevator to a new position by dragging SELECT on the drag area;
- clicking SELECT on the top/left or bottom/right anchors;
- clicking or pressing SELECT on the Cable.

***XtNsliderValue***

| Class          | Type | Default | Access |
|----------------|------|---------|--------|
| XtCSliderValue | int  | 0       | SGI    |

Synopsis: The current position of the elevator.

Values: XtNsliderMin ≤ XtNsliderValue ≤ XtNsliderMax - XtNproportionLength

The Scrollbar widget keeps this resource up to date; however, an application can also get the current value through the XtNsliderMoved callbacks.

***XtNstopPosition***

| Class           | Type     | Default | Access |
|-----------------|----------|---------|--------|
| XtCStopPosition | OlDefine | OL_ALL  | SGI    |

Synopsis: The disposition of the drag box at the end of a drag operation.

Values: OL\_ALL/"all" - Upon the release of the SELECT button in a drag operation, the drag box will be positioned at where it stops.  
 OL\_GRANULARITY/"granularity" - The drag box will snap to the nearest granularity position.

***XtNuseSetValCallback***

| Class                | Type    | Default | Access |
|----------------------|---------|---------|--------|
| XtCUseSetValCallback | Boolean | FALSE   | SGI    |

Synopsis: The callback action with programmatic control of slider value.

Values: TRUE/"true" - The callbacks specified in XtNsliderMoved are called when the slider value changes, even if it changes under program control instead of as the result of mouse action.  
 FALSE/"false" - The callbacks specified in XtNsliderMoved are not called when the slider value is changed programmatically.

The callbacks listed in XtNsliderMoved are called as the result of mouse action by the user changing the slider value. This resource allows those callbacks to be called when the slider value changes under program control as well as by mouse action.

## Activation Types

The following table lists the activation types used by the Scrollbar.

Table 9-17 Scrollbar Activation Types

| Activation Type    | Semantics      | Resource Name       |
|--------------------|----------------|---------------------|
| OL_CANCEL          | CANCEL         | XtNcancelKey        |
| OL_DEFAULTACTION   | DEFAULTACTION  | XtNdefaultActionKey |
| OL_HELP            | HELP           | XtNhelpKey          |
| OL_HSBMENU         | ALT+CTRL R     | XtNhorizSBMenuKey   |
| OL_MENU            | MENU           | XtNmenuBtn          |
| OL_MENUKEY         | MENU           | XtNmenuKey          |
| OL_MOVEDOWN        | MOVEDOWN       | XtNdownKey          |
| OL_MOVELEFT        | MOVELEFT       | XtNleftKey          |
| OL_MOVERIGHT       | MOVERIGHT      | XtNrightKey         |
| OL_MOVEUP          | MOVEUP         | XtNupKey            |
| OL_NEXTFIELD       | NEXTFIELD      | XtNnextFieldKey     |
| OL_PAGEDOWN        | ALT+PANE DOWN  | XtNpageDownKey      |
| OL_PAGELEFT        | ALT+PANE LEFT  | XtNpageLeftKey      |
| OL_PAGERIGHT       | ALT+PANE RIGHT | XtNpageRightKey     |
| OL_PAGEUP          | ALT+PANE UP    | XtNpageUpKey        |
| OL_PREVFIELD       | PREVFIELD      | XtNprevFieldKey     |
| OL_SCROLLBOTTOM    | DATA END       | XtNscrollBottomKey  |
| OL_SCROLLDOWN      | ALT+DOWN       | XtNscrollDownKey    |
| OL_SCROLLLEFT      | ALT+LEFT       | XtNscrollLeftKey    |
| OL_SCROLLLEFTEDGE  | ALT+{          | XtNscrollLeftEdge   |
| OL_SCROLLRIGHT     | ALT+RIGHT      | XtNscrollRightKey   |
| OL_SCROLLRIGHTEDGE | ALT+}          | XtNscrollRightEdge  |
| OL_SCROLLTOP       | DATA START     | XtNscrollTopKey     |
| OL_SCROLLUP        | ALT+UP         | XtNscrollUpKey      |
| OL_SELECT          | SELECT         | XtNselectBtn        |
| OL_SELECTKEY       | SELECT         | XtNselectKey        |
| OL_TOGGLEPUSHPIN   | TOGGLEPUSHPIN  | XtNtogglePushpinKey |
| OL_VSBMENU         | CTRL R         | XtNvertSBMenuKey    |

Activation types not described in the following table are described in “Common Activation Types” on page 68.”

***OL\_MENU***

This activation type pops up the Scrollbar menu as described in the *OPEN LOOK GUI Functional Specification* section “Scrollbar Menu” in Chapter 10. Activation of the Scrollbar menu items calls the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_MENUKEY/  
OL\_VSBMENU/  
OL\_HSBMENU***

These activation types will pop up the Scrollbar menu. When the Scrollbar menu is posted via keyboard traversal, the “Here to top” and “Top to here” buttons will not be sensitive. These buttons will depend on the position of the pointer when the menu is posted, and so they will not be applicable when the menu is posted from the keyboard. The `OL_VSBMENU` activation type applies only to Scrollbars with an `XtNOrientation` of `OL_VERTICAL`; and the `OL_HSBMENU` activation type applies only to Scrollbars with an `XtNOrientation` of `OL_HORIZONTAL`.

***OL\_PAGEDOWN***

For a scrollbar with `XtNOrientation` of `OL_VERTICAL`, this activation type will decrement the slider value by `XtNproportionLength` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_PAGELEFT***

For a scrollbar with `XtNOrientation` of `OL_HORIZONTAL`, this activation type will decrement the slider value by the value of `XtNproportionLength` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_PAGERIGHT***

For a scrollbar with `XtNOrientation` of `OL_HORIZONTAL`, this activation type will increment the slider value by the value of `XtNproportionLength` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.



***OL\_PAGEUP***

For a scrollbar with `XtNorientation` of `OL_VERTICAL`, this activation type will increment the slider value by `XtNproportionLength` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLBOTTOM***

For a scrollbar with `XtNorientation` of `OL_VERTICAL`, this activation type will move the slider to the value of `XtNsliderMin` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLDOWN/  
OL\_SCROLLLEFT***

These activation types will move the slider one negative unit of granularity and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLLEFTEDGE***

For a scrollbar with `XtNorientation` of `OL_HORIZONTAL`, this activation type will move the slider to the value of `XtNsliderMin` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLRIGHTEDGE***

For a scrollbar with `XtNorientation` of `OL_HORIZONTAL`, this activation type will move the slider to the value of `XtNsliderMax` minus the value of `XtNproportionLength` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLTOP***

For a scrollbar with `XtNorientation` of `OL_VERTICAL`, this activation type will move the slider to the value of `XtNsliderMax` minus the value of `XtNproportionLength` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLUP/  
OL\_SCROLLRIGHT***

These activation types will move the slider one positive unit of granularity and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SELECT***

This activation type depends on the position of the pointer within the Scrollbar widget. When the pointer is positioned on the elevator, the `XtNsliderMoved` callback will be called according to the value of the `XtNdragCBType`. When the pointer is positioned on the right anchor, the behavior will be the same as the `OL_SCROLLRIGHTEDGE` activation type. When the pointer is positioned on the top anchor, the behavior will be the same as the `OL_SCROLLTOP` activation type. When the pointer is positioned on the left anchor, the behavior will be the same as the `OL_SCROLLLEFTEDGE` activation type. When the pointer is positioned on the bottom anchor, the behavior will be the same as the `OL_SCROLLBOTTOM` activation type. When the pointer is positioned on the up arrow, the behavior will be the same as the `OL_SCROLLUP` activation type. When the pointer is positioned on the right arrow, the behavior will be the same as the `OL_SCROLLRIGHT` activation type. When the pointer is positioned on the left arrow, the behavior will be the same as the `OL_SCROLLLEFT` activation type. When the pointer is positioned on the down arrow, the behavior will be the same as the `OL_SCROLLDOWN` activation type. When the pointer is positioned on the cable above or to the right of the elevator, the behavior will be the same as the `OL_PAGEUP` and `OL_PAGERIGHT` activation types, respectively. When the pointer is positioned on the cable below or to the left of the elevator, the behavior will be the same as the `OL_PAGEDOWN` and `OL_PAGELEFT` activation types, respectively.

***See Also***

“MenuShell Widget” on page 414.

## *ScrolledWindow Widget*

### *Class*

*Class Name:* ScrolledWindow  
*Class Pointer:* scrolledWindowWidgetClass

### *Ancestry*

Core-Composite-Constraint-Manager-ScrolledWindow

### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/ScrolledWi.h>
```

### *Description*

The ScrolledWindow has no native text or graphic capabilities, but provides the basis for implementing the OPEN LOOK scrollable text or graphics pane.

### *Components*

The ScrolledWindow widget has the following components:

- Vertical scrollbar (typically)
- Horizontal scrollbar (typically)
- Content (not necessarily all visible)
- View of the content (visible part of content)
- View border

*ScrolledWindow Widget*

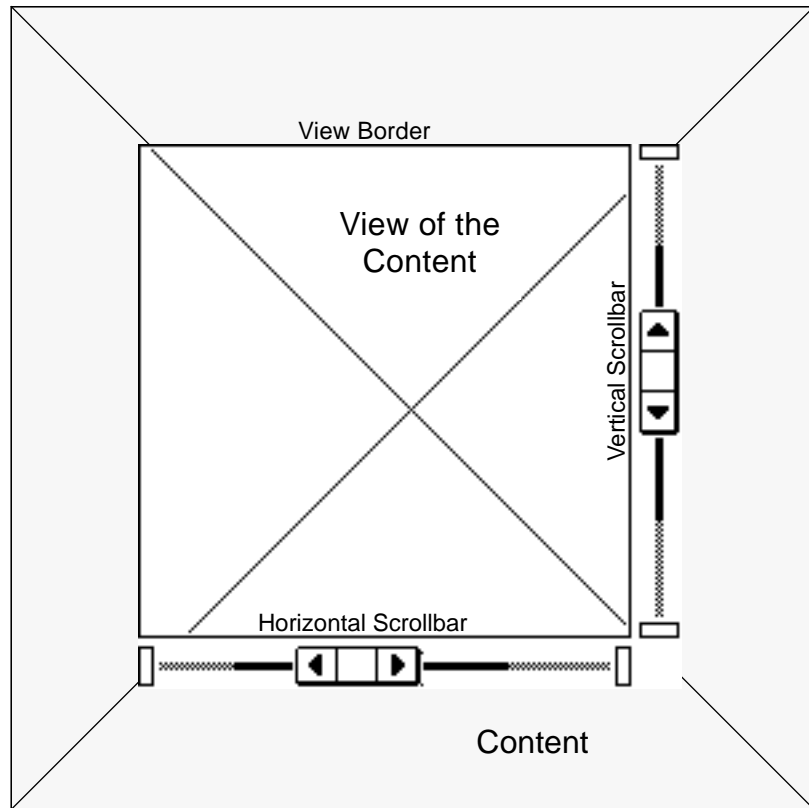


Figure 9-8 ScrolledWindow Components

**Subwidgets**

The ScrolledWindow contains five subwidgets: a BulletinBoard to hold the child contents; two Scrollbars, horizontal and vertical, along with their associated menus, created automatically, and accessible through the following resources:

- XtNhScrollbar
- XtNvScrollbar
- XtNhMenupane
- XtNvMenupane

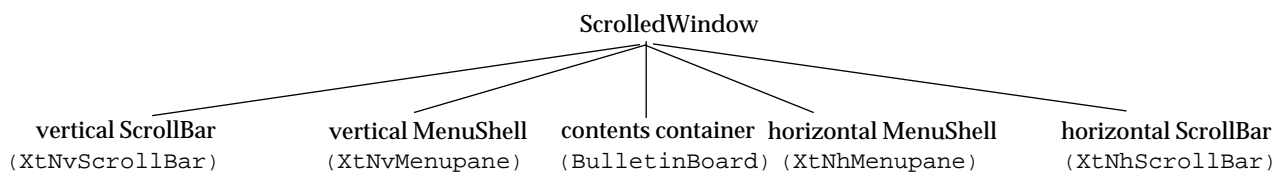


Figure 9-9 ScrolledWindow Subwidgets

### **View Border**

The view border is a narrow outline around the view of the content: in 2D, it is a 1-pixel outline; in 3D, it is a 2-point beveled outline.

### **Content and View of Content**

The ScrolledWindow widget implements a scrollable visible window (the *view of the content*) onto another, typically larger, data display (the *content*). The view can be moved through the content using the scroll bars.

To use the scrolled window, the application creates a widget capable of displaying the entire content as a child of the ScrolledWindow widget. The ScrolledWindow widget positions the child widget “within” the view of the content, and creates scroll bars for the horizontal and vertical dimensions, as needed. When the user performs some action on the scroll bars, the child widget will be repositioned accordingly within the view of the content. A larger child widget positioned within a smaller view by the ScrolledWindow widget, is forced to display only the viewed part of itself. This is all handled through normal widget geometry management.

The word “within” is used strictly in the widget sense: the larger child widget is positioned within the smaller view of the content part of the ScrolledWindow widget, which necessarily forces the child widget to display only the visible part of itself. The protocol for this is through normal widget geometry interactions.

### **Upper Left Corner Fixed on Resize**

If the ScrolledWindow widget is resized, the upper left corner of the view stays fixed over the same spot in the content, unless this would cause the view to extend past the right or bottom edge of the content. If necessary, the upper left corner will shift left or up only enough to keep the view from extending past the right or bottom edge.

***View Larger than Content***

Generally, the view of the content never becomes larger than needed to show the content. This default behavior can be overridden using the `XtNrecomputeHeight` and `XtNrecomputeWidth` resources (see page 543).

***Scrollbars***

The scrollbars are configured to scroll integer values, in pixels, through the width and length of the content. This allows the finest degree of control of the positioning of the view of the content. However, the application can set the step rate through these values to avoid a large number of view updates as the user scrolls through the content.

Unless forced to appear (see “`XtNforceHorizontalSB`” on page 538), a scrollbar is removed from the side where it is no longer needed. Remaining scrollbars stay a fixed distance from the view.

***Application Controlled Scrolling***

The `ScrolledWindow` widget also supports application-controlled scrolling, including the scrolling of large amounts of data such as text. In this mode of operation, the application monitors user interaction with the Scrollbars and displays the appropriate data in the view.

The application specifies this mode of operation by setting the `XtNvAutoScroll/XtNhAutoScroll` resources to `FALSE`. Normally, these settings are combined with the setting of the `XtNvSliderMoved` and `XtNhSliderMoved` callbacks. Also, the application should specify an `XtNcomputeGeometries` callback, which is used to lay out the `ScrolledWindow`. To programmatically manipulate the scrollbars, the application must set the `XtNuseSetValCallback` resource of the Scrollbar widget to `TRUE`. The `TextEdit` widget (see page 623) recognizes when it is a child of a `ScrolledWindow` widget and operates in this mode.

### Coloration

The diagram illustrates the resources that affect ScrolledWindow coloration.

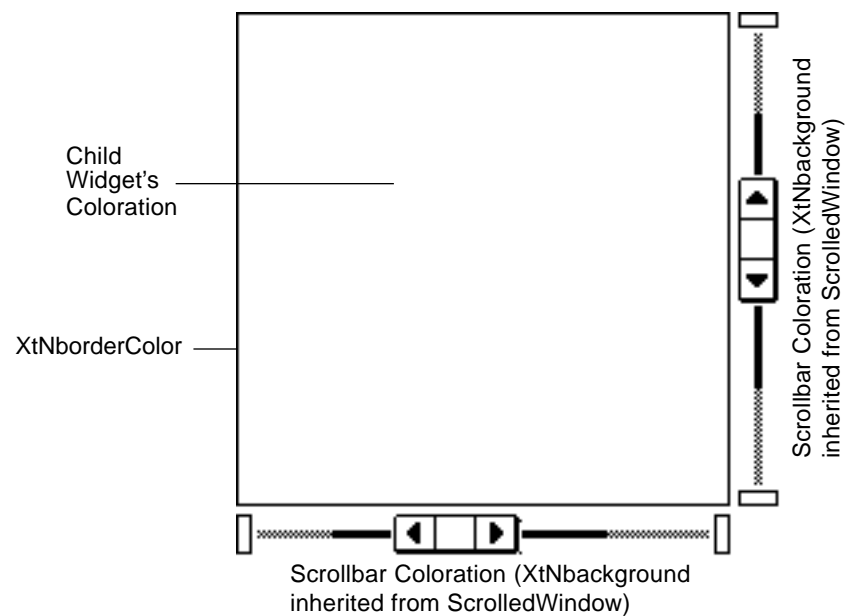


Figure 9-10 ScrolledWindow Coloration

### Keyboard Traversal

The ScrolledWindow controls the keyboard traversal between the content, the horizontal ScrollBar, and the vertical ScrollBar. The scrollbars that are created by the ScrolledWindow have the `XtNtraversalOn` resource set to `FALSE`.

A content widget added to the ScrolledWindow with traversal enabled will be added to the traversable widgets in the window with the scrollbars so that the user can move between them with the `NEXTFIELD` (or `MOVEUP` or `MOVELEFT`) and `PREVFIELD` (or `MOVEDOWN` or `MOVERIGHT`) keys.

*ScrolledWindow Widget*  
*Resources*

*Table 9-18 ScrolledWindow Core Resources*

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | (parent's)          | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | 0                   | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 9-19 ScrolledWindow Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 9-20 ScrolledWindow Manager Resources*

| <b>Name</b>        | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|--------------------|----------------|----------------|---------------|
| XtNconsumeEvent    | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor | Pixel          | Red            | SGID          |



*ScrolledWindow Widget**Table 9-20 ScrolledWindow Manager Resources (Continued)*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 9-21 ScrolledWindow Resources*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|----------------------|----------------|---------------------|---------------|
| XtNalignHorizontal   | OlDefine       | OL_BOTTOM           | SGI           |
| XtNalignVertical     | OlDefine       | OL_RIGHT            | SGI           |
| XtNcomputeGeometries | Function       | NULL                | SGI           |
| XtNcurrentPage       | int            | 1                   | SGI           |
| XtNforceHorizontalSB | Boolean        | FALSE               | SGI           |
| XtNforceVerticalSB   | Boolean        | FALSE               | SGI           |
| XtNforeground        | Pixel          | XtDefaultForeground | SGID          |
| XtNhAutoScroll       | Boolean        | TRUE                | SGI           |
| XtNhInitialDelay     | int            | 500 (msec)          | SGI           |
| XtNhMenuPane         | Widget         | NULL                | G             |
| XtNhRepeatRate       | int            | 100 (msec)          | SGI           |
| XtNhScrollbar        | Widget         | NULL                | G             |
| XtNhSliderMoved      | XtCallbackList | NULL                | SGIO          |
| XtNhStepSize         | int            | 1                   | SGI           |
| XtNinitialX          | int            | 0                   | GI            |
| XtNinitialY          | int            | 0                   | GI            |
| XtNrecomputeHeight   | Boolean        | TRUE                | SGI           |
| XtNrecomputeWidth    | Boolean        | TRUE                | SGI           |
| XtNshowPage          | OlDefine       | OL_NONE             | SGI           |
| XtNvAutoScroll       | Boolean        | TRUE                | SGI           |
| XtNviewHeight        | Dimension      | 0                   | SGI           |
| XtNviewWidth         | Dimension      | 0                   | SGI           |
| XtNvInitialDelay     | int            | 500 (msec)          | SGI           |
| XtNvMenuPane         | Widget         | NULL                | G             |
| XtNvRepeatRate       | int            | 100 (msec)          | SGI           |

*ScrolledWindow Widget*

Table 9-21 ScrolledWindow Resources (Continued)

| Name            | Type           | Default | Access |
|-----------------|----------------|---------|--------|
| XtNvScrollbar   | Widget         | NULL    | G      |
| XtNvSliderMoved | XtCallbackList | NULL    | SGIO   |
| XtNvStepSize    | int            | 1       | SGI    |

***XtNalignHorizontal***

| Class              | Type     | Default   | Access |
|--------------------|----------|-----------|--------|
| XtCAlignHorizontal | OlDefine | OL_BOTTOM | SGI    |

Synopsis: The placement of the horizontal scrollbar.

Values: OL\_BOTTOM/"bottom" - Place the horizontal scrollbar on the bottom.

OL\_TOP/"top" - Place the horizontal scrollbar on the top.

***XtNalignVertical***

| Class            | Type     | Default  | Access |
|------------------|----------|----------|--------|
| XtCAlignVertical | OlDefine | OL_RIGHT | SGI    |

Synopsis: The placement of the vertical scrollbar.

Values: OL\_RIGHT/"right" - Place the vertical scrollbar on the right.

OL\_LEFT/"left" - Place the vertical scrollbar on the left.

***XtNcomputeGeometries***

| Class                | Type     | Default | Access |
|----------------------|----------|---------|--------|
| XtCComputeGeometries | Function | NULL    | SGI    |

Synopsis: A function called whenever the ScrolledWindow needs to lay out its children.

The content widget sets this resource to a pointer to a function that is called whenever the ScrolledWindow needs to lay out its children:

```
void (*compute_geometries)(
 Widget content_widget,
 OlSWGeometries *geometries);
```

---

## ScrolledWindow Widget

The structure pointed to be *geometries* is defined as:

```
typedef struct _OlSWGeometries {
 Widget sw;
 Widget vsb;
 Widget hsb;
 Dimension bb_border_width;
 Dimension vsb_width;
 Dimension vsb_min_height;
 Dimension hsb_height;
 Dimension hsb_min_width;
 Dimension sw_view_width;
 Dimension sw_view_height;
 Dimension bbc_width;
 Dimension bbc_height;
 Dimension bbc_real_width;
 Dimension bbc_real_height;
 Boolean force_hsb;
 Boolean force_vsb;
} OlSWGeometries;
```

The ScrolledWindow widget populates the values in this structure before the call and examines them after the call to perform the layout operation. The fields are as follows:

|                        |                                                            |
|------------------------|------------------------------------------------------------|
| <i>sw</i>              | The widget ID of the ScrolledWindow widget.                |
| <i>vsb</i>             | The widget ID of its vertical Scrollbar widget.            |
| <i>hsb</i>             | The widget ID of its horizontal Scrollbar widget.          |
| <i>bb_border_width</i> | The width of the border around the view.                   |
| <i>vsb_width</i>       | The width of the vertical Scrollbar.                       |
| <i>vsb_min_height</i>  | The minimum height of the vertical Scrollbar.              |
| <i>hsb_height</i>      | The height of the horizontal Scrollbar.                    |
| <i>hsb_min_width</i>   | The minimum width of the horizontal Scrollbar.             |
| <i>sw_view_width</i>   | The width of the entire ScrolledWindow.                    |
| <i>sw_view_height</i>  | The height of the entire ScrolledWindow.                   |
| <i>bbc_width</i>       | The width of the view.                                     |
| <i>bbc_height</i>      | The height of the view.                                    |
| <i>bbc_real_width</i>  | The width of the contents.                                 |
| <i>bbc_real_height</i> | The height of the contents.                                |
| <i>force_hsb</i>       | When TRUE, a horizontal Scrollbar is forced to be present. |
| <i>force_vsb</i>       | When TRUE, a vertical Scrollbar is forced to be present.   |

*ScrolledWindow Widget*

The called function is responsible for populating the following fields of this structure:

```

bbc_width
bbc_height
bbc_real_width
bbc_real_height
force_hsb
force_vsb

```

For example, if the application wants to make the view 300 pixels wide by 200 pixels high into a contents that is 1200 pixels wide and 700 pixels high, it needs to set these fields as follows:

```

bbc_width = 300
bbc_height = 200
bbc_real_width = 1200
bbc_real_height = 700

```

Since the view is smaller than the contents in both width and height, both scrollbars will appear in this case. Therefore, the values supplied for *force\_vsb* and *force\_hsb* are not important. However, one could have situations where one of the scrollbars is always wanted independent of the relationship between the view and content sizes; in such cases, *force\_hsb* and *force\_vsb* are useful.

***XtNcurrentPage***

| Class          | Type | Default | Access |
|----------------|------|---------|--------|
| XtCCurrentPage | int  | 1       | SGI    |

Synopsis: The value to be used by the ScrolledWindow vertical scrollbar.

See “Scrollbar Widget” on page 508 for more details.

***XtNforceHorizontalSB***

| Class                | Type    | Default | Access |
|----------------------|---------|---------|--------|
| XtCForceHorizontalSB | Boolean | FALSE   | SGI    |

Synopsis: Force attachment of a horizontal scrollbar.

Values: TRUE/“true” - Disables size checking and forces the scrollbar to be attached to the window regardless of the size of the child widget.

---

*ScrolledWindow Widget*

FALSE/"false" - Performs size checking and does not force attachment.

When the child widget is created and positioned within the scrolled window, its width is examined. If the entire child widget will fit within the width of the scrolled window, and the horizontal scrollbar is not forced and the horizontal scrollbar will not be created, since there is no need to scroll in that direction.

If a scrollbar is forced but not needed because the content fits within the view, the scrollbar is made insensitive.

***XtNforceVerticalSB***

| Class              | Type    | Default | Access |
|--------------------|---------|---------|--------|
| XtCForceVerticalSB | Boolean | FALSE   | SGI    |

Synopsis: Force attachment of a vertical scrollbar.

Values: TRUE/"true" - Disables size checking and forces the scrollbar to be attached to the window, regardless of the size of the child widget.

FALSE/"false" - Performs size checking and does not force attachment.

When the child widget is created and positioned within the scrolled window, its height is examined. If the entire child widget will fit within the height of the scrolled window, and the vertical scrollbar is not forced, the vertical scrollbar will not be created, since there is no need to scroll in that direction.

If a scrollbar is forced but not needed because the content fits within the view, the scrollbar is made insensitive.

***XtNforeground***

| Class         | Type  | Default             | Access |
|---------------|-------|---------------------|--------|
| XtCForeground | Pixel | XtDefaultForeground | SGID   |

Synopsis: The foreground color used by the ScrolledWindow to draw non-textual content.

Values: Any Pixel value valid for the current display, or any name from the \$OPENWINHOME/lib/rgb.txt file. (Pixel values are used in C programs, rgb.txt values in X resource files. See "Resource Files" on page 7.)

*ScrolledWindow Widget*

***XtNhAutoScroll/  
XtNvAutoScroll***

| <b>Class</b>   | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|----------------|-------------|----------------|---------------|
| XtCHAutoScroll | Boolean     | TRUE           | SGI           |
| XtCVAutoScroll | Boolean     | TRUE           | SGI           |

**Synopsis:** The scrolling mode in the horizontal (vertical) direction.  
**Values:** TRUE/"true" - The ScrolledWindow widget is responsible for all interaction with the scrollbar and the positioning of the content window within the view.  
 FALSE/"false" - The application is responsible for all scrollbar interaction and scrolling of the data within the content window.

***XtNhInitialDelay/  
XtNvInitialDelay***

| <b>Class</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtCHInitialDelay | int         | 500 (msec)     | SGI           |
| XtCVInitialDelay | int         | 500 (msec)     | SGI           |

**Synopsis:** The time in milliseconds of the initial repeat delay to be used when the scrolling arrows of the horizontal (vertical) scrollbar component of the ScrolledWindow are pressed.

***XtNhMenuPane /  
XtNvMenuPane***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCHMenuPane | Widget      | NULL           | G             |
| XtCVMenuPane | Widget      | NULL           | G             |

**Synopsis:** A widget for a menu that can be popped up from the horizontal scrollbar.

These resources provide for the retrieval of a handle to the menu that may be popped up over a horizontal (vertical) scrollbar. This is useful for customizing the menu. For example, the application may add the concept of a page or chapter. The menu could then have items for scrolling forward and backward by those units.

See "Scrollbar Widget" on page 508 for more details.

***XtNhRepeatRate/  
XtNvRepeatRate***

| Class          | Type | Default    | Access |
|----------------|------|------------|--------|
| XtCHRepeatRate | int  | 100 (msec) | SGI    |
| XtCVRepeatRate | int  | 100 (msec) | SGI    |

Synopsis: The time in milliseconds of the repeat delay to be used when the scrolling arrows of the horizontal (vertical) scrollbar component of the ScrolledWindow are pressed.

***XtNhScrollbar/  
XtNvScrollbar***

| Class        | Type   | Default | Access |
|--------------|--------|---------|--------|
| XtCScrollbar | Widget | NULL    | G      |
| XtVScrollbar | Widget | NULL    | G      |

Synopsis: The widget ID of the horizontal (vertical) scrollbar.

An application uses these values to set scrollbar characteristics, such as coloration.

***XtNhSliderMoved/  
XtNvSliderMoved***

| Class        | Type           | Default | Access |
|--------------|----------------|---------|--------|
| XtCScrollbar | XtCallbackList | NULL    | SGIO   |
| XtVScrollbar | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback lists for tracking child position.

These resources mimic the `XtNsliderMoved` resource of the horizontal (vertical) scrollbar. The `call_data` parameter for this callback is a pointer to an `OlScrollbarVerify` structure, as in the Scrollbar widget. The application can validate a scroll attempt before the ScrolledWindow widget will reposition the view of the content, and can update the page number and adjust the scrollbar elevator position. See “Scrollbar Widget” on page 508 for more details.

*ScrolledWindow Widget*

***XtNhStepSize/  
XtNhStepSize***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCHStepSize | int         | 1              | SGI           |
| XtCVStepSize | int         | 1              | SGI           |

Synopsis: The minimum unit of horizontal (vertical) scrolling.  
 Values: The size in pixels of the minimum scrollable unit in the content:  
 $0 < XtNhStepSize$   
 $0 < XtNvStepSize$

For instance, to allow the user to scroll a single pixel in either direction, the value would be 1. Or, to allow the user to scroll a character at a time horizontally, the value would be the width of a character. Scrolling a character at a time requires a constant width font, of course.

The ScrolledWindow widget uses this value to calibrate the minimum scrolling step, *XtNgranularity*, of the scrollbars.

***XtNinitialX/  
XtNinitialY***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCInitialX  | int         | 0              | GI            |
| XtCInitialY  | int         | 0              | GI            |

Synopsis: The initial x- and y-position of the child widget.  
 Values:  $0 \geq XtNinitialX$   
 $0 \geq XtNinitialY$

The child widget is initially positioned at the upper left corner, (x,y) coordinates of (0,0). This positioning can be changed by specifying a new x,y location. The scrollbars are adjusted to give a visual indication of the offset specified in these resources. The content is positioned within the view of the content, so as the view of the content moves progressively further through the content, the coordinates of the position become more *negative*. Thus, the initial coordinate given in this resource should be zero or negative to ensure proper operation of the scrolled window.



***XtNrecomputeHeight/  
XtNrecomputeWidth***

| Class              | Type    | Default | Access |
|--------------------|---------|---------|--------|
| XtCRecomputeHeight | Boolean | TRUE    | SGI    |
| XtCRecomputeWidth  | Boolean | TRUE    | SGI    |

Synopsis: The resizing method used by the ScrolledWindow widget.

Values: TRUE/"true" - The ScrolledWindow shrinks the view of the content in the corresponding direction to absorb the change in the ScrolledWindow widget's size.

FALSE/"false" - The ScrolledWindow does not shrink the view in that direction.

These resources, together with XtNviewWidth and XtNviewHeight, are typically used to set a preferred dimension in a direction that should not be scrolled.

***XtNshowPage***

| Class       | Type     | Default | Access |
|-------------|----------|---------|--------|
| XtCShowPage | OlDefine | OL_NONE | SGI    |

Synopsis: The actions of the scrollbar page indicator during dragging. This value is directed to the vertical scrollbar in the ScrolledWindow widget.

Values: OL\_LEFT/"left" - Display the page indicator to the left of the drag box.

OL\_RIGHT/"right" - Display the page indicator to the right of the drag box.

OL\_NONE/"none" - Display no page indicator.

See "Scrollbar Widget" on page 508 for more details.

***Activation Types***

The following table lists the activation types used by the ScrolledWindow.

Table 9-22 ScrolledWindow Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |

*ScrolledWindow Widget*

*Table 9-22 ScrolledWindow Activation Types (Continued)*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_HSBMENU             | HSBMENU          | XtNhorizSBMenuKey    |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PAGEDOWN            | PAGEDOWN         | XtNpageDownKey       |
| OL_PAGELEFT            | PAGELEFT         | XtNpageLeftKey       |
| OL_PAGERIGHT           | PAGERIGHT        | XtNpageRightKey      |
| OL_PAGEUP              | PAGEUP           | XtNpageUpKey         |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_SCROLLBOTTOM        | SCROLLBOTTOM     | XtNscrollBottomKey   |
| OL_SCROLLDOWN          | SCROLLDOWN       | XtNscrollDownKey     |
| OL_SCROLLLEFT          | SCROLLLEFT       | XtNscrollLeftKey     |
| OL_SCROLLLEFTEDGE      | SCROLLLEFTEDGE   | XtNscrollLeftEdge    |
| OL_SCROLLRIGHT         | SCROLLRIGHT      | XtNscrollRightKey    |
| OL_SCROLLRIGHTEDGE     | SCROLLRIGHTEDGE  | XtNscrollRightEdge   |
| OL_SCROLLTOP           | SCROLLTOP        | XtNscrolltopKey      |
| OL_SCROLLUP            | SCROLLUP         | XtNscrollUpKey       |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |
| OL_VSBMENU             | VSBMENU          | XtNvertSBMenuKey     |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

***OL\_HSBMENU***

This activation type will pop up the menu associated with the `XtNhScrollbar` widget. When this menu is posted via keyboard traversal, the “Here to left” and “Left to here” buttons will not be sensitive. Activating the “Previous” menu item on this menu calls the `XtNhSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_PAGEDOWN***

This activation type will decrement the `XtNvScrollbar`'s slider value by an amount necessary to scroll the view down by the height of the view, and call the `XtNvSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_PAGELEFT***

This activation type will decrement the `XtNhScrollbar`'s slider value by an amount necessary to scroll the view left by the width of the view, and call the `XtNhSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_PAGERIGHT***

This activation type will increment the `XtNhScrollbar`'s slider value by an amount necessary to scroll the view right by the width of the view, and call the `XtNhSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_PAGEUP***

This activation type will increment the `XtNvScrollbar`'s slider value by an amount necessary to scroll the view up by the height of the view, and call the `XtNvSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLBOTTOM***

This activation type will move the vertical scrollbar's slider such that the view is moved to the bottom of the content, and call the `XtNvSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLDOWN***

This activation type will move the vertical scrollbar's slider one negative unit of granularity and call the `XtNvSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLLEFT***

This activation type will move the horizontal scrollbar's slider one negative unit of granularity and call the `XtNhSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLLEFTEDGE***

This activation type will move the horizontal scrollbar's slider such that the view is moved to the left edge of the content, and call the `XtNhSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLRIGHT***

This activation type will move the horizontal scrollbar's slider one positive unit of granularity and call the `XtNhSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLRIGHTEDGE***

This activation type will move the horizontal scrollbar's slider such that the view is moved to the right edge of the content, and call the `XtNhSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLTOP***

This activation type will move the vertical scrollbar's slider such that the view is moved to the top of the content, and call the `XtNvSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLUP***

This activation type will move the vertical scrollbar's slider one positive unit of granularity and call the `XtNvSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***OL\_VSBMENU***

This activation type will pop up the menu associated with the `XtNvScrollbar` widget. When this menu is posted via keyboard traversal, the "Here to top" and "Top to here" buttons will not be sensitive. Activating the "Previous" menu item on this menu calls the `XtNvSliderMoved` callback list with the appropriate `OlScrollbarVerify` structure.

***See Also***

"Scrollbar Widget" on page 508.

## ScrollingList Widget

### Class

**Class Name:** ScrollingList  
**Class Pointer:** scrollingListWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-Form-ScrollingList

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/ScrollingL.h>
```

### Description

The ScrollingList widget provides a list of items that the user can scroll through and choose. The application can make the choice of items exclusive, or allow the user to make multiple choices. It can also enable the user to edit items within the list.

OLIT releases after 3.1 have some new features and a new Application Programming Interface (API). The new features include multiple ScrollingList modes, item sensitivity, and double-click support. This *new API* enables the application to use these new features and also obsoletes the old API with a richer set of convenience functions. Refer to “ScrollingList Modes” on page 552 for more details.

### Components

Each ScrollingList widget has the following parts: Items, a Scrollbar, and a View. If the application allows the list to be edited in place (in the view), the ScrollingList widget also uses an editable text field.

*ScrollingList Widget*

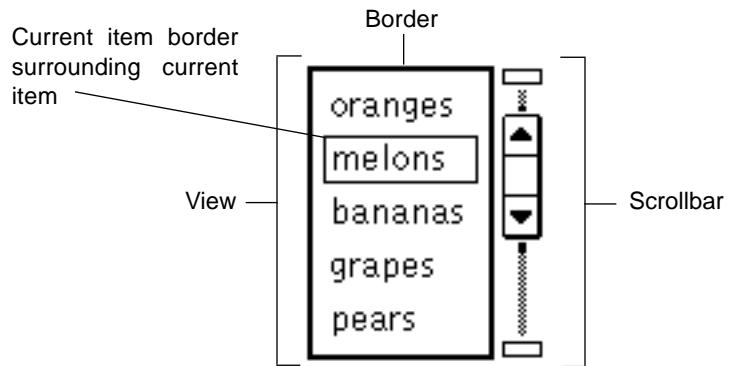


Figure 9-11 ScrollingList Components

**Editable ScrollingList**

The application can choose whether to allow the user to edit the items in a scrolling list. The editable text field is the interface for entering the new item, and is described later. Other aspects of the user interface for editing are controlled by the application. For example, the application can attach a menu to the scrolling list to allow the user to select where a new item is to be inserted, and can employ popup windows to gather additional information about a new item.

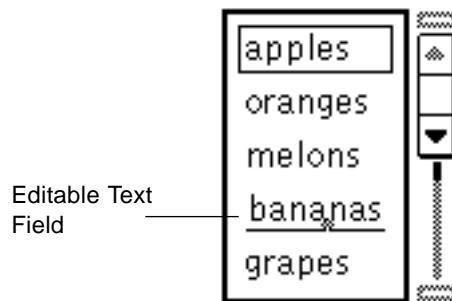


Figure 9-12 Editable ScrollingList

### Subwidgets

The ScrollingList contains three subwidgets: a Scrollbar, a ListPane, and a TextField.

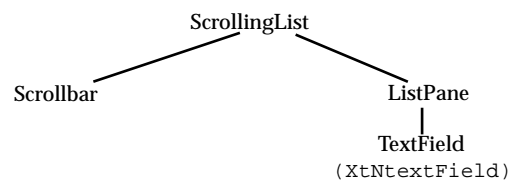


Figure 9-13 ScrollingList Subwidgets

### Editable Text Field

The application can request that the ScrollingList widget provide an editable text field that will allow the user to change an existing item in the view. The TextField subwidget implements this editable text field. The ScrollingList widget manages the TextField widget as follows:

- **Opening/closing** – The application asks the ScrollingList widget to “open” and “close” the editable text field. Opening the editable text field widget maps it and positions it so that, as the user types in the name of a new or changed item, the name lines up with the existing item names. Closing the editable text field widget unmaps it. (As described below, there may be times when the widget is unmapped, yet still open.)
- **Editing** – If an existing item is being edited, the application requests the editable text field to overlay the item.
- **Insertion** – If a new item is being inserted, the application requests items to be scrolled down in the view to accommodate the editable text field.
- **Mapping** – The ScrollingList widget maps and unmaps the editable text field widget; the application does not.
- **Scrolling** – If the user scrolls the list while the editable text field is still open, the ScrollingList widget scrolls it with the rest of the items. If it has to be scrolled out of the view, it is scrolled out entirely, causing it to be unmapped but not closed. The application should not try to remap the child since it will be remapped when the list is scrolled back again.
- **Setting/selecting** – If the user attempts to make a selection or set a current item, the editable text field is automatically closed.

---

## ScrollingList Widget

The application is responsible for handling the verification callbacks of the editable text field and for telling the ScrollingList widget to add a new item or change an existing item as a result of the user input.

### **Selectable ScrollingList**

The application can choose whether to allow the user to select items from a scrolling list. If items can be selected, they can be copied elsewhere as text, and may be deletable (“cut”).

### **Deleting Selected Items**

The user can delete selected items. The ScrollingList widget provides some deletion capabilities through the selection mechanisms (see the discussion under “Text Selections on Items” on page 551), and the application can provide other capabilities, such as with a popup menu choice. The application verifies that each selected item can be deleted; it is responsible for providing feedback to the user for any items it will not delete. The ScrollingList widget updates the view to remove any deleted items.

### **Item Order**

The list is assumed to have an order defined by the application. As it adds items, the application tells the ScrollingList widget where to insert them: either before an item already in the list or at the end of the list.

The application may change the content of a list at any time, including while it is displayed. The widget updates the view, if necessary, to reflect the changed list. To avoid unnecessary updates to the view when several changes need to be made, the application can instruct the ScrollingList widget to avoid updates until the changes are finished.

### **Making an Item Current**

The user can make an item *current* by:

- Pressing SELECT over it, or
- Moving the input focus into the widget and typing the first letter of the item’s name.

Since OLIT 3.2, either of these actions will add the item to the list of *current* items or will select a new current item, depending on the mode (exclusive,



---

## ScrollingList Widget

exclusive-none-set, or nonexclusive). Double-clicking will invoke an application-specified callback.

In the default API, either of these actions caused a callback to the application, which could decide if the item should be made a current item, remain a current item, or be changed to a regular item, depending on the current state of the item and the needs of the application. Thus, the application can make the scrolling list behave as a set of exclusive or nonexclusive items.

Pressing SELECT also starts a selection, as described below.

### **Text Selections on Items**

The ScrollingList widget allows selection operations on the items if the `XtNselectable` resource is TRUE; see page 567. Items that are moved or copied from the view are treated as a newline-separated list of text items, in the order they appear in the scrolling list, with no leading or trailing blanks on any item.

- Selecting a single item. Clicking SELECT on an item selects it and deselects any other active selection on the screen.
- Selecting other items. Clicking ADJUST on an item toggles its state, making an unselected item selected and a selected item unselected.
- Wipe-through selection, with SELECT. Pressing and dragging SELECT over items selects them and deselects any other active selection on the screen. The selection starts with the item where SELECT is pressed and extends to the item where SELECT is released. If the pointer moves above or below the view, the view scrolls additional items into the view, selecting them as well. The rate at which items scroll into the view is the same as when pressing SELECT on the up or down arrows of the Scrollbar. The pointer can move out of the view to the left or right without interrupting the selection.
- Wipe-through selection, with ADJUST. Pressing and dragging ADJUST marks the bounds of a selection the same way as pressing and dragging SELECT, except that the items covered are “toggled.” (Previously selected items are deselected and previously unselected items are selected.)
- Copying items. Pressing COPY copies any selected items to the clipboard and deselects them.
- Cutting items. Pressing CUT moves any selected items to the clipboard and deletes them from the list. This operation is allowed only if the scrolling list is editable.

*ScrollingList* Widget

***ScrollingList* Modes**

This feature is newly introduced in OLIT 3.2. The *ScrollingList* Widget can be created in one of the following modes:

|                       |                                                                                                                                                                                        |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OL_EXCLUSIVE          | This “Exclusive” mode lets the user choose one item from a list of items, so that at any given time exactly one item is current or chosen.                                             |
| OL_EXCLUSIVE_NONESSET | This is a variation of the “Exclusive” <i>ScrollingList</i> . Here the user can choose one or none of the items on the list.                                                           |
| OL_NONEXCLUSIVE       | This lets the user choose none, one, or multiple items from the list.                                                                                                                  |
| OL_NONE               | None of the above three modes. This is the default mode for the <i>ScrollingList</i> and is termed the <i>default API</i> in this section. It is provided for backwards compatibility. |

---

**Note** – The old API uses the `OLListToken` datatype; the new API uses the `OLSlstItemPtr` datatype instead. Applications cannot use any functions in the old API to manipulate the new features. Applications should also take care not to mix functions from the two APIs.

---

In this section the term *item* refers to a *ScrollingList* item of datatype `OLSlstItemPtr`.

***Coloration***

For both 2D and 3D, `XtNfontColor` is used to draw the *ScrollingList* item labels. If `XtNselectable` is set to `TRUE`, when a list item is selected (for `COPY` and `PASTE`), the *ScrollingList* reverses `XtNbackground` and `XtNfontColor` for the selected item. See individual “Coloration” sections for *Scrollbar* (page 514) and *TextField* (page 665) for information on the *ScrollingList*’s scrollbar and textfield component coloration.

For 3D, *ScrollingList* item coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.” `XtNbackground` is used for `BG1`, and the `BG2` (pressed-in), `BG3` (shadow), and Highlight colors are derived by the toolkit from `BG1`.

---

## ScrollingList Widget

For 2D, `XtNbackground` and `XtNforeground` are used to render the `ScrollingList` items as described by the *OPEN LOOK GUI Functional Specification*, Chapter 8, “Scrolling Lists.”

If the toolkit resource `XtNmouseless` is set to `TRUE` and the toolkit resource `XtNinputFocusFeedback` is set to `OL_INPUT_FOCUS_COLOR`, then the background of the list item with focus will be drawn with the value of `XtNinputFocusColor` when the `ScrollingList` receives input focus. However, if `XtNinputFocusColor` is the same as `XtNbackground`, then the `ScrollingList` inverts `XtNfontColor` and `XtNbackground` within the list item with focus. Once the input focus leaves the widget, the original coloration is restored.

### Keyboard Traversal

The default value of the `XtNtraversalOn` resource is `TRUE`.

The `ScrollingList` widget responds to the following keyboard navigation keys:

- `NEXTFIELD` moves to the next traversable widget in the window
- `PREVFIELD` moves to the previous traversable widget in the window
- `NEXTWINDOW` moves to the next window in the application
- `PREVWINDOW` moves to the previous window in the application
- `NEXTAPP` moves to the first window in the next application
- `PREVAPP` moves to the first window in the previous application
- `MOVEUP` moves the input focus up one line
- `MOVEDOWN` moves the input focus down one line
- `PANESTART` moves the input focus to the first item in the pane
- `PANEEND` moves the input focus to the last item in the pane
- `SCROLLUP` scrolls up one item in the list
- `SCROLLDOWN` scrolls down one item in the list
- `SCROLLTOP` scrolls to the first item in the list
- `SCROLLBOTTOM` scrolls to the last item in the list
- `PAGEUP` scrolls up one page so that the first item visible is the last item visible in the pane
- `PAGEDOWN` scrolls down one page so that the last item visible is the first item visible in the pane

When an Editable Text Field is in the `ScrollingList`, the keyboard traversal keys defined for `TextField` widgets apply.

*ScrollingList Widget*

The SELECTKEY selects the Current Item and unselects any other active selection on the screen. The ADJUSTKEY toggles the Current Item's state, making an unselected Item selected and a selected Item unselected.

The scrolling keys of interest are defined within the ScrollingList and traversal to the Scrollbar is not necessary to manipulate the ScrollingList.

***Keyboard Mnemonic Display***

The ScrollingList widget displays the mnemonic accelerator for each item as part of its label. If the mnemonic character is in the label, then that character is displayed or highlighted according to the value of the toolkit resource XtNshowMnemonics (see page 16). If the mnemonic character is not in the label, it is displayed to the right of the label in parentheses and highlighted according to the value of XtNshowMnemonics.

If truncation is necessary, the mnemonic displayed in parentheses is truncated as a unit.

***Known Deficiencies***

ScrollingList does not support wide character string formats. The application should use the multibyte interface as a workaround.

***Resources***

*Table 9-23 ScrollingList Core Resources*

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | 0                   | SGI           |

*ScrollingList Widget**Table 9-23 ScrollingList Core Resources (Continued)*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNmappedWhenManaged | Boolean        | TRUE           | SGI           |
| XtNscreen            | Screen *       | (parent's)     | G             |
| XtNsensitive         | Boolean        | TRUE           | GIO           |
| XtNtranslations      | XtTranslations | NULL           | SGI           |
| XtNwidth             | Dimension      | 0              | SGI           |
| XtNx                 | Position       | 0              | SGI           |
| XtNy                 | Position       | 0              | SGI           |

*Table 9-24 ScrollingList Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 9-25 ScrollingList Manager Resources*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor   | Pixel          | Red            | SGID          |
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 9-26 ScrollingList Resources*

| <b>Name</b>                    | <b>Type</b>          | <b>Default</b> | <b>Access</b> |
|--------------------------------|----------------------|----------------|---------------|
| XtNalign                       | Boolean              | TRUE           | SGI           |
| XtNapplAddItem <sup>1</sup>    | OIApplAddItemProc    | (special)      | G             |
| XtNapplDeleteItem <sup>1</sup> | OIApplDeleteItemProc | (special)      | G             |
| XtNapplEditClose <sup>1</sup>  | OIApplEditCloseProc  | (special)      | G             |

ScrollingList Widget

Table 9-26 ScrollingList Resources (Continued)

| Name                            | Type                 | Default      | Access |
|---------------------------------|----------------------|--------------|--------|
| XtNapplEditOpen <sup>1</sup>    | OIApplEditOpenProc   | (special)    | G      |
| XtNapplTouchItem <sup>1</sup>   | OIApplItemProc       | (special)    | G      |
| XtNapplUpdateView <sup>1</sup>  | OIApplUpdateViewProc | (special)    | G      |
| XtNapplViewItem <sup>1</sup>    | OIApplItemProc       | (special)    | G      |
| XtNcurrentItems                 | OISlistItemPtr *     | NULL         | G      |
| XtNfirstViewableItem            | OISlistItemPtr       | (calculated) | SG     |
| XtNitemCurrentCallback          | XtCallbackList       | NULL         | SGIO   |
| XtNitemHeight                   | Dimension            | 0            | SGI    |
| XtNitemNotCurrentCallback       | XtCallbackList       | NULL         | SGIO   |
| XtNlastViewableItem             | OISlistItemPtr       | (calculated) | SG     |
| XtNlistPane                     | Widget               | (special)    | G      |
| XtNmultiClickCallback           | XtCallbackList       | NULL         | SGIO   |
| XtNnumCurrentItems              | int                  | 0            | G      |
| XtNnumItems                     | int                  | 0            | G      |
| XtNposition                     | OIDefine             | OL_LEFT      | SGI    |
| XtNprefMaxWidth                 | Dimension            | 0            | SGI    |
| XtNprefMinWidth                 | Dimension            | 0            | SGI    |
| XtNrecomputeWidth               | Boolean              | TRUE         | SGI    |
| XtNscrollingListItems           | OISlistItemPtr *     | NULL         | G      |
| XtNscrollingListMode            | OIDefine             | OL_NONE      | GI     |
| XtNselectable                   | Boolean              | FALSE        | SGI    |
| XtNsensitive                    | Boolean              | TRUE         | GIO    |
| XtNspace                        | Dimension            | 4            | SGI    |
| XtNtextField                    | Widget               | (special)    | G      |
| XtNuserDeleteItems              | XtCallbackList       | NULL         | SGIO   |
| XtNuserMakeCurrent <sup>1</sup> | XtCallbackList       | NULL         | SGIO   |
| XtNviewableItems                | OISlistItemPtr *     | NULL         | G      |
| XtNviewHeight                   | Dimension            | (calculated) | SGI    |

1. This resource cannot be used with the new API and related functions.

**XtNalign**

| Class    | Type    | Default | Access |
|----------|---------|---------|--------|
| XtCAlign | Boolean | TRUE    | SGI    |

Synopsis: The alignment of combined string/glyph items.

Values: TRUE/"true" - The second components of all items having both string and glyph are aligned, at an offset of XtNspace from the widest first component.

FALSE/"false" - The items are not aligned.

**XtNapplAddItem**

| Class          | Type              | Default   | Access |
|----------------|-------------------|-----------|--------|
| XtCApplAddItem | OlApplAddItemProc | (special) | G      |

Synopsis: The system-provided routine the application should use when it adds a new item to the list. Applications should use XtGetValues() to retrieve this routine.

---

**Note** - This resource cannot be used with the new API and related functions.

---

The prototype for this routine is as follows:

```
typedef OlListToken (*OlApplAddItemProc)(Widget widget,
 OlListToken parent,
 OlListToken reference,
 OlListItem item);
```

*widget* Identifies the ScrollingList widget instance.

*parent* Should be set to 0, for compatibility with future changes.

*reference* Identifies an item before which to insert the new item. This value can be zero to append the new item to the list.

*item* Describes the new item.

The content of the OlListItem structure is copied by the ScrollingList widget into space that it maintains; however, the data pointed to by the label and glyph members are not copied. The application can access the copied data directly, using the OlListItemPointer() macro to get a pointer to the OlListItem structure for the item.

If it changes the data, the application should use the XtNapplTouchItem routine to let the ScrollingList widget know the data has changed.

*ScrollingList* Widget

If mapped and if allowed by the application (see “XtNapplUpdateView” on page 561), the *ScrollingList* widget updates the view if the new item will be in the view. The view is changed as little as possible: if the new item is in the upper half of the view, the items above it are scrolled up and the top item is scrolled off; if the new item is in the lower half of the view, the items below it are scrolled down and the bottom item is scrolled off.

This routine is also used to build the item list from scratch.

***XtNapplDeleteItem***

| Class             | Type                 | Default   | Access |
|-------------------|----------------------|-----------|--------|
| XtCApplDeleteItem | OlApplDeleteItemProc | (special) | G      |

Synopsis: The system-provided routine the application should call when it deletes an item from the list. Applications should use `XtGetValues()` to retrieve this routine.

---

**Note** – This resource cannot be used with the new API and related functions.

---

The prototype for this routine is as follows:

```
typedef void (*OlApplDeleteItemProc)(Widget widget,
 OlListToken token);
```

*widget* Identifies the *ScrollingList* widget instance.

*token* Identifies the deleted item.

If mapped and if allowed by the application (see “XtNapplUpdateView” on page 561), the *ScrollingList* widget updates the view if the deleted item was visible.

The view is changed as little as possible:

- If the deleted item was in the upper half of the view, items above it are scrolled down and an item is scrolled in from the top;
- If the deleted item was in the lower half of the view, items below it are scrolled up and an item is scrolled in from the bottom.
- If the view is already at the top or bottom, the additional item is scrolled in from the other end, if possible.



***XtNapplEditClose***

| Class            | Type            | Default   | Access |
|------------------|-----------------|-----------|--------|
| XtCApplEditClose | OlApplEditClose | (special) | G      |

**Synopsis:** The system-provided routine the application should call when the user has finished editing an item in the view. Applications should use `XtGetValues()` to retrieve this routine.

---

**Note** – This resource cannot be used with the new API and related functions.

---

The prototype for this routine is as follows:

```
typedef void (*OlApplEditCloseProc)(Widget widget);
```

*widget* Identifies the ScrollingList widget instance.

When this routine is called, the ScrollingList widget unmaps the editable text field widget, scrolling up the items below it if they had been scrolled down to allow an insert.

The application is responsible for calling the `XtNapplAddItem` routine to add the new item, or calling the `XtNapplTouchItem` routine to mark the item as changed.

To avoid unnecessary updates to the view, the application should add the new item (`XtNapplAddItem`) or mark the changed item (`XtNapplTouchItem`) before closing the editable text field. A later call to the `XtNapplEditClose` routine without an intervening call to the `XtNapplEditOpen` routine is ignored.

If mapped, the ScrollingList widget updates the view, even if the application had halted updates (see “`XtNapplUpdateView`” on page 561). If the application had halted updates, they will continue to be halted afterwards.

***XtNapplEditOpen***

| Class           | Type           | Default   | Access |
|-----------------|----------------|-----------|--------|
| XtCApplEditOpen | OlApplEditOpen | (special) | G      |

**Synopsis:** The system-provided routine the application should use when it wants to allow the user to insert a new item or change an existing item in the view. Applications should use `XtGetValues()` to retrieve this routine.

*ScrollingList* Widget

---

**Note** – This resource cannot be used with the new API and related functions.

---

The prototype for this routine is as follows:

```
typedef void (*OlApplEditOpenProc) (Widget widget,
 Boolean insert,
 OlListToken reference);
```

- widget* Identifies the *ScrollingList* widget instance.
- insert* Tells whether items should be scrolled down to make room for inserting a new item. A value of FALSE implies that an item is being edited in place and no items are to be scrolled.
- reference* Identifies an item before which a new item is to be inserted (*insert* is TRUE) or identifies the item that is being changed (*insert* is FALSE). If *insert* is TRUE, this value can be zero to append a new item at the end of the list. If *insert* is FALSE, this value must refer to an existing item. The referenced item does not have to be in the view (see below).

If a new item is being inserted, the *ScrollingList* widget makes room for the editable text field by scrolling down the referenced item and any items below it. If the referenced item is not in the view, it is automatically made visible just as if the application had called the *XtNapplViewItem* routine first. The *XtNapplEditOpen* routine can be called again before an intervening call to the *XtNapplEditClose* routine. The effect is as if the *XtNapplEditClose* routine was called, but without multiple updates to the view. For example, this allows the application to let the user insert several new items in succession: the editable text field moves down as each item is inserted, but is never removed from the view.

---

**Note** – This item-insertion feature is currently not implemented.

---

If mapped, the *ScrollingList* widget updates the view, even if the application had halted updates (see “*XtNapplUpdateView*” on page 561). If the application had halted updates, they will continue to be halted afterwards.

***XtNapplTouchItem***

| Class            | Type                | Default   | Access |
|------------------|---------------------|-----------|--------|
| XtCApplTouchItem | OlApplTouchItemProc | (special) | G      |

Synopsis: The system-provided routine the application should use when it changes an item in the list. Applications should use `XtGetValues()` to retrieve this routine.

---

**Note** – This resource cannot be used with the new API and related functions.

---

The prototype for this routine is as follows:

```
typedef void (*OlApplTouchItemProc)(Widget widget,
 OlListToken token);
```

*widget* Identifies the ScrollingList widget instance.

*token* Identifies the item that has changed.

If mapped and if allowed by the application (see `XtNapplUpdateView`), the ScrollingList widget updates the view if the changed item is visible.

***XtNapplUpdateView***

| Class             | Type             | Default   | Access |
|-------------------|------------------|-----------|--------|
| XtCApplUpdateView | OlApplUpdateView | (special) | G      |

Synopsis: The system-provided routine the application can call to keep the ScrollingList widget from updating the view, or to let it again update the view. Applications should use `XtGetValues()` to retrieve this routine.

---

**Note** – This resource cannot be used with the new API and related functions.

---

The prototype for this routine is as follows:

```
typedef void (*OlApplUpdateViewProc)(Widget widget,
 Boolean ok);
```

*widget* Identifies the ScrollingList widget instance.

*ok* Either TRUE or FALSE, depending on whether the ScrollingList can update the View as it changes, or not, respectively.

From the time the `XtNapplUpdateView` routine is called with a FALSE argument until it is called with a TRUE argument, the ScrollingList does not

*ScrollingList Widget*

update the view in response to application-made changes. The *ScrollingList* widget updates the view once for each of these exceptions, each time an exception occurs. An application should use this routine to bracket a set of changes to avoid spurious changes to the view. This routine is not needed if only one change is made to the list. The following example illustrates the use of the *XtNapplUpdateView* routine.

```

 /* Stop view updates. */
(*applUpdateView)(widget, FALSE);
 /* Make some changes. */
(*applDeleteItem)(widget,...);
(*applDeleteItem)(widget,...);
(*applAddItem)(widget,...);
(*applTouchItem)(widget, ...);
 /* Allow the view to be updated again. */
(*applUpdateView)(widget, TRUE);

```

***XtNapplViewItem***

| Class           | Type               | Default   | Access |
|-----------------|--------------------|-----------|--------|
| XtCApplViewItem | OlApplViewItemProc | (special) | G      |

**Synopsis:** The system-provided routine the application can call when it wants a particular item placed in the view. Applications should use *XtGetValues()* to retrieve this routine.

---

**Note** – This resource cannot be used with the new API and related functions.

---

The prototype for this routine is as follows:

```

typedef void (*OlApplViewItemProc)(Widget widget,
 OlListToken token);

```

*widget*    Identifies the *ScrollingList* widget instance.

*token*     Identifies the item to move into the view.

The item is moved into the view in a way that minimizes the change to the view. If the item is currently in the view, nothing is changed. If scrolling the list up or down brings the item into the view while keeping at least one previously viewed item in the view, the list is scrolled. Otherwise, the item is placed at the top of the view, or as close to the top as possible if there aren't enough items in the current Level to fill the view below it. If mapped and if allowed by the application (see "*XtNapplUpdateView*" on page 561), the *ScrollingList* widget updates the view. See also "Known Deficiencies" on page 585.

**XtNcurrentItems**

| Class           | Type             | Default | Access |
|-----------------|------------------|---------|--------|
| XtCCurrentItems | OlSListItemPtr * | NULL    | G      |

Synopsis: The array of OlSListItemPtr items that are current in the list.

XtGetValues() for this resource returns the items that are current and the application must not free these items.

**XtNfirstViewableItem**

| Class                | Type           | Default      | Access |
|----------------------|----------------|--------------|--------|
| XtCFirstViewableItem | OlSListItemPtr | (calculated) | SG     |

Synopsis: Which item is the first viewable item in the list. The default is the first item added to the list.

**XtNitemCurrentCallback**

| Class                  | Type           | Default | Access |
|------------------------|----------------|---------|--------|
| XtCItemCurrentCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The list of callbacks to be invoked when an item is made current by the user pressing the SELECT mouse button over an item.

This callback list is not invoked if XtNscrollingListMode is OL\_NONE. The *call\_data* structure is OlSListCallbackStruct (see page 571) and the reason field in the *call\_data* structure will be set to OL\_REASON\_ITEM\_CURRENT.

**XtNitemHeight**

| Class         | Type      | Default | Access |
|---------------|-----------|---------|--------|
| XtCItemHeight | Dimension | 0       | SGL    |

Synopsis: The height of each item.

Values:  $0 \leq \text{XtNitemHeight}$

If set to zero, the height of each item will be the height of the font specified by XtNfont. If the items in the ScrollingList include glyphs, this resource should be set to the maximum height of all the glyphs.

ScrollingList Widget

***XtNitemNotCurrentCallback***

| Class                     | Type           | Default | Access |
|---------------------------|----------------|---------|--------|
| XtCItemNotCurrentCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the user presses the SELECT mouse button on a current item.

This callback list is not invoked if `XtNscrollingListMode` is `OL_NONE`. The `call_data` structure is `OlSlistCallbackStruct` (see page 571) and the `reason` field in the `call_data` structure will be set to `OL_REASON_ITEM_NOT_CURRENT`.

***XtNlastViewableItem***

| Class               | Type           | Default      | Access |
|---------------------|----------------|--------------|--------|
| XtCLastViewableItem | OlSlistItemPtr | (calculated) | SG     |

Synopsis: The last viewable item in the list.

***XtNlistPane***

| Class       | Type   | Default   | Access |
|-------------|--------|-----------|--------|
| XtCReadOnly | Widget | (special) | G      |

Synopsis: The ListPane child of the ScrollingList.

The ListPane widget is created as a child by the ScrollingList widget during instantiation.

The ListPane widget is subclassed off the Primitive widget. The ScrollingList items are rendered on the ListPane widget. `XtNconsumeEvent` callbacks or event handlers for the ScrollingList must be registered on this widget.

***XtNmultiClickCallback***

| Class                 | Type           | Default | Access |
|-----------------------|----------------|---------|--------|
| XtCMultiClickCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the user presses the SELECT mouse button on an item multiple times.

This callback list is not invoked if `XtNscrollingListMode` is `OL_NONE`. The `call_data` structure is `OlSlistCallbackStruct` (see page 571) and the `reason` field in the `call_data` structure will be set to `OL_REASON_DOUBLE_CLICK`. The `XtNmultiClickTimeout` toolkit resource defines the timer interval in

*ScrollingList Widget*

milliseconds within which two successive button clicks are interpreted as a multclick.

***XtNnumCurrentItems***

| Class              | Type | Default | Access |
|--------------------|------|---------|--------|
| XtCNumCurrentItems | int  | 0       | G      |

Synopsis: The number of current items.

***XtNnumItems***

| Class       | Type | Default | Access |
|-------------|------|---------|--------|
| XtCNumItems | int  | 0       | G      |

Synopsis: The total number of items in the list at any given time.

This resource automatically gets updated by the ScrollingList whenever an item gets added or deleted.

***XtNposition***

| Class       | Type     | Default | Access |
|-------------|----------|---------|--------|
| XtCPosition | OlDefine | OL_LEFT | SGI    |

Synopsis: The position of the string with respect to the glyph in items having both components.

Values: OL\_LEFT/"left" - Sets the string to the left of the glyph.  
OL\_RIGHT/"right" - Sets the string to the right of the glyph.

***XtNprefMaxWidth/  
XtNprefMinWidth***

| Class           | Type      | Default | Access |
|-----------------|-----------|---------|--------|
| XtCPrefMaxWidth | Dimension | 0       | SGI    |
| XtCPrefMinWidth | Dimension | 0       | SGI    |

Synopsis: Controls dynamic resizing.

Values:  $0 \leq \text{XtNprefMaxWidth}$   
 $0 \leq \text{XtNprefMinWidth}$

If the value of these resources is specified as zero, the static sizing behavior is preserved (which is the default). However, if these are specified as nonzero, then the width of the ScrollingList will be no less than `XtNprefMinWidth` and no greater than `XtNprefMaxWidth`, regardless of the length of the items

*ScrollingList Widget*

inside. Items that are longer than `XtNprefMaxWidth` will automatically be marked with a caret, indicating that they are not completely visible.

If the application wishes to set the List to a single width, then `XtNprefMinWidth` should equal `XtNprefMaxWidth`, thus forcing the `ScrollingList` width to a particular size.

***XtNrecomputeWidth***

| Class                          | Type    | Default | Access |
|--------------------------------|---------|---------|--------|
| <code>XtCRecomputeWidth</code> | Boolean | TRUE    | SGI    |

Synopsis: Whether the `ScrollingList` widget should dynamically resize itself in the horizontal direction.

Values: `TRUE/"true"` - The `ScrollingList` grows horizontally to accommodate the widest item in the list.  
`FALSE/"false"` - The `ScrollingList` does not grow horizontally to accommodate changes in item dimensions.

The `ScrollingList` never shrinks horizontally to accommodate changes in item dimensions.

***XtNscrollingListItems***

| Class                              | Type                          | Default | Access |
|------------------------------------|-------------------------------|---------|--------|
| <code>XtCScrollingListItems</code> | <code>OlSlistItemPtr *</code> | NULL    | G      |

Synopsis: The entire list of items in the widget.

The number of items in this list is determined by the resource `XtNnumItems`. If the widget is empty it returns `NULL`. `XtGetValues()` on this resource returns the items themselves and not a copy of the list items. The application must not free the returned items. The application must not free the returned items and must make a copy if this returned list has to be used in any of the convenience functions.

***XtNscrollingListMode***

| Class                             | Type                  | Default              | Access |
|-----------------------------------|-----------------------|----------------------|--------|
| <code>XtCScrollingListMode</code> | <code>OlDefine</code> | <code>OL_NONE</code> | GI     |

Synopsis: The mode of the `ScrollingList`.

Values: `OL_EXCLUSIVE` - Exclusive mode; one item is current at all times.  
`OL_EXCLUSIVE_NONESET` - Exclusive-none-set mode; zero or



---

*ScrollingList Widget*

one item is current at all times.

OL\_NONEXCLUSIVE - Nonexclusive mode; zero, one, or many items are current.

OL\_NONE - Default API mode, for backwards compatibility.

***XtNselectable***

| Class         | Type    | Default | Access |
|---------------|---------|---------|--------|
| XtCSelectable | Boolean | FALSE   | SGI    |

Synopsis: Whether the user can select items in the scrolling list.

Values: TRUE/"true" - Items can be selected with SELECT and ADJUST and copied with the COPY key. Items may be deleted with the CUT key, although the application can stop some or all selected items from being deleted.

FALSE/"false" - Items cannot be selected and the COPY and CUT keys have no effect.

***XtNspace***

| Class    | Type      | Default | Access |
|----------|-----------|---------|--------|
| XtCSpace | Dimension | 4       | SGI    |

Synopsis: The spacing between the string and the glyph in an item.

Values:  $0 \leq \text{XtNspace}$

***XtNtextField***

| Class        | Type   | Default   | Access |
|--------------|--------|-----------|--------|
| XtCTextField | Widget | (special) | G      |

Synopsis: The widget ID of the editable text field widget.

This value is available once the ScrollingList widget has been created. The ScrollingList widget resets the following TextField resources before returning from each invocation of the XtNappEditOpen routine:

| Name      | Class     | Value                      |
|-----------|-----------|----------------------------|
| XtNwidth  | XtCWidth  | Width available in View    |
| XtNstring | XtCString | Name of Item to be changed |

*ScrollingList* Widget

***XtNuserDeleteItems***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The list of callbacks invoked when the user tries to delete items from the list.

Currently, the only way the *ScrollingList* widget handles deletions is through a *cut* operation. The *call\_data* parameter has different values depending on the mode of the *ScrollingList*.

In the default API (`XtNscrollingListMode = OL_NONE`), the *call\_data* parameter points to a structure `OlListDelete` defined as:

```
typedef struct _OlListDelete {
 OlListToken *tokens;
 Cardinal num_tokens;
} OlListDelete;
```

***tokens*** A list identifying the items to be deleted. The application is expected to act on each item separately, calling the `XtNapplDeleteItem` routine to delete each from the list. The application may refuse to delete some or all of the items, and is responsible for providing any feedback to the user.

***num\_tokens*** The number of items to delete.

In the new API (`XtNscrollingListMode ≠ OL_NONE`), the *call\_data* parameter points to a structure `OlSlistUserDeleteCallbackStruct` defined as:

```
typedef struct {
 int reason;
 XEvent *event;
 OlDefine mode;
 OlSlistItemPtr *user_del_items;
 int num_del_items;
} OlSlistUserDeleteCallbackStruct;
```

***reason*** Indicates the reason for the callback.

***mode*** Indicates what mode the scrollinglist is in:  
 OL\_EXCLUSIVE  
 OL\_EXCLUSIVES\_NONESET  
 OL\_NONEXCLUSIVE

---

*ScrollingList* Widget

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>event</i>         | Points to the <code>XEvent</code> that triggered the callback. This member can be <code>NULL</code> .                                                                                                                                                                                                                                                                                                                                          |
| <i>item</i>          | The item that was most recently made <i>current</i> when the event that triggered that callback occurred.                                                                                                                                                                                                                                                                                                                                      |
| <i>item_pos</i>      | Specifies the <code>Position</code> of the item in the list.                                                                                                                                                                                                                                                                                                                                                                                   |
| <i>cur_items</i>     | Specifies a list of current items at the time of the event that triggered the callback. When the <i>mode</i> is <code>OL_NONEXCLUSIVE</code> , this gives a list of all the current items. When the <i>mode</i> is <code>OL_EXCLUSIVE</code> or <code>OL_EXCLUSIVE_NONESSET</code> , this variable has the same value as the field <i>item</i> . This points to temporary storage that the application must copy into its own space if needed. |
| <i>num_cur_items</i> | Gives the number of current items in the list.                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>cur_items_pos</i> | An array of integers, representing one for each current item. This points to temporary storage that the application must copy into its own space if needed.                                                                                                                                                                                                                                                                                    |

and the *reason* field will be set to `OL_REASON_USER_DELETE_ITEMS`. It is up to the application to act on each item by calling `OlSlistDeleteItem()` or `OlSlistDeleteItems()` (see page 579) and providing necessary feedback.

***XtNuserMakeCurrent***

| Class       | Class          | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the user presses `SELECT` over an item.

---

**Note** – This resource cannot be used with the new API and related functions.

---

The *call\_data* parameter is the `OLListToken` value that identifies the item. The application is expected to decide if the current item status of this item should change. The *attributes* member of the `OLListItem` structure for this item is not automatically changed by the *ScrollingList* widget.

*ScrollingList Widget*

***XtNviewableItems***

| <b>Class</b>     | <b>Type</b>      | <b>Default</b> | <b>Access</b> |
|------------------|------------------|----------------|---------------|
| XtCViewableItems | OISListItemPtr * | NULL           | G             |

Synopsis: The list of viewable items in the widget.

The number of items in this list is determined by the `XtNviewHeight` resource. There are three possible cases: if `XtNnumItems` is zero the value returned is NULL. If the value of `XtNnumItems` is greater than zero but less than the value of `XtNviewHeight` then the list item pointer returned will be valid only for the first `XtNnumItems` in the list. Otherwise the list item pointer is valid for all `XtNviewHeight` items.

`XtGetValues()` on this resource returns the items themselves and not a copy of the list items. The application must not free the returned items. The application must make a copy if this returned list has to be used in any of the convenience functions.

***XtNviewHeight***

| <b>Class</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtCViewHeight | Dimension   | (calculated)   | SGI           |

Synopsis: The preferred height of the view as the number of items to show.

Values:  $0 \leq XtNviewHeight$

If a nonzero value is given, the corresponding `XtNheight` resource is computed by converting this number to pixels and adding any padding or border thickness. In this case, any value given in the `XtNheight` resource is overwritten. If a zero value is given in the `XtNviewHeight` resource, the `XtNheight` resource is used as an estimate. The view is sized to show an integral number of items, such that the overall height of the `ScrollingList` widget is less than or equal to `XtNheight`, if possible. However, the view is always large enough to show at least one item, and is no shorter than the minimum scrollbar size. If neither the `XtNviewHeight` resource nor the `XtNheight` resource is set, or both are set to zero, the view is made as small as possible, limited as described above.

## Callback Information

### ***OlSlistCallbackStruct***

This structure is used by a number of the callbacks defined for the new API.

```
typedef struct {
 int reason;
 XEvent *event;
 OlDefine mode;
 OlSlistItemPtr item;
 int item_pos;
 OlSlistItemPtr *cur_items;
 int num_cur_items;
 int *cur_items_pos;
} OlSlistCallbackStruct;
```

|                             |                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>reason</i></b>        | Indicates the reason for the callback.                                                                                                                                                                                                                                                                                                                                                     |
| <b><i>mode</i></b>          | The mode of the scrollinglist:<br>OL_EXCLUSIVE<br>OL_EXCLUSIVES_NONESET<br>OL_NONEXCLUSIVE                                                                                                                                                                                                                                                                                                 |
| <b><i>event</i></b>         | Points to the XEvent that triggered the callback. This member can be NULL.                                                                                                                                                                                                                                                                                                                 |
| <b><i>item</i></b>          | The item that was most recently made <i>current</i> when the event that triggered that callback occurred.                                                                                                                                                                                                                                                                                  |
| <b><i>item_pos</i></b>      | The Position of the item in the list.                                                                                                                                                                                                                                                                                                                                                      |
| <b><i>cur_items</i></b>     | A list of current items at the time of the event that triggered the callback. When the <i>mode</i> is OL_NONEXCLUSIVE, this gives a list of all the current items. When the <i>mode</i> is OL_EXCLUSIVE or OL_EXCLUSIVE_NONESET, this variable has the same value as the field <i>item</i> . This points to temporary storage that the application must copy into its own space if needed. |
| <b><i>num_cur_items</i></b> | The number of current items in the list.                                                                                                                                                                                                                                                                                                                                                   |
| <b><i>cur_items_pos</i></b> | An array of integers, representing one for each current item. This points to temporary storage that the application must copy into its own space if needed. This field is currently not implemented in OLIT.                                                                                                                                                                               |

**OlSlistItemAttrs**

The item data structure `OlSlistItemAttrs` is used in the `OlSlistAddItem()` convenience function to indicate the attributes of interest for a `ScrollingList` item. See also “`OlSlistGetItemAttrs`” on page 580 and “`OlSlistSetItemAttrs`” on page 584.

```
typedef struct {
 long flags;
 OlDefine label_type;
 OlStr item_label;
 XImage *item_image;
 Boolean item_sensitive;
 Boolean item_current;
 unsigned char item_mnemonic;
 XtPointer user_data;
} OlSlistItemAttrs, *OlSlistItemAttrsPtr;
```

- flags** Marks which fields in this structure are defined, as the bitwise inclusive OR of the following:
  - #define OlItemLabelType (1L << 0)
  - #define OlItemLabel (1L << 1)
  - #define OlItemImage (1L << 2)
  - #define OlItemSensitive (1L << 3)
  - #define OlItemCurrent (1L << 4)
  - #define OlItemMnemonic (1L << 5)
  - #define OlItemUserData (1L << 6)
- label\_type** The type of label to display for the Item in the view. It can have one of the values:
  - OL\_STRING for a text label
  - OL\_IMAGE for an image label
  - OL\_BOTH for a text and image label.
- item\_label** The string to display for the Item in the view.
- item\_image** The glyph to be displayed for the item in the view. The type of the value of this member depends on the value of the *label\_type* member—if `OL_BOTH` or `OL_IMAGE`, a glyph is displayed.
- item\_sensitive** TRUE if the item is sensitive. See also “`OlSlistGetItemSensitivity`” on page 581.
- item\_current** TRUE if the item is current. See also “`OlSlistIsItemCurrent`” on page 582.
- item\_mnemonic** A single character that is used as a mnemonic accelerator for keyboard traversal.
- user\_data** A pointer to an area available for application-specific data.

**OLListItem Structure**

Several of the resources defined for the default API use the following OLListItem structure:

```
typedef struct _OLListItem {
 OlDefine label_type;
 XtPointer label;
 XImage *glyph;
 OlBitMask attr;
 XtPointer user_data;
 unsigned char mnemonic;
} OLListItem;
```

|                   |                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>label_type</b> | Identifies the type of label to display for the Item in the view. It can have one of the values:<br>OL_STRING if the label consists only of text<br>OL_IMAGE if the label consists only of an image<br>OL_BOTH if the label comprises both text and image                                                                                          |
| <b>label</b>      | Specifies the string to display for the Item in the view.                                                                                                                                                                                                                                                                                          |
| <b>glyph</b>      | Specifies the glyph to be displayed for the item in the view. The type of the value of this member depends on the value of the <i>label_type</i> member—if OL_BOTH or OL_IMAGE, a glyph is displayed.                                                                                                                                              |
| <b>attr</b>       | Defines attributes of the Item. It is a 32-bit vector with the field:<br>OL_LIST_ATTR_APPL—available for application use. This field consists of the low-order 16 bits, to be used as the application sees fit.<br>OL_LIST_ATTR_CURRENT—if the Item is a current item.<br>Other bit values are defined, but should not be used by the application. |
| <b>mnemonic</b>   | A single character that is used as a mnemonic accelerator for keyboard traversal.                                                                                                                                                                                                                                                                  |
| <b>user_data</b>  | A pointer to an area available for application-specific data.                                                                                                                                                                                                                                                                                      |

**OLListToken Structure**

In the default API, the ScrollingList widget identifies each item with a “token” of type OLListToken. The ScrollingList widget assigns the token when an item is added by the application, and the application uses the token in later references to the item. A zero value is allowed in some contexts where an OLListToken is expected, as a way to refer to no item.

*ScrollingList Widget*

As a convenience to the application, the macro `OlListItemPointer(token)` converts an `OlListToken` value into a pointer to the corresponding `OlListItem`. The application can change the values of the `OlListItem` members, but should let the `ScrollingList` widget know that they have changed, using the `XtNapplTouchItem` routine. No checking is done for incorrect `OlListToken` arguments to the `OlListItemPointer` macro.

The `OlListToken` value can be coerced into the type `caddr_t` and back without loss of precision.

*Activation Types*

The following table lists the activation types used by the `ScrollingList`.

*Table 9-27 ScrollingList Activation Types*

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_ADJUST              | ADJUST           | XtNadjustBtn         |
| OL_ADJUSTKEY           | ADJUST           | XtNadjustKey         |
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_COPY                | COPY             | XtNcopyKey           |
| OL_CUT                 | CUT              | XtNcutKey            |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MENU                | MENU             | XtNmenuBtn           |
| OL_MENUKEY             | MENU             | XtNmenuKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_MULTIDOWN           | JUMP DOWN        | XtNmultiDownKey      |
| OL_MULTIUPT            | JUMP UP          | XtNmultiUpKey        |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PAGEDOWN            | ALT+PANE DOWN    | XtNpageDownKey       |
| OL_PAGEUP              | ALT+PANE UP      | XtNpageUpKey         |
| OL_PANEEND             | PANE DOWN        | XtNpaneEndKey        |
| OL_PANESTART           | PANE UP          | XtNpaneStartKey      |
| OL_PASTE               | PASTE            | XtNpasteKey          |



Table 9-27 ScrollingList Activation Types (Continued)

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_PREVFIELD     | PREVFIELD     | XtNprevFieldKey     |
| OL_SCROLLBOTTOM  | DATA END      | XtNscrollBottomKey  |
| OL_SCROLLDOWN    | ALT+DOWN      | XtNscrollDownKey    |
| OL_SCROLLTOP     | DATA START    | XtNscrollTopKey     |
| OL_SCROLLUP      | ALT+UP        | XtNscrollUpKey      |
| OL_SELECT        | SELECT        | XtNselectBtn        |
| OL_SELECTKEY     | SELECT        | XtNselectKey        |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |

Activation types not described in the following table are described in “Common Activation Types” on page 68.

### ***OL\_ADJUST/ OL\_ADJUSTKEY***

These activation types can be used to make additional list items current. They apply only if the ScrollingList is selectable. If the XtNtextField widget is being edited, editing will be terminated and its XtNverification callback (if any) will be called. If other list items are already selected (highlighted), they will be cleared. The current item will become the current selection, and will be highlighted if it is not already. If the item has an XtNuserMakeCurrent callback, it will be called.

### ***OL\_COPY***

This activation type will copy the currently selected items to the clipboard as a <newline>-separated list of text items in the order they appear. The OL\_LIST\_ATTR\_CURRENT *attr* field of the currently selected items will be cleared.

### ***OL\_CUT***

This activation type applies only to ScrollingLists that are editable. The currently selected items are moved to the clipboard as a <newline>-separated list of text items in the order they appear in the ScrollingList. The XtNuserDeleteItems callback will be called with an OlListDelete structure containing the currently selected items.

---

*ScrollingList Widget****OL\_MENU/  
OL\_MENUKEY***

These activation types will post the ScrollingList menu.

***OL\_MULTIDOWN/  
OL\_PAGEDOWN***

These activation types will scroll the list down a number of items equivalent to the current `XtNviewHeight`.

***OL\_MULTIUP/  
OL\_PAGEUP***

These activation types will scroll the list up a number of items equivalent to the current `XtNviewHeight`.

***OL\_PANEEND***

This activation will type make the last list item the current item.

***OL\_PANESTART***

This activation type will make the first list item the current item.

***OL\_PASTE***

This activation type applies only to an editable ScrollingList. When the `XtNtextField` widget has focus within an editable ScrollingList, then the `OL_PASTE` activation type will insert the contents of the clipboard at the current insert position.

***OL\_SCROLLBOTTOM***

This activation type will scroll the list so the last item is at the bottom of the view and update the scrollbar appropriately.

***OL\_SCROLLDOWN***

If the list is not already at the bottom, this activation type will scroll the list down one item, updating the list view and scrollbar.

***OL\_SCROLLTOP***

This activation type will scroll the list so the first item is at the top of the view, and updates the scrollbar appropriately.

***OL\_SCROLLUP***

If the list is not already at the top, this activation type will scroll the list up one item, updating the list view and scrollbar.

***OL\_SELECT/  
OL\_SELECTKEY***

These activation types can be used to make one or more list items current. They apply only if the `ScrollingList` is selectable. If the `XtNtextField` widget is being edited, editing will be terminated and its `XtNverification` callback (if any) will be called. If other list items are already selected (highlighted), they will be cleared. The item will become the current selection, and will be highlighted if it is not already. If the item has an `XtNuserMakeCurrent` callback, it will be called.

***See Also***

“Scrollbar Widget” on page 508,  
“ScrollingList Functions” on page 578,  
“TextField Widget” on page 665.

## ScrollingList Functions

The following convenience functions are provided to manipulate the ScrollingList widget in the *new API* mode; see “ScrollingList Modes” on page 552.

### *OlSlistAddItem*

```
#include <Xol/ScrollingL.h>
OlSlistItemPtr OlSlistAddItem(
 Widget widget,
 OlSlistItemAttrsPtr item_attr,
 OlSlistItemPtr ref_item);
```

`OlSlistAddItem()` adds an item before *ref\_item*, given the attributes of the item, and returns a pointer to the item. The value of *ref\_item* can be zero to append the new item to the bottom of list. The content of *item\_attr* will be copied by the ScrollingList widget into space that it maintains; however, the data pointed to by the *item\_image* member will not be copied. The application must use the `OlSlistGetItemAttrs()` and `OlSlistSetItemAttrs()` functions to get and set the data associated with an item and then use the `OlSlistTouchItem()` function to inform the ScrollingList widget that the data has been changed. If mapped and if allowed by the application, the ScrollingList widget updates the view if the new item will be in the view. The view is changed as little as possible: if the new item is in the upper half of the view, the items above it are scrolled up and the top item is scrolled off; if the new item is in the lower half of the view, the items below it are scrolled down and the bottom item is scrolled off.

### *OlSlistDeleteAllItems*

```
#include <Xol/ScrollingL.h>
void OlSlistDeleteAllItems(Widget widget);
```

`OlSlistDeleteAllItems()` deletes all items from the Scrollinglist.

### ***OlSlistDeleteItem***

```
#include <Xol/ScrollingL.h>
void OlSlistDeleteItem(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistDeleteItem()` deletes *item* from the `ScrollingList`.

### ***OlSlistDeleteItems***

```
#include <Xol/ScrollingL.h>
void OlSlistDeleteItems(
 Widget widget,
 OlSlistItemPtr *items,
 int num_items);
```

`OlSlistDeleteItems()` deletes *items* from the `ScrollingList`.

### ***OlSlistEditItem***

```
#include <Xol/ScrollingL.h>
void OlSlistEditItem(
 Widget widget,
 Boolean insert,
 OlSlistItemPtr ref_item);
```

`OlSlistEditItem()` allows programmatic editing of a `ScrollingList` item. If *insert* is TRUE, then *ref\_item* identifies the item before which a new item is to be inserted. If *insert* is FALSE, then *ref\_item* refers to the item that should be edited and must be a valid item in the list.

### ***OlSlistEndEdit***

```
#include <Xol/ScrollingL.h>
void OlSlistEndEdit(Widget widget);
```

`OlSlistEndEdit()` should be called by the application when the user has finished editing an item.

***OlSlistFirstViewableItem***

```
#include <Xol/ScrollingL.h>
void OlSlistFirstViewableItem(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistFirstViewableItem()` sets *item* as the first viewable item in the list.

***OlSlistGetItemAttrs***

```
#include <Xol/ScrollingL.h>
void OlSlistGetItemAttrs(
 Widget widget,
 OlSlistItemPtr item,
 OlSlistItemAttrs *get_attrs);
```

`OlSlistGetItemAttrs()` returns the item attributes of *item* in *get\_attrs*. The *flags* field must be set appropriately in *get\_attrs*. The application must not free any members of the *get\_attrs* field and must make copies of the data as needed.

***OlSlistGetItemImage***

```
#include <Xol/ScrollingL.h>
XImage *OlSlistGetItemImage(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistGetItemImage()` returns the image corresponding to *item*, or NULL if the item does not have an associated image.

***OlSlistGetItemLabel***

```
#include <Xol/ScrollingL.h>
OlStr OlSlistGetItemLabel(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistGetItemLabel()` returns the label corresponding to *item*, or NULL if the item does not have an associated label.

### *OlSlistGetItemSensitivity*

```
#include <Xol/ScrollingL.h>
Boolean OlSlistGetItemSensitivity(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistGetItemSensitivity()` returns TRUE if the *item* is sensitive; otherwise, it returns FALSE.

### *OlSlistGetItemType*

```
#include <Xol/ScrollingL.h>
OlDefine OlSlistGetItemType(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistGetItemType()` returns the item type, given the *item*. Valid values are:

- OL\_STRING if *item* just has a string
- OL\_IMAGE if *item* just has a image
- OL\_BOTH if *item* has both string and image
- OL\_NONE if *item* is not a valid item in the list

### *OlSlistGetItemUserData*

```
#include <Xol/ScrollingL.h>
XtPointer OlSlistGetItemUserData(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistGetItemUserData()` returns the *item's user\_data*, if any.

### *OlSlistGetMode*

```
#include <Xol/ScrollingL.h>
OlDefine OlSlistGetMode(Widget widget);
```

`OlSlistGetMode()` returns the mode of the ScrollingList:

OL\_EXCLUSIVE                      Exclusive mode; one item is current at all times.

---

## ScrollingList Functions

|                      |                                                                    |
|----------------------|--------------------------------------------------------------------|
| OL_EXCLUSIVE_NONESET | Exclusive-none-set mode; zero or one item is current at all times. |
| OL_NONEXCLUSIVE      | Nonexclusive mode; zero, one, or many items are current.           |
| OL_NONE              | Default API mode, for backwards compatibility.                     |

### *OlSlistGetNextItem*

```
#include <Xol/ScrollingL.h>
OlSlistItemPtr OlSlistGetNextItem(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistGetNextItem()` returns the next item, given a valid item. If *item* is the last item in the `ScrollingList`, then it returns `NULL`.

### *OlSlistGetPrevItem*

```
#include <Xol/ScrollingL.h>
OlSlistItemPtr OlSlistGetPrevItem(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistGetPrevItem()` returns the previous item, given a valid *item*. If *item* is the first item in the `ScrollingList`, then it returns `NULL`.

### *OlSlistIsItemCurrent*

```
#include <Xol/ScrollingL.h>
Boolean OlSlistIsItemCurrent(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistIsItemCurrent()` returns `TRUE` if the *item* is current; otherwise, it returns `FALSE`.



### ***OISlistIsValidItem***

```
#include <Xol/ScrollingL.h>
Boolean OISlistIsValidItem(
 Widget widget,
 OISlistItemPtr item);
```

`OISlistIsValidItem()` returns TRUE if the specified *item* is a valid item in the list; otherwise, it returns FALSE.

### ***OISlistLastViewableItem***

```
#include <Xol/ScrollingL.h>
void OISlistLastViewableItem(
 Widget widget,
 OISlistItemPtr item);
```

`OISlistLastViewableItem()` sets *item* to be the last viewable item in the list.

### ***OISlistMakeAllItemsNotCurrent***

```
#include <Xol/ScrollingL.h>
void OISlistMakeAllItemsNotCurrent(Widget widget);
```

`OISlistMakeAllItemsNotCurrent()` makes all items not current.

### ***OISlistMakeItemCurrent***

```
#include <Xol/ScrollingL.h>
void OISlistMakeItemCurrent(
 Widget widget,
 OISlistItemPtr item,
 Boolean issue_callback);
```

`OISlistMakeItemCurrent()` makes an item current, adds it to the list of current items, and issues a current callback if the Boolean value *issue\_callback* is TRUE.

### *OlSlistMakeItemNotCurrent*

```
#include <Xol/ScrollingL.h>
void OlSlistMakeItemNotCurrent(
 Widget widget,
 OlSlistItemPtr item,
 Boolean issue_callback);
```

`OlSlistMakeItemNotCurrent()` makes an *item* not current in the `ScrollingList` and also removes it from the current list. It issues an `XtNitemNotCurrentCallback` if *issue\_callback* is TRUE.

### *OlSlistSetItemAttrs*

```
#include <Xol/ScrollingL.h>
void OlSlistSetItemAttrs(
 Widget widget,
 OlSlistItemPtr item,
 OlSlistItemAttrs *set_attrs);
```

`OlSlistSetItemAttrs()` sets the attributes pointed to in *set\_attrs* on the specified *item*. The *flags* field must be appropriately set in *set\_attrs*. The widget then copies this data into space that it maintains. The data pointed to by the *item\_image* member of *set\_attrs* will not be copied.

### *OlSlistTouchItem*

```
#include <Xol/ScrollingL.h>
void OlSlistTouchItem(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistTouchItem()` should be used by the application when it changes the attributes of an item. If mapped and if allowed by the application (see “`OlSlistUpdateView`” on page 585), the `ScrollingList` widget updates the view if the changed *item* is visible.

### *OlSlistUpdateView*

```
#include <Xol/ScrollingL.h>
void OlSlistUpdateView(
 Widget widget,
 Boolean ok);
```

`OlSlistUpdateView()` can be called by the application to keep the `ScrollingList` widget from updating the view or to let it again update the view. If `ok` is `TRUE`, the `ScrollingList` widget can update the View as it changes; otherwise, it cannot update the view.

### *OlSlistViewItem*

```
#include <Xol/ScrollingL.h>
void OlSlistViewItem(
 Widget widget,
 OlSlistItemPtr item);
```

`OlSlistViewItem()` can be called by the application to place a particular `ScrollingList` item in view. See description of the `XtNapplViewItem` resource in the `ScrollingList` widget section for more details.

### *Known Deficiencies*

There is no programmatic interface equivalent to the `XtNapplViewItem` resource (such as `OlSlistUpdateView()`'s relationship to `XtNapplUpdateView`). The following workaround is available:

```
OlApplViewItemProc ViewItem;
OlSlistItemPtr item; /* the item to be in view */
...
XtVaGetValues(slist, XtNapplViewItem, &ViewItem, NULL);
(*ViewItem)(slist, item); /* this brings 'item' into view */
```

*Slider Widget*  
*Slider Widget*

**Class**

*Class Name:* Slider  
*Class Pointer:* sliderWidgetClass

**Ancestry**

Core-Primitive-Slider

**Required Header Files**

```
#include <Xol/OpenLook>
#include <Xol/Slider.h>
```

**Description**

The Slider widget enables the user to set a numeric value graphically.

**Components**

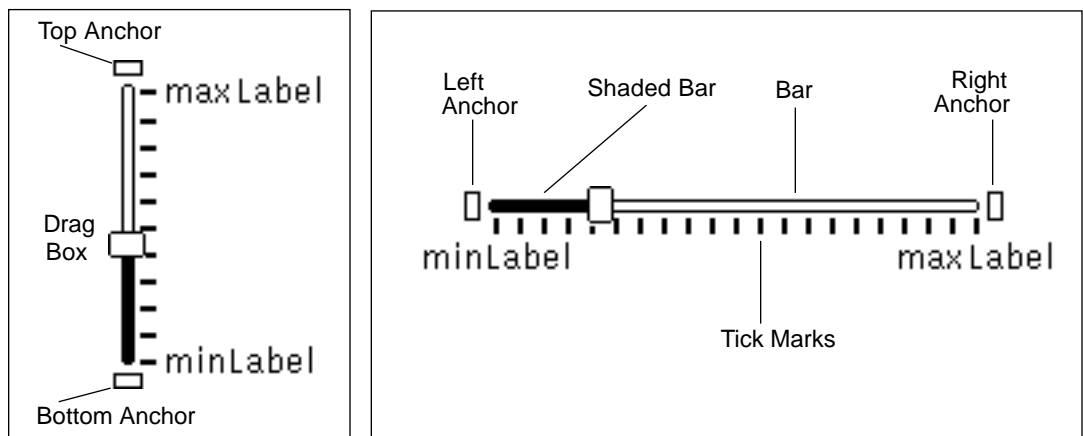


Figure 9-14 Vertical and Horizontal Sliders

A Slider consists of the following elements:

- Top (left) anchor
- Bottom (right) anchor
- Drag box
- Bar (typically)
- Shaded bar (typically)
- Current value (not visible)
- Minimum value (not visible)
- Maximum value (not visible)

The current value is the numeric value a user attempts to change with the Slider widget.

### ***Drag Box Motion***

As visual feedback to the user, the drag box moves up or down (or left or right) along the bar as the current value changes.

### ***Dragging SELECT***

The user can change the current value by dragging the drag box with SELECT. The pressing of SELECT must start with the pointer in the drag box, but the drag box (and the current value) track the pointer motion regardless of where it goes while SELECT is pressed. This means it is not possible for the user to change the current value by first pressing SELECT outside the drag box and then moving the pointer into it. Only the component of the pointer motion parallel to the Bar is tracked, and the motion of the drag box (and change in the current value) are limited by the length of the Bar.

### ***Clicking SELECT***

Clicking SELECT above the drag box for a vertical slider, or to the right for a horizontal slider, increases the current value by an application-specified amount, moves the drag box to correspond to the new current value, and moves the pointer to keep it on the drag box.

Clicking SELECT to the other side of the drag box decreases the value by the same amount and moves the drag box and pointer accordingly.

Pressing SELECT repeats this action.

***Moving Drag Box to Limits***

Clicking SELECT on one of the bottom/left or top/right anchors causes the current value to take on the minimum value or maximum value, respectively, and moves the drag box to the limit in the direction of the Anchor. If the drag box is already at the limit, nothing happens.

Clicking SELECT on an anchor highlights the anchor while the current value is changed.

***Application Notification***

The application gets notified about a change in the current value on the release of SELECT for either the drag or click. It is responsible for providing any feedback to the user deemed appropriate, such as updating the current value in a text field.

***Coloration***

For 3D and 2D, the area surrounding the Slider and its labels is drawn with the parent's `XtNbackground`. `XtNfontColor` is used to draw the minimum and maximum labels.

For 3D, the slider, endbox, and tickmark coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, "Color and Three-Dimensional Design." `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1. `XtNforeground` is used to draw the current-value indicator bar on the Slider.

For 2D, `XtNbackground` and `XtNforeground` are used to render the slider, endboxes, and tickmarks as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, "Controls."

If the toolkit resource `XtNmouseless` is set to TRUE and the toolkit resource `XtNinputFocusFeedback` is set to `OL_INPUT_FOCUS_COLOR`, then the background of the Slider will be drawn with the value of `XtNinputFocusColor` when it receives input focus. However, if `XtNinputFocusColor` is the same as `XtNbackground`, then the Slider inverts `XtNforeground` and `XtNbackground`. Once the input focus leaves the Slider, the original coloration is restored.

### ***Keyboard Traversal***

The Slider's default value of the `XtNtraversalOn` resource is `TRUE`.

The user can operate the Slider by using the keyboard to move the Drag Box and access the Anchors. The following keys manipulate the Current Value:

- `SCROLLUP` and `SCROLLRIGHT` increase the Current Value by an application-specified amount, and move the Drag Box to correspond to the new Current Value.
- `SCROLLEDOWN` and `SCROLLELEFT` decrease the Current Value by an application-specified amount, and move the Drag Box to correspond to the new Current Value.
- `SCROLLTOP` and `SCROLLRIGHTEDGE` cause the Current Value to take on the Maximum Value, and move the Drag Box to a vertical slider's top anchor or a horizontal slider's right anchor. The anchor is briefly highlighted while the Current Value is changed and the Drag Box is moved.
- `SCROLLBOTTOM` and `SCROLLELEFTEDGE` cause the Current Value to take on the Minimum Value, and move the Drag Box to a vertical slider's bottom anchor or a horizontal slider's left anchor. The anchor is briefly highlighted while the Current Value is changed and the Drag Box is moved.

The Slider widget responds to the following keyboard navigation keys:

- `NEXTFIELD`, `MOVEDOWN`, and `MOVERIGHT` move to the next traversable widget in the window
- `PREVFIELD`, `MOVEUP`, and `MOVELEFT` move to the previous traversable widget in the window
- `NEXTWINDOW` moves to the next window in the application
- `PREVWINDOW` moves to the previous window in the application
- `NEXTAPP` moves to the first window in the next application
- `PREVAPP` moves to the first window in the previous application

### ***Keyboard Mnemonic Display***

The Slider does not display the mnemonic accelerator. If the Slider is the child of a Caption widget, the Caption widget will display the mnemonic as part of the label.

**Keyboard Accelerator Display**

The Slider does not respond to a keyboard accelerator because the results of clicking the SELECT button on a Slider depends on the pointer position. So, the Slider does not display a keyboard accelerator.

**Resources**

Table 9-28 Slider Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | 0                   | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | 0                   | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

Table 9-29 Slider Primitive Resources

| Name               | Type           | Default       | Access |
|--------------------|----------------|---------------|--------|
| XtNaccelerator     | String         | NULL          | SGI    |
| XtNacceleratorText | String         | NULL          | SGI    |
| XtNconsumeEvent    | XtCallbackList | NULL          | SGIO   |
| XtNfont            | OlFont         | XtDefaultFont | SGID   |



Table 9-29 Slider Primitive Resources (Continued)

| Name               | Type          | Default             | Access |
|--------------------|---------------|---------------------|--------|
| XtNfontColor       | Pixel         | XtDefaultForeground | SGID   |
| XtNforeground      | Pixel         | XtDefaultForeground | SGID   |
| XtNinputFocusColor | Pixel         | Red                 | SGID   |
| XtNmnemonic        | unsigned char | '\0'                | SGI    |
| XtNreferenceName   | String        | NULL                | GI     |
| XtNreferenceWidget | Widget        | NULL                | GI     |
| XtNscale           | int           | 12                  | SGI    |
| XtNtextFormat      | OlStrRep      | OL_SB_STR_REP       | GI     |
| XtNtraversalOn     | Boolean       | TRUE                | SGI    |
| XtNuserData        | XtPointer     | NULL                | SGI    |

Table 9-30 Slider Resources

| Name                 | Type           | Default       | Access |
|----------------------|----------------|---------------|--------|
| XtNdragCBType        | OlDefine       | OL_CONTINUOUS | SGI    |
| XtNendBoxes          | Boolean        | TRUE          | SGI    |
| XtNgranularity       | int            | 1             | SGI    |
| XtNinitialDelay      | int            | 500           | SGI    |
| XtNmaxLabel          | OlStr          | NULL          | SGI    |
| XtNminLabel          | OlStr          | NULL          | SGI    |
| XtNorientation       | OlDefine       | OL_VERTICAL   | GI     |
| XtNpointerWarping    | Boolean        | TRUE          | SGI    |
| XtNrecomputeSize     | Boolean        | FALSE         | SGI    |
| XtNrepeatRate        | int            | 100           | SGI    |
| XtNsliderMax         | int            | 100           | SGI    |
| XtNsliderMin         | int            | 0             | SGI    |
| XtNsliderMoved       | XtCallbackList | NULL          | SGIO   |
| XtNsliderValue       | int            | 0             | SGI    |
| XtNspan              | Dimension      | OL_IGNORE     | SGI    |
| XtNstopPosition      | OlDefine       | OL_ALL        | SGI    |
| XtNticks             | int            | 0             | SGI    |
| XtNtickUnit          | OlDefine       | OL_NONE       | SGI    |
| XtNuseSetValCallback | Boolean        | FALSE         | SGI    |

***XtNdragCBType***

| Class         | Type     | Default       | Access |
|---------------|----------|---------------|--------|
| XtCDragCBType | OlDefine | OL_CONTINUOUS | SGI    |

Synopsis: The frequency of issuing XtNsliderMoved callbacks during a drag operation.

Values: OL\_CONTINUOUS/"continuous" - Issue callbacks continuously.  
 OL\_GRANULARITY/"granularity" - Issue callbacks only when the drag box crosses any granularity positions.  
 OL\_RELEASE/"release" - Issue callbacks only once when the SELECT button is released.

***XtNendBoxes***

| Class       | Type    | Default | Access |
|-------------|---------|---------|--------|
| XtCEndBoxes | Boolean | TRUE    | SGI    |

Synopsis: Whether the end boxes are displayed.

Values: TRUE/"true" - Display end boxes.  
 FALSE/"false" - Do not display end boxes.

***XtNgranularity***

| Class          | Type | Default | Access |
|----------------|------|---------|--------|
| XtCGranularity | int  | 1       | SGI    |

Synopsis: How much clicking SELECT changes the current value.

Values:  $1 \leq \text{XtNgranularity} \leq (\text{XtNsliderMax} - \text{XtNsliderMin})$

Clicking SELECT on the bar or shaded bar attempts to change the current value by the amount given in this resource. Dragging the drag box with SELECT changes the current value by this amount before the XtNsliderMoved callbacks are issued.

***XtNinitialDelay***

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCInitialDelay | int  | 500     | SGI    |

Synopsis: The time, in milliseconds, before the first action occurs when SELECT is pressed on the bar or shaded bar.

Values:  $0 < \text{XtNinitialDelay}$

***XtNmaxLabel***

| Class    | Type  | Default | Access |
|----------|-------|---------|--------|
| XtCLabel | O1Str | NULL    | SGI    |

Synopsis: The label to be placed next to the maximum value position.

Values: Any O1Str value valid in the current locale.

For a vertical slider, the label will be placed to the right of the minimum value position. If there is not enough space for the entire label and `XtNrecomputeSize` is FALSE, the label will be truncated from the right end. If there is not enough space for the entire label and `XtNrecomputeSize` is TRUE, then the widget will request more space to show the entire label.

For a horizontal slider, the label will be placed centered and below the maximum value position. If there is not enough room to center the label and `XtNrecomputeSize` is set to FALSE, the right end of the label will be aligned with the right anchor. If this label collides with the `XtNminLabel`, some part of the labels will overlap. If there is not enough room to center the label and `XtNrecomputeSize` is set to TRUE, the widget will request more space to center the label below the maximum value position.

***XtNminLabel***

| Class    | Type  | Default | Access |
|----------|-------|---------|--------|
| XtCLabel | O1Str | NULL    | SGI    |

Synopsis: The label to be placed next to the minimum value position.

Values: Any O1Str value valid in the current locale.

For a vertical slider, the label is placed to the right of the minimum value position. If there is not enough space for the entire label and `XtNrecomputeSize` is FALSE, the label will be truncated from the right end. If there is not enough space for the entire label and `XtNrecomputeSize` is TRUE, then the widget will request more space to show the entire label.

For a horizontal slider, the label will be placed centered and below the minimum value position. If there is not enough room to center the label and `XtNrecomputeSize` is set to FALSE, the beginning of the label will be aligned with the left anchor and will be drawn to the right. If this label collides with the `XtNmaxLabel`, some part of the labels will overlap. If there is not enough room to center the label and `XtNrecomputeSize` is set to TRUE, the widget will request more space to center the label below the minimum value position.

*Slider Widget*

***XtNorientation***

| <b>Class</b>   | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|----------------|-------------|----------------|---------------|
| XtCOrientation | OIDefine    | OL_VERTICAL    | GI            |

Synopsis: The direction for the visual presentation of the widget.

Values: OL\_HORIZONTAL/"horizontal" - Define a horizontal slider.  
 OL\_VERTICAL/"vertical" - Define a vertical slider.

***XtNpointerWarping***

| <b>Class</b>      | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtCPointerWarping | Boolean     | TRUE           | SGI           |

Synopsis: Whether the pointer will warp; i.e., follow the drag box as it moves.

Values: TRUE/"true" - The pointer will warp.  
 FALSE/"false" - The pointer will not warp.

***XtNrecomputeSize***

| <b>Class</b>     | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|------------------|-------------|----------------|---------------|
| XtCRecomputeSize | Boolean     | FALSE          | SGI           |

Synopsis: Whether the Slider widget should resize itself whenever needed, to compensate for the space needed to show the tick marks and the labels.

Values: TRUE/"true" - The Slider will resize itself.  
 FALSE/"false" - The Slider will not resize itself.

The Slider widget uses the value of XtNspan, the sizes of the labels, and XtNtickUnit to determine the preferred size.

***XtNrepeatRate***

| <b>Class</b>  | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|---------------|-------------|----------------|---------------|
| XtCRepeatRate | int         | 100            | SGI           |

Synopsis: The time, in milliseconds, between repeated actions when SELECT is pressed on the bar or shaded bar.

Values: 0 < XtNrepeatRate

**XtNsliderMax/  
XtNsliderMin**

| Class        | Type | Default | Access |
|--------------|------|---------|--------|
| XtCSliderMax | int  | 100     | SGI    |
| XtCSliderMin | int  | 0       | SGI    |

Synopsis: The range of values tracked by the Slider widget.

Values:  $XtNsliderMin < XtNsliderMax$

Mathematically, the range is closed on both ends; that is, the range is the following subset of the set of integers:

$$XtNsliderMin \leq range \leq XtNsliderMax$$

This is independent of the drag box displayed in the Slider widget. The Slider widget takes into account the size of the drag box when relating the physical range of movement to the range of values.

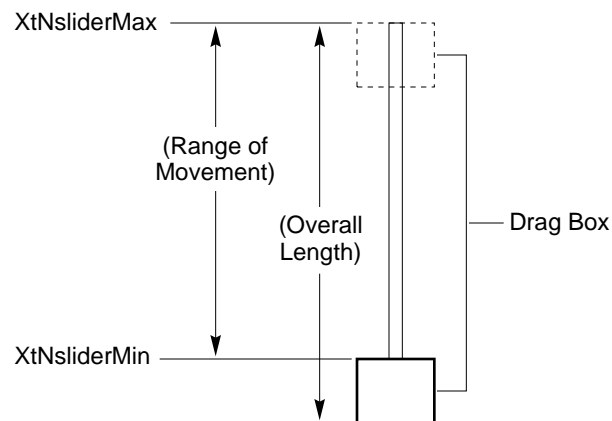


Figure 9-15 Drag Box Range of Movement

**XtNsliderMoved**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the Slider widget is manipulated.

*Slider Widget*

The *call\_data* parameter is a pointer to a structure of the form:

```
typedef struct OlSliderVerify {
 int new_location;
 Boolean more_cb_pending;
} OlSliderVerify;
```

- new\_location*      The new location of the Slider.
- more\_cb\_pending*    Specifies if there are more callbacks pending.

An `XtGetValues()` inside the callback will return the previous value.

***XtNsliderValue***

| Class          | Type | Default | Access |
|----------------|------|---------|--------|
| XtCSliderValue | int  | 0       | SGI    |

Synopsis:    The current position of the drag box.  
 Values:      $XtNsliderMin \leq XtNsliderValue \leq XtNsliderMax$

The Slider widget keeps this resource up to date; however, an application can also get the current value through the `XtNsliderMoved` callbacks.

***XtNspan***

| Class   | Type      | Default   | Access |
|---------|-----------|-----------|--------|
| XtCspan | Dimension | OL_IGNORE | SGI    |

Synopsis:    If `XtNrecomputeSize` is set to TRUE, then `XtNspan` should be set to reflect the preferred length of the slider, not counting the space needed for the labels. The Slider widget uses the span value, the sizes of the labels, and `XtNtickUnit` to determine the preferred size.

***XtNstopPosition***

| Class           | Type     | Default | Access |
|-----------------|----------|---------|--------|
| XtCStopPosition | OlDefine | OL_ALL  | SGI    |

Synopsis:    The behavior of the drag box at the end of a drag operation.  
 Values:     `OL_ALL/"all"`    - At the end of a drag operation, the drag box will be positioned where it stops.  
              `OL_GRANULARITY/"granularity"`    - The drag box will snap to the nearest granularity position.  
              `OL_TICKMARK/"tickmark"`    - The drag box will snap to the nearest tickmark position.

**XtNticks**

| Class    | Type | Default | Access |
|----------|------|---------|--------|
| XtCTicks | int  | 0       | SGI    |

Synopsis: The interval between tick marks.

Values: The unit of the interval value is determined by XtNtickUnit.

**XtNtickUnit**

| Class       | Type     | Default | Access |
|-------------|----------|---------|--------|
| XtCTickUnit | OlDefine | OL_NONE | SGI    |

Synopsis: The number of units of the interval between tick marks.

Values: OL\_NONE/"none" - The widget will display no tick marks and will ignore XtNticks.  
 OL\_SLIDERVALUE/"slidervalue" - The widget will interpret XtNticks as the same unit as the slider value.  
 OL\_PERCENT/"percent" - The widget will interpret XtNticks as the percentage of the slider value range.

**XtNuseSetValCallback**

| Class                | Type    | Default | Access |
|----------------------|---------|---------|--------|
| XtCUseSetValCallback | Boolean | FALSE   | SGI    |

Synopsis: Whether a XtNsliderMoved callback is called when Slider is manipulated by doing an XtSetValues() call.

Values: TRUE/"true" - The callback will be called.  
 FALSE/"false" - The callback will not be called.

**Activation Types**

The following table lists the activation types used by the Slider.

Table 9-31 Slider Activation Types

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_CANCEL        | CANCEL        | XtNcancelKey        |
| OL_DEFAULTACTION | DEFAULTACTION | XtNdefaultActionKey |
| OL_HELP          | HELP          | XtNhelpKey          |
| OL_MOVEDOWN      | MOVEDOWN      | XtNdownKey          |
| OL_MOVELEFT      | MOVELEFT      | XtNleftKey          |

Table 9-31 Slider Activation Types (Continued)

| Activation Type    | Semantics     | Resource Name       |
|--------------------|---------------|---------------------|
| OL_MOVERIGHT       | MOVERIGHT     | XtNrightKey         |
| OL_MOVEUP          | MOVEUP        | XtNupKey            |
| OL_NEXTFIELD       | NEXTFIELD     | XtNnextFieldKey     |
| OL_PREVFIELD       | PREVFIELD     | XtNprevFieldKey     |
| OL_SCROLLBOTTOM    | DATA END      | XtNscrollBottomKey  |
| OL_SCROLLDOWN      | ALT+DOWN      | XtNscrollDownKey    |
| OL_SCROLLLEFT      | ALT+LEFT      | XtNscrollLeftKey    |
| OL_SCROLLLEFTEDGE  | ALT+{         | XtNscrollLeftEdge   |
| OL_SCROLLRIGHT     | ALT+RIGHT     | XtNscrollRightKey   |
| OL_SCROLLRIGHTEDGE | ALT+}         | XtNscrollRightEdge  |
| OL_SCROLLTOP       | DATA START    | XtNscrollTopKey     |
| OL_SCROLLUP        | ALT+UP        | XtNscrollUpKey      |
| OL_SELECT          | SELECT        | XtNselectBtn        |
| OL_SELECTKEY       | SELECT        | XtNselectKey        |
| OL_TOGGLEPUSHPIN   | TOGGLEPUSHPIN | XtNtogglePushpinKey |

Activation types not described in the following table are described in “Common Activation Types” on page 68.

**OL\_SCROLLBOTTOM**

For a scrollbar with `XtNOrientation` of `OL_VERTICAL`, this activation type will move the slider to the value of `XtNsliderMin` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

**OL\_SCROLLDOWN/  
OL\_SCROLLLEFT**

These activation types will move the slider one negative unit of granularity and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

**OL\_SCROLLLEFTEDGE**

For a slider with `XtNOrientation` of `OL_HORIZONTAL`, this activation type will move the slider to the value of `XtNsliderMin` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.



***OL\_SCROLLRIGHTEDGE***

For a slider with `XtNorientation` of `OL_HORIZONTAL`, this activation type will move the slider to the value of `XtNsliderMax` minus the value of `XtNproportionLength` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLTOP***

For a slider with `XtNorientation` of `OL_VERTICAL`, this activation type will move the slider to the value of `XtNsliderMax` and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SCROLLUP/  
OL\_SCROLLRIGHT***

These activation types will move the slider one positive unit of granularity and call the `XtNsliderMoved` callback with the appropriate `OlScrollbarVerify` structure.

***OL\_SELECT***

This activation type depends on the position of the pointer within the Slider widget. When the pointer is positioned on the drag box, the `XtNsliderMoved` callback will be called according to the value of the `XtNdragCBType`. When the pointer is positioned on the right anchor, the behavior will be the same as the `OL_SCROLLRIGHTEDGE` activation type. When the pointer is positioned on the top anchor, the behavior will be the same as the `OL_SCROLLTOP` activation type. When the pointer is positioned on the left anchor, the behavior will be the same as the `OL_SCROLLLEFTEDGE` activation type. When the pointer is positioned on the bottom anchor, the behavior will be the same as the `OL_SCROLLBOTTOM` activation type. When the pointer is positioned on the bar above or to the right of the drag box, the behavior will be the same as the `OL_SCROLLUP` and `OL_SCROLLRIGHT` activation type. When the pointer is positioned on the below or to the left of the drag box, the behavior will be the same as the `OL_SCROLLLEFT` and `OL_SCROLLDOWN` activation type.

***See Also***

“Gauge Widget” on page 395,  
“Scrollbar Widget” on page 508.

---

## *StaticText Widget*

### *StaticText Widget*

#### *Class*

*Class Name:*     StaticText  
*Class Pointer:*  staticTextWidgetClass

#### *Ancestry*

Core-Primitive-StaticText

#### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/StaticText.h>
```

#### *Description*

The StaticText widget provides a way to present an uneditable block of single or multi-line text to the user. The layout of the text is configured by a few simple layout controls.

##### ***Word Wrap***

If the text is too long to fit in the width provided by the StaticText widget, the text may be “wrapped” if the application requests it. The wrapping occurs at a space between words, if possible, leaving as many words on a line as will fit. If a word is too long for the width, it is wrapped between characters. An embedded newline will always cause a wrap.

##### ***Text Clipping***

If the text is not wrapped, it will be truncated if it cannot fit in the width of the StaticText widget. The application can choose whether the truncation occurs on the left, right, or evenly on both sides of each line of the text.

If the text is too large to fit in the height provided by the StaticText widget, the text is clipped on the bottom. The clipping falls on a pixel boundary, not between lines, so that it is possible that only the upper part of the last line of text may be visible.

### Space Stripping

The application can choose to have leading spaces or trailing spaces (or both) stripped from the text before display, or can choose to have no stripping done.

### Selecting and Operating on Text

The StaticText widget allows text to be selected in several ways and then copied. See “Text Selection Operations” on page 204 for the description of these operations. The application can control whether or not this selectability is allowed on a StaticText widget.

### Coloration

The diagram illustrates the resources that affect StaticText coloration.

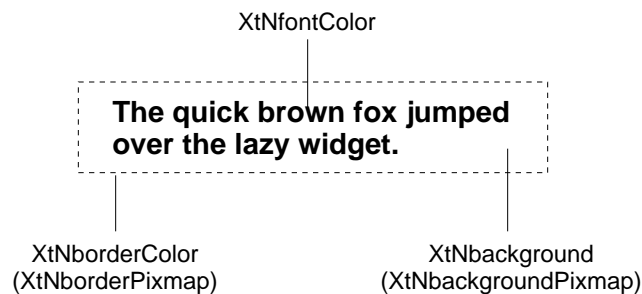


Figure 9-16 StaticText Coloration

### Keyboard Traversal

The default value of the `XtNtraversalOn` resource is FALSE.

If `XtNtraversalOn` is set to TRUE, the widget responds to the following keyboard navigation keys:

- NEXTFIELD moves to the next traversable widget in the window
- PREVIOUSFIELD moves to the previous traversable widget in the window
- NEXTWINDOW moves to the next window in the application
- PREVIOUSWINDOW moves to the previous window in the application
- NEXTAPP moves to the first window in the next application
- PREVIOUSAPP moves to the first window in the previous application

*StaticText Widget*

**Keyboard Mnemonic and Accelerator Display**

The *StaticText* does not have keyboard mnemonic or keyboard accelerator capabilities.

*Resources*

*Table 9-32 StaticText Core Resources*

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | (calculated)        | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | (calculated)        | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 9-33 StaticText Primitive Resources*

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>      | <b>Access</b> |
|--------------------|----------------|---------------------|---------------|
| XtNaccelerator     | String         | NULL                | n/a           |
| XtNacceleratorText | String         | NULL                | n/a           |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont       | SGID          |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID          |

Table 9-33 StaticText Primitive Resources (Continued)

| Name               | Type          | Default             | Access |
|--------------------|---------------|---------------------|--------|
| XtNforeground      | Pixel         | XtDefaultForeground | SGID   |
| XtNinputFocusColor | Pixel         | Red                 | SGID   |
| XtNmnemonic        | unsigned char | '\0'                | n/a    |
| XtNreferenceName   | String        | NULL                | GI     |
| XtNreferenceWidget | Widget        | NULL                | GI     |
| XtNscale           | int           | 12                  | SGI    |
| XtNtextFormat      | OlStrRep      | OL_SB_STR_REP       | GI     |
| XtNtraversalOn     | Boolean       | TRUE                | SGI    |
| XtNuserData        | XtPointer     | NULL                | SGI    |

Table 9-34 StaticText Resources

| Name             | Type      | Default       | Access |
|------------------|-----------|---------------|--------|
| XtNalignment     | OlDefine  | OL_LEFT       | SGI    |
| XtNgravity       | int       | CenterGravity | SGI    |
| XtNhSpace        | Dimension | 2             | SGI    |
| XtNlineSpace     | int       | 0             | SGI    |
| XtNrecomputeSize | Boolean   | TRUE          | SGI    |
| XtNselectable    | Boolean   | TRUE          | SGI    |
| XtNstring        | OlStr     | NULL          | SGI    |
| XtNstrip         | Boolean   | TRUE          | SGI    |
| XtNvSpace        | Dimension | 2             | SGI    |
| XtNwrap          | Boolean   | TRUE          | SGI    |

**XtNalignment**

| Class         | Type     | Default | Access |
|---------------|----------|---------|--------|
| XtCAalignment | OlDefine | OL_LEFT | SGI    |

Synopsis: The alignment of the lines to be applied when drawing the text.

Values: OL\_LEFT/"left" - The left sides of lines are vertically aligned.  
 OL\_CENTER/"center" - The centers are vertically aligned.  
 OL\_RIGHT/"right" - The right sides are vertically aligned.

*StaticText* Widget

***XtNgravity***

| Class      | Type | Default       | Access |
|------------|------|---------------|--------|
| XtCGravity | Int  | CenterGravity | SGI    |

Synopsis: How extra space is used with the *StaticText* widget. See Table 9-35.

If `XtNrecomputeSize` is set to `FALSE`, the application can set a width and height to the *StaticText* widget that exceeds the size needed to display the string. This resource controls the use of any extra space.

Table 9-35 XtNgravity Values

| Value            | Action                                                                                            |
|------------------|---------------------------------------------------------------------------------------------------|
| CenterGravity    | String is centered vertically and horizontally in the extra space.                                |
| NorthGravity     | Top edge of the string is aligned with the top edge of the space and centered horizontally.       |
| SouthGravity     | Bottom edge of the string is aligned with the bottom edge of the space and centered horizontally. |
| EastGravity      | Right edge of the string is aligned with the right edge of the space and centered vertically.     |
| WestGravity      | Left edge of the string is aligned with the left edge of the space and centered vertically.       |
| NorthWestGravity | Top and left edges of the string are aligned with the top and left edges of the space.            |
| NorthEastGravity | Top and right edges of the string are aligned with the top and right edges of the space.          |
| SouthWestGravity | Bottom and left edges of the string are aligned with the bottom and left edges of the space.      |
| SouthEastGravity | Bottom and right edges of the string are aligned with the bottom and right edges of the space.    |

***XtNheight***

| Class     | Type      | Default      | Access |
|-----------|-----------|--------------|--------|
| XtCHeight | Dimension | (calculated) | SGI    |

Synopsis: The height of the *StaticText* widget, not including the border.

If `XtNrecomputeSize` is set to `TRUE`, `XtNheight` is set to:  
 $(\text{height of text}) + 2 \times \text{XtNhSpace}$ .

***XtNhSpace/  
XtNvSpace***

| Class     | Type      | Default | Access |
|-----------|-----------|---------|--------|
| XtCHSpace | Dimension | 2       | SGI    |
| XtCVSpace | Dimension | 2       | SGI    |

Synopsis: The number of pixels of horizontal (vertical) padding between the window borders and the displayed string.

Values:  $0 \leq \text{XtNhSpace}$   
 $0 \leq \text{XtNvSpace}$

***XtNlineSpace***

| Class        | Type | Default | Access |
|--------------|------|---------|--------|
| XtCLineSpace | int  | 0       | SGI    |

Synopsis: The amount of space between lines of text.

Values:  $-100 \leq \text{XtNlineSpace}$

The spacing is specified as a percentage of the font height, and is the distance between the baseline of one text line and the top of the next font line. Thus, the distance between successive text baselines, in percentage of the font height, is  $\text{XtNlineSpace} + 100$ .

***XtNrecomputeSize***

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCRecomputeSize | Boolean | TRUE    | SGI    |

Synopsis: Whether the StaticText widget should calculate its size and automatically set the `XtNheight` and `XtNwidth` resources.

Values: TRUE/"true" - The StaticText widget will do normal size calculations that may cause its geometry to change.  
 FALSE/"false" - The StaticText widget will leave its size alone; this may cause truncation of the visible image being shown by the StaticText widget if the fixed size is too small, or may cause centering if the fixed size is too large.

***XtNselectable***

| Class         | Type    | Default | Access |
|---------------|---------|---------|--------|
| XtCSelectable | Boolean | TRUE    | SGI    |

Synopsis: Whether the user can select text in the widget.

*StaticText* Widget

Values: TRUE/"true" - The user can select text.  
 FALSE/"false" - The user cannot select text. SELECT or ADJUST mouse actions and the COPY key action will have no effect.

***XtNstring***

| Class     | Type  | Default | Access |
|-----------|-------|---------|--------|
| XtCString | OlStr | NULL    | SGI    |

Synopsis: The (NULL terminated) string to be drawn.  
 Values: Any OlStr value valid in the current locale.

***XtNstrip***

| Class    | Type    | Default | Access |
|----------|---------|---------|--------|
| XtCStrip | Boolean | TRUE    | SGI    |

Synopsis: How leading and trailing spaces are stripped during the layout of the text string.  
 Values: TRUE/"true", FALSE/"false", as shown in the following table.

| XtNstrip | XtNalignment | Spaces stripped                           |
|----------|--------------|-------------------------------------------|
| TRUE     | OL_LEFT      | Leading spaces stripped.                  |
|          | OL_RIGHT     | Trailing spaces stripped.                 |
|          | OL_CENTER    | Both leading and trailing spaces stripped |
| FALSE    | any          | None.                                     |

***XtNwidth***

| Class    | Type      | Default      | Access |
|----------|-----------|--------------|--------|
| XtCWidth | Dimension | (calculated) | SGI    |

Synopsis: The width of the StaticText widget, not including the border.

If XtNrecomputeSize is set to TRUE, XtNwidth is set to:  
 (width of text) + 2 × XtNvSpace.

***XtNwrap***

| Class   | Type    | Default | Access |
|---------|---------|---------|--------|
| XtCWrap | Boolean | TRUE    | SGI    |

Synopsis: How lines that are too long to fit in the width of the StaticText widget are wrapped.



Values: TRUE/"true", FALSE/"false", as shown in the following table. .

| <b>XtNwrap</b> | <b>XtNalignment</b> | <b>Wrap action</b>                                                                                      |
|----------------|---------------------|---------------------------------------------------------------------------------------------------------|
| FALSE          | OL_LEFT             | Clipped on the right                                                                                    |
|                | OL_RIGHT            | Clipped on the left                                                                                     |
|                | OL_CENTER           | Clipped equally on both left and right                                                                  |
| TRUE           | any                 | Long text is broken between words with each line of the displayed text having as many words as can fit. |

## *Activation Types*

The following table lists the activation types used by the StaticText.

*Table 9-36* StaticText Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_ADJUST              | ADJUST           | XtNadjustBtn         |
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_COPY                | COPY             | XtNcopyKey           |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_SELECT              | SELECT           | XtNselectBtn         |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

---

## *StaticText Widget*

### ***OL\_ADJUST***

This activation type can be used to alter a selection, as described in the *OPEN LOOK GUI Functional Specification* section “Adjusting a Text Selection” in Chapter 17.

### ***OL\_COPY***

This activation type can be used to copy the currently selected text to the clipboard, as described in the *OPEN LOOK GUI Functional Specification* section “Copying Text” in Chapter 17.

### ***OL\_SELECT***

This activation type can be used to select characters, as described in the *OPEN LOOK GUI Functional Specification* section “Selecting Text” in Chapter 17.

## *See Also*

“TextField Functions” on page 686,  
“Text Selection Operations” on page 204,  
“TextEdit Functions” on page 660,  
“TextEdit Widget” on page 623,  
“TextField Widget” on page 665.

## Stub Widget

### Class

*Class Name:* Stub  
*Class Pointer:* stubWidgetClass

### Ancestry

Core-Primitive-Stub

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/Stub.h>
```

### Description

The Stub widget is a method-driven widget that allows the application to specify procedures at creation and/or `XtSetValues()` time that are normally restricted to a widget's class part.

Most class part procedures have been attached to the instance part. For example, with the Stub widget, it is possible to set the procedure that is called whenever an exposure occurs. It is also possible to set the *setValues* and *initialize* procedures.

### Local Widgets

By allowing the application to specify procedures outside the widget class structure, applications can use the Stub widget to build local widgets without having going through the formal steps. For example, suppose an application wanted to create a menu separator widget that inherits its parent's background color at creation time. It would be wasteful to create a new widget to perform these trivial tasks. Instead, the application would use a Stub widget and specify an Initialize procedure for it.

### Graphics Applications

The Stub widget can be used to implement graphics applications. Since the application has direct access to the widget's internal expose procedure, the application can take advantage of the exposure compression provided with the *region* argument. This field is not accessible if the application used an Event Handler to trap exposures. Also, since the application has access to the SetValues and SetValuesHook procedures, the application can programmatically modify graphic-related resources of the Stub widget.

### Inheriting Procedures from Existing Widgets

Once a Stub widget is created, other Stub widgets can inherit its methods without the application having to specify them again. All the application has to do is specify a reference Stub widget in the creation Arg list and the new Stub widget will inherit all instance methods from the referenced Stub widget.

### Wrapping Widgets Around Existing Windows

The Stub widget also allows the application to give widget functionality to existing X windows. For example, if the application wanted to track button presses on the root window, the application would create a Stub widget using the RootWindow ID as the `XtNwindow` resource. Once this has been done, the application can monitor events on the root window by attaching event handlers to the Stub widget.

### Coloration

The diagram illustrates the resources used for Stub coloration.

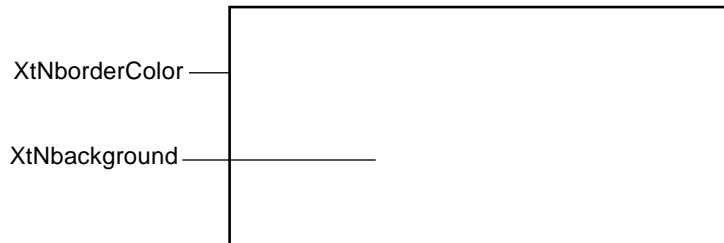


Figure 9-17 Stub Coloration

### **Keyboard Traversal**

The Stub is a Primitive widget and it inherits the translations for traversal actions from the `Primitive` class. The user of a Stub widget should add translations for dealing with the navigation events listed in the section of “TextField Widget” on page 665 that apply to the particular use of the Stub.

### **Keyboard Mnemonic Display**

The Stub does not display the mnemonic accelerator. If the Stub is the child of a Caption widget, the Caption widget can be used to display the Stub’s mnemonic.

### **Keyboard Accelerator Display**

The Stub does not display the keyboard accelerator. If the Stub is the child of a Caption widget, the Caption widget can be used to display the Stub’s accelerator as part of the label.

### **Coloration**

The Stub widget should be able to display a state that indicates it has input focus. The general heuristic used for this display by OPEN LOOK widgets is that the background color is replaced with the input focus color found in the resource `XtNinputFocusColor`.

## **Resources**

Table 9-37 Stub Core Resources

| <b>Name</b>                       | <b>Type</b>                   | <b>Default</b>                   | <b>Access</b> |
|-----------------------------------|-------------------------------|----------------------------------|---------------|
| <code>XtNaccelerators</code>      | <code>AcceleratorTable</code> | NULL                             | SGI           |
| <code>XtNancestorSensitive</code> | Boolean                       | TRUE                             | G             |
| <code>XtNbackground</code>        | Pixel                         | <code>XtDefaultBackground</code> | SGID          |
| <code>XtNbackgroundPixmap</code>  | Pixmap                        | <code>XtUnspecifiedPixmap</code> | SGI           |
| <code>XtNborderColor</code>       | Pixel                         | <code>XtDefaultForeground</code> | SGID          |
| <code>XtNborderPixmap</code>      | Pixmap                        | <code>XtUnspecifiedPixmap</code> | SGI           |
| <code>XtNborderWidth</code>       | Dimension                     | 1                                | SGI           |
| <code>XtNcolormap</code>          | Colormap                      | (parent’s)                       | SGI           |
| <code>XtNdepth</code>             | int                           | (parent’s)                       | GI            |

Stub Widget

Table 9-37 Stub Core Resources (Continued)

| Name                 | Type           | Default    | Access |
|----------------------|----------------|------------|--------|
| XtNdestroyCallback   | XtCallbackList | NULL       | SGIO   |
| XtNheight            | Dimension      | 0          | SGI    |
| XtNmappedWhenManaged | Boolean        | TRUE       | SGI    |
| XtNscreen            | Screen *       | (parent's) | G      |
| XtNsensitive         | Boolean        | TRUE       | GIO    |
| XtNtranslations      | XtTranslations | NULL       | SGI    |
| XtNwidth             | Dimension      | 0          | SGI    |
| XtNx                 | Position       | 0          | SGI    |
| XtNy                 | Position       | 0          | SGI    |

Table 9-38 Stub Primitive Resources

| Name               | Type           | Default             | Access |
|--------------------|----------------|---------------------|--------|
| XtNaccelerator     | String         | NULL                | SGI    |
| XtNacceleratorText | String         | NULL                | SGI    |
| XtNconsumeEvent    | XtCallbackList | NULL                | SGIO   |
| XtNfont            | OlFont         | XtDefaultFont       | SGID   |
| XtNfontColor       | Pixel          | XtDefaultForeground | SGID   |
| XtNforeground      | Pixel          | XtDefaultForeground | SGID   |
| XtNinputFocusColor | Pixel          | Red                 | SGID   |
| XtNmnemonic        | unsigned char  | '\0'                | SGI    |
| XtNreferenceName   | String         | NULL                | GI     |
| XtNreferenceWidget | Widget         | NULL                | GI     |
| XtNscale           | int            | 12                  | SGI    |
| XtNtextFormat      | OlStrRep       | OL_SB_STR_REP       | GI     |
| XtNtraversalOn     | Boolean        | FALSE               | SGI    |
| XtNuserData        | XtPointer      | NULL                | SGI    |

Table 9-39 Stub Resources

| Name               | Type     | Default | Access |
|--------------------|----------|---------|--------|
| XtNacceptFocusFunc | Function | NULL    | SGI    |
| XtNactivateFunc    | Function | NULL    | SGI    |
| XtNdestroy         | Function | NULL    | SGI    |

Table 9-39 Stub Resources (Continued)

| Name                    | Type     | Default      | Access |
|-------------------------|----------|--------------|--------|
| XtNexpose               | Function | NULL         | SGI    |
| XtNgetValuesHook        | Function | NULL         | SGI    |
| XtNhighlightHandlerProc | Function | NULL         | SGI    |
| XtNinitialize           | Function | (private)    | GI     |
| XtNinitializeHook       | Function | NULL         | GI     |
| XtNqueryGeometry        | Function | NULL         | SGI    |
| XtNrealize              | Function | (private)    | SGI    |
| XtNreferenceStub        | Widget   | NULL         | GI     |
| XtNregisterFocusFunc    | Function | NULL         | SGI    |
| XtNresize               | Function | NULL         | SGI    |
| XtNsetValues            | Function | NULL         | SGI    |
| XtNsetValuesAlmost      | Function | (superclass) | SGI    |
| XtNsetValuesHook        | Function | NULL         | SGI    |
| XtNtraversalHandlerFunc | Function | NULL         | SGI    |
| XtNwindow               | Window   | NULL         | GI     |

***XtNacceptFocusFunc***

| Class              | Type     | Default | Access |
|--------------------|----------|---------|--------|
| XtCAcceptFocusFunc | Function | NULL    | SGI    |

Synopsis: The function that allows applications to override the default accept focus procedure.

This procedure has the same semantics as the `XtAcceptFocusFunc` Core Widget Class procedure and it is called by the Stub Widget Class's accept focus class procedure. When overriding the default accept focus procedure, the routine `OlCanAcceptFocus()` can be used to check the widget's focus-taking eligibility. `OlSetInputFocus()` (see page 151) should be used instead of `XSetInputFocus()` when explicitly setting focus to a window.

***XtNactivateFunc***

| Class           | Type     | Default | Access |
|-----------------|----------|---------|--------|
| XtCActivateFunc | Function | NULL    | SGI    |

Synopsis: The procedure called whenever `OlActivateWidget()` is called with the Stub widget ID for which this function was assigned.

*Stub Widget*

The procedure has the following declaration:

```
Boolean activateFunc(
 Widget w,
 OlVirtualName activation_type,
 XtPointer data);
```

If the *activation\_type* is valid, the routine should take the appropriate action and return TRUE; otherwise, the routine should return FALSE.

***XtNdestroy***

| Class      | Type     | Default | Access |
|------------|----------|---------|--------|
| XtCDestroy | Function | NULL    | SGI    |

Synopsis: The procedure called when this Stub instance is destroyed.

The procedure has the following declaration:

```
void destroy(Widget w);
```

***XtNexpose***

| Class     | Type     | Default | Access |
|-----------|----------|---------|--------|
| XtCExpose | Function | NULL    | SGI    |

Synopsis: The procedure called when the corresponding Stub widget receives an exposure event.

The procedure has the following declaration:

```
void expose(
 Widget w,
 XEvent *xevent,
 Region region);
```

Since the Stub widget class has requested exposure compression, the region field is valid.

***XtNgetValuesHook***

| Class            | Type     | Default | Access |
|------------------|----------|---------|--------|
| XtCGetValuesHook | Function | NULL    | SGI    |

Synopsis: The procedure called whenever the application makes an *XtGetValues()* call on the corresponding Stub widget.



The procedure has the following declaration:

```
void getValuesHook(
 Widget w,
 ArgList args,
 Cardinal *num_args);
```

### ***XtNheight***

| Class     | Type      | Default | Access |
|-----------|-----------|---------|--------|
| XtCHeight | Dimension | 0       | SGI    |

Synopsis: The height of the widget.

If *XtNwindow* has a NULL value, the application must ensure that the dimensions of *XtNwidth* and *XtNheight* are non-NULL. The application can specify the width and height with an Arg list or specify an *initialize* procedure that sets them with non-NULL values. If either of these dimensions is NULL when the application attempts to realize the Stub widget, an error will result.

### ***XtNhighlightHandlerProc***

| Class                   | Type     | Default | Access |
|-------------------------|----------|---------|--------|
| XtCHighlightHandlerProc | Function | NULL    | SGI    |

Synopsis: Whether the Stub widget displays a visual indication that it has, or does not have, input focus.

The *XtNhighlightHandlerProc* procedure will be called to notify a Stub widget that input focus has either moved to or from the widget. The *XtNhighlightHandlerProc* will be called after the widget has lost or accepted focus as a result of calling *OlSetInputFocus()*, calling *OlMoveFocus()*, or matching a KeyPress event to a keyboard traversal command (see “*OlMoveFocus*” on page 152). The procedure has the following declaration:

```
void highlightHandlerProc(
 Widget widget,
 OlDefine highlight_type);
```

This procedure has the responsibility of displaying the object in the appropriate state based on the *highlight\_type*. Whenever a Stub widget receives focus, its *XtNhighlightHandlerProc* will be called with a *highlight\_type* of *OL\_IN*. An *OL\_OUT* *highlight\_type* is used to signal that the Stub widget has lost focus.

***XtNinitialize***

| Class         | Type     | Default   | Access |
|---------------|----------|-----------|--------|
| XtCInitialize | Function | (private) | SGI    |

Synopsis: The procedure called by `XtCreateWidget()` for a Stub widget instance.

The procedure has the following declaration:

```
void initialize(
 Widget request,
 Widget new,
 ArgList args,
 Cardinal *num_args);
```

If the application supplies its own initialize procedure, it is the application's responsibility to deal with the `XtNwindow` resource. When the `XtNwindow` resource is non-NULL, the default initialize procedure fills in the Stub widget's `XtNx`, `XtNy`, `XtNwidth`, and `XtNheight` resource values with the corresponding window attribute values obtained from the window specified by the `XtNwindow` resource.

***XtNinitializeHook***

| Class             | Type     | Default | Access |
|-------------------|----------|---------|--------|
| XtCInitializeHook | Function | NULL    | SGI    |

Synopsis: The procedure called by `XtCreateWidget()` for a Stub widget instance after the *initialize* procedure has been called.

The procedure has the following declaration:

```
void initializeHook(
 Widget w,
 ArgList args,
 Cardinal *num_args);
```

The application can access the creation argument list through this routine. The widget specified with the `w` argument is the *new* widget from the *initialize* procedure.

**XtNqueryGeometry**

| Class            | Type     | Default | Access |
|------------------|----------|---------|--------|
| XtCQueryGeometry | Function | NULL    | SGI    |

Synopsis: The procedure called whenever the application does an `XtQueryGeometry()` request on the corresponding Stub widget.

The procedure has the following declaration:

```
void queryGeometry(
 Widget w,
 XtWidgetGeometry *request,
 XtWidgetGeometry *preferred_return);
```

**XtNrealize**

| Class      | Type     | Default   | Access |
|------------|----------|-----------|--------|
| XtCRealize | Function | (private) | SGI    |

Synopsis: The procedure called to realize a Stub widget instance.

The procedure has the following declaration:

```
void realize(
 Widget w,
 XtValueMask *value_mask,
 XSetWindowAttributes *attributes);
```

If the application supplies its own realize procedure, it is the application's responsibility to deal with the `XtNwindow` resource. When `XtNwindow` is non-NULL, the realize procedure uses this window for the widget instance instead of creating a new window. The default realize procedure gives an error message if another widget in its process space is referencing the window already; it does not reparent the specified window.

**XtNreferenceStub**

| Class            | Type   | Default | Access |
|------------------|--------|---------|--------|
| XtCReferenceStub | Widget | NULL    | GI     |

Synopsis: The pointer to an existing Stub widget.

If this pointer is non-NULL, the new Stub will inherit all instance methods from the referenced Stub widget. An `XtSetValues()` request on the new Stub widget should be used to change any inherited methods.

***XtNregisterFocusFunc***

| Class                | Type     | Default | Access |
|----------------------|----------|---------|--------|
| XtCRegisterFocusFunc | Function | NULL    | SGI    |

Synopsis: The procedure called whenever a Stub widget gains focus.

The procedure has the following declaration:

```
Widget registerFocus(Widget stub_widget);
```

Whenever a Stub widget gains focus, this procedure is called and the Stub's shell sets the "current focus widget" (see "OIGetCurrentFocusWidget" on page 151) to the value returned by it. If this function is NULL or returns NULL, the current focus widget is set to the Stub widget. This is the typical case. If this procedure returns a widget ID other than the Stub widget's, that ID is used to update the current focus widget so that a subsequent call to `OIGetCurrentFocusWidget()` would return it. Returning a widget ID other than the Stub widget's will not move the focus away from the Stub widget.

***XtNresize***

| Class     | Type     | Default | Access |
|-----------|----------|---------|--------|
| XtCResize | Function | NULL    | SGI    |

Synopsis: The procedure called whenever a Stub widget instance is resized.

The procedure has the following declaration:

```
void resize(Widget w);
```

***XtNsetValues***

| Class        | Type     | Default | Access |
|--------------|----------|---------|--------|
| XtCSetValues | Function | NULL    | SGI    |

Synopsis: The procedure called whenever the application makes an `XtSetValues()` call on a Stub widget instance.

The procedure has the following declaration:

```
Boolean setValues(
 Widget current,
 Widget request,
 Widget new,
 ArgList args,
 Cardinal *num_args);
```

***XtNsetValuesAlmost***

| Class              | Type     | Default      | Access |
|--------------------|----------|--------------|--------|
| XtCSetValuesAlmost | Function | (superclass) | SGI    |

Synopsis: The procedure called when a parent rejects the requested geometry.

The procedure has the following declaration:

```
void setValuesAlmost(
 Widget w,
 Widget new_widget,
 XtWidgetGeometry *request,
 XtWidgetGeometry *reply);
```

This procedure is called when the application attempts to set a Stub widget's geometry via an `XtSetValues()` call and the Stub widget's parent did not accept the requested geometry. The default `setValuesAlmost` procedure simply accepts the suggested compromise.

***XtNsetValuesHook***

| Class            | Type     | Default | Access |
|------------------|----------|---------|--------|
| XtCSetValuesHook | Function | NULL    | SGI    |

Synopsis: The procedure called whenever the application makes an `XtSetValues()` call on a Stub widget instance.

The procedure has the following declaration:

```
Boolean setValuesHook(
 Widget w,
 ArgList args,
 Cardinal *num_args);
```

Since this procedure is called after the `setValues` procedure, the widget specified by the `w` argument is the *new* widget from the `setValues` procedure.

***XtNtraversalHandlerFunc***

| Class                   | Type     | Default | Access |
|-------------------------|----------|---------|--------|
| XtCTraversalHandlerFunc | Function | NULL    | SGI    |

Synopsis: The procedure called to process traversal commands whenever the Stub widget has focus.

If an application wants the Stub widget to process traversal commands whenever the Stub widget has focus, this resource is used to supply the

*Stub Widget*

traversal routine. An example of a case when this is desirable is when a Stub widget is used to implement a spreadsheet. In this case, the Stub widget would trap the OL\_MOVERIGHT, OL\_MOVELEFT, etc. commands to move focus between the cells in the spreadsheet. The traversal handling routine has the following declaration:

```
Widget traversalHandler(
 Widget w,
 Widget start,
 OlVirtualName direction,
 Time time);
```

If the traversal routine can process the traversal command, it returns the ID of the widget that now has focus. The widget ID returned can be the Stub widget's ID. This is the case when the traversal command was processed, but focus did not leave the Stub widget. If the traversal routine cannot process the given command, it should return NULL.

***XtNwidth***

| Class    | Type      | Default | Access |
|----------|-----------|---------|--------|
| XtCWidth | Dimension | 0       | SGI    |

Synopsis: The width of the widget.

If *XtNwindow* has a NULL value, the application must ensure that the dimensions of *XtNwidth* and *XtNheight* are non-NULL. The application can specify the width and height with an Arg list or specify an *initialize* procedure that sets them with non-NULL values. If either of these dimensions is NULL when the application attempts to realize the Stub widget, an error will result.

***XtNwindow***

| Class     | Type   | Default | Access |
|-----------|--------|---------|--------|
| XtCWindow | Window | NULL    | GI     |

Synopsis: The window ID that the Stub widget should associate with its instance data at realization time.

The *XtNwindow* resource can be specified at initialization time only. If a window ID is supplied, that Stub widget instance will trap events on the given window. After the Stub widget instance is realized, the function *XtWindow()* will return this window ID. If the Stub widget is managed by its parent widget, the supplied window will be included in geometry calculations even though the Stub widget (by default) does not reparent the supplied window to be a

child of the parent widget's window. Explicit calls to `XtMoveWidget()`, `XtResizeWidget()`, `XtConfigureWidget()`, or `XtSetValues()` can be used to modify the window's attributes.

---

**Note** – When the Stub widget instance is destroyed, the window will be destroyed along with it.

---

## *Activation Types*

The following table lists the activation types used by the Stub.

*Table 9-40* Stub Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |

The Stub widget has no activation types besides the ones in “Common Activation Types” on page 68.

*Example*

The following example illustrates how an application can use the Stub widget to perform a particular type of exposure handling. Since an *initialize* procedure was not specified and the *XtNwindow* resource was not used, the initial Arg list includes non-NULL values for the widget's width and height.

```
static void Redisplay(
 Widget w;
 XEvent *xevent;
 Region region);
{
 /* do something interesting here */
} /* END OF Redisplay() */

main(...)
{
 Widget base;
 Widget stub;
 static Arg args[] = {
 { XtNexpose, (XtArgVal) Redisplay },
 { XtNwidth, (XtArgVal) 1 },
 { XtNheight, (XtArgVal) 1 }
 };

 OlToolkitInitialize();
 base = XtAppInitialize(...);
 stub = XtCreateManagedWidget("graphics pane", stubWidgetClass,
 base, args, XtNumber(args));
 ...
} /* END of main() */
```

*See Also*

“Input Focus Functions” on page 150.



### *TextEdit Widget*

#### *Class*

*Class Name:*     TextEdit  
*Class Pointer:*  textEditWidgetClass

#### *Ancestry*

Core-Primitive-TextEdit

#### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/TextEdit.h>
```

#### *Description*

The TextEdit widget provides a multi-line text editing facility that has both a customizable user interface and a programmatic interface. It provides a consistent editing paradigm for textual data. An example is shown in the following figure.

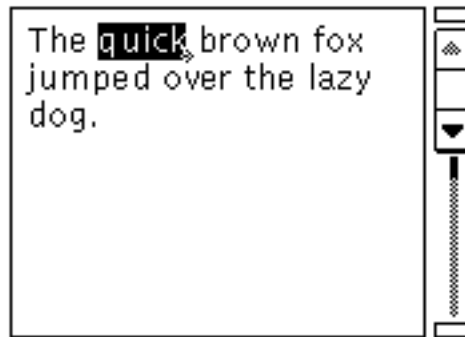


Figure 10-1 Simple TextEdit Widget

The TextEdit widget provides three text wrap modes: `OL_WRAP_OFF`, `OL_WRAP_ANY`, and `OL_WRAP_WHITE_SPACE`. The TextEdit widget manages its text data using a `TextBuffer` data structure and allows the application to access and manipulate its `TextBuffer` in order to implement more complex textual operations.

The TextEdit widget provides several distinct callback lists used to monitor the state of the textual data: insertion cursor movement, modification of the text, and post modification notification. Each of these callbacks provide information to the application regarding the intended action. The application can simply examine this information to maintain its current state or can disallow the action and perform any of the programmatic manipulations instead.

The TextEdit widget provides distinct callback lists for user input: mouse button down and key press. The `call_data` for these callbacks decodes the input for the application. The application can examine the input and either consume the action, and perform any of the programmatic manipulations, or allow the widget to act upon it.

The TextEdit widget also provides the application with a callback list invoked when the widget is redisplayed. With this callback the application can add callbacks that can be used to display information in the margins of the TextEdit, such as line numbers or update marks. (See “`XtNmargin`” on page 636.)

### *Editing Capabilities*

The TextEdit widget provides editing capabilities to move the insert point, select text, delete text, scroll the display, perform cut, copy, paste, and undo operations, and refresh the text display. All of these capabilities are represented by OLIT Activation Types, which are mapped to key bindings via global resources stored in the X server. All of these settings dynamically change immediately after new resource values are stored in the server. See Table 10-4 on page 642 for a complete list of these Activation Types supported by the TextEdit widget.

### *Text Hierarchy*

Text is considered to be hierarchically composed of white space, words, lines, and paragraphs. These terms are defined as:

|                          |                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <i>white space</i>       | Any non-empty sequence of the ASCII characters space, tab, linefeed, or carriage return (decimal values of 32, 9, 10, 13, respectively) |
| <i>word</i>              | Any non-empty sequence of characters bounded on both sides by white space.                                                              |
| <i>source line</i>       | Any (possibly empty) sequence of characters bounded by newline characters.                                                              |
| <i>display line</i>      | Any (possibly empty) sequence of characters appearing on a single window display line.                                                  |
| <i>source paragraph</i>  | Any sequence of characters bounded by sets of two or more adjacent newline characters.                                                  |
| <i>display paragraph</i> | Any (possibly empty) sequence of characters bounded by newline characters. (This is a synonym for <i>source line</i> .)                 |

In all cases, the beginning or end of the edit text is an acceptable bounding element in the previous definitions.

### *The Text Buffer*

The TextEdit widget uses a *TextBuffer* to store and manage the text data. If the application needs to implement complex operations on the text (i.e., search for a string, replace a paragraph, etc.), it can access the handle to the TextEdit's TextBuffer and then use the set of TextBuffer routines to manipulate the TextBuffer. Since the TextBuffer provides more detailed information on the

structure of the text (words, lines, pages, etc.) than the TextEdit widget itself, the application can use the TextBuffer to have more programmatic control over the text. There are two variants of the TextBuffer that are used, depending on the value of the widget's `XtNtextFormat` resource. See “Text Buffer Functions” on page 163 and “Text Buffer Functions for Internationalization” on page 176 for information on the TextBuffer.

### *Sizing the Display*

When making display decisions, the TextEdit widget first will use either the application-specified width and height or, if these values are not specified, calculate width and height by applying the values of the `XtNcharsVisible` and `XtNlinesVisible` resources. Once the width and height are determined, the TextEdit widget will request an appropriate size from its parent (considering the margins). If the request is denied or only partially satisfied, no future growth requests will be made unless there is an intervening resize operation externally imposed.

Once the size of the display is settled, the TextEdit widget calculates the display lines based on this size, the various margins, the font, tab table, and wrap mode.

### *Wrapping*

If the wrap mode (see “`XtNwrapMode`” on page 642) is `OL_WRAP_ANY`, as many characters from the source line as will entirely fit before the right margin are written to the current display line, then the next character starts at the left margin of the next display line, and so on.

If the wrap mode is `OL_WRAP_WHITE_SPACE`, the line wrap occurs at the first whitespace character that follows the last full word that does fit on the current display line. If the first full word that does not fit is the first word on the display line, however, the wrap is made as if `OL_WRAP_ANY` were selected.

If the wrap mode is `OL_WRAP_OFF`, the lines are not wrapped but are clipped at the right margin. In this mode the text is horizontally scrollable.

### Text Scrolling

The application can get horizontal and/or vertical scrolling of a TextEdit by creating it as a child of a ScrolledWindow widget; see “ScrolledWindow Widget” on page 529. The ScrolledWindow will completely manage the scrolling of text for the application by using its scrollbars.

The proportion indicators on the scrollbars show relatively how much of the text is currently in the display.

As the user enters text, the view automatically scrolls when the insert point moves beyond a margin boundary (right or bottom) to keep the insert point in view.

### Coloration

The following diagram illustrates the resources that affect TextEdit coloration.

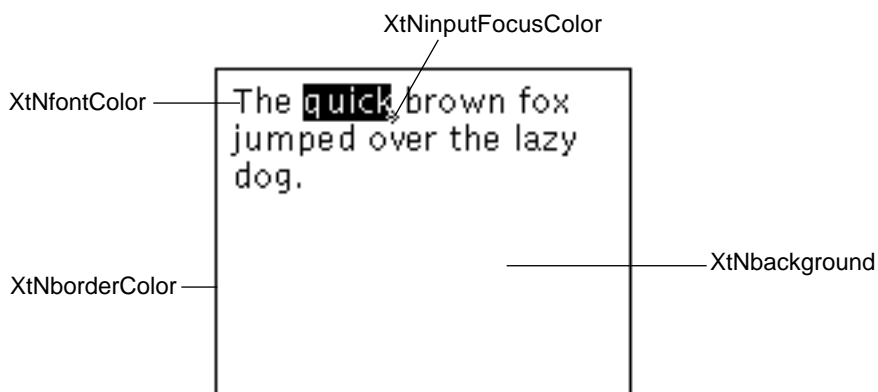


Figure 10-2 TextEdit Coloration

### Keyboard Traversal

The default value of the `XtNtraversalOn` resource is TRUE.

The TextEdit widget responds to the following Activation Types for keyboard navigation. For more information on these Activation Types and their key bindings, see Table 10-4 on page 642.

*TextEdit Widget*

|             |              |            |
|-------------|--------------|------------|
| OL_CHARBAK  | OL_LINESTART | OL_ROWDOWN |
| OL_CHARFWD  | OL_NEXTFIELD | OL_ROWUP   |
| OL_DOCEND   | OL_PANEEND   | OL_WORDBAK |
| OL_DOCSTART | OL_PANESTART | OL_WORDFWD |
| OL_LINEEND  | OL_PREVFIELD |            |

It is expected that the user will use the alternate bindings for NEXTFIELD and PREVFIELD because the primary binding, <Tab> and <Shift-Tab>, are valid characters in a Text pane (unless `XtNinsertTabs` is set to FALSE).

The TextEdit widget responds to the following Activation Types for selection. For more information on these Activation Types and their key bindings, see Table 10-4 on page 642.

|                |               |               |
|----------------|---------------|---------------|
| OL_SELCHARBAK  | OL_SELLINE    | OL_SELWORDBAK |
| OL_SELCHARFWD  | OL_SELLINEBAK | OL_SELWORDFWD |
| OL_SELFLIPENDS | OL_SELLINEFWD |               |

The TextEdit widget responds to the following Activation Types for scrolling. For more information on these Activation Types and their key bindings, see Table 10-4 on page 642.

|              |                   |                    |
|--------------|-------------------|--------------------|
| OL_PAGEDOWN  | OL_SCROLLBOTTOM   | OL_SCROLLRIGHT     |
| OL_PAGELEFT  | OL_SCROLLDOWN     | OL_SCROLLRIGHTEDGE |
| OL_PAGERIGHT | OL_SCROLLLEFT     | OL_SCROLLTOP       |
| OL_PAGEUP    | OL_SCROLLLEFTEDGE | OL_SCROLLUP        |

The TextEdit widget responds to the following Activation Types for editing. For more information on these Activation Types and their key bindings, see Table 10-4 on page 642.

|               |               |               |
|---------------|---------------|---------------|
| OL_DELCHARBAK | OL_DELLINEBAK | OL_DELWORDFWD |
| OL_DELCHARFWD | OL_DELLINEFWD | OL_UNDO       |
| OL_DELLINE    | OL_DELWORDBAK |               |

***Keyboard Mnemonic Display***

The TextEdit does not display the mnemonic. If the TextEdit widget is the child of a Caption widget, the Caption widget can be used to display the mnemonic.

***Keyboard Accelerator Display***

The TextEdit does not display the keyboard accelerator. If the TextEdit is the child of a Caption widget, the Caption widget can be used to display the accelerator as part of the label.

*Resources**Table 10-1* TextEdit Core Resources

| <b>Name</b>          | <b>Type</b>      | <b>Default</b>      | <b>Access</b> |
|----------------------|------------------|---------------------|---------------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI           |
| XtNancestorSensitive | Boolean          | TRUE                | G             |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID          |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID          |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI           |
| XtNborderWidth       | Dimension        | 1                   | SGI           |
| XtNcolormap          | Colormap         | (parent's)          | SGI           |
| XtNdepth             | Int              | (parent's)          | GI            |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO          |
| XtNheight            | Dimension        | (calculated)        | SGI           |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI           |
| XtNscreen            | Screen *         | (parent's)          | G             |
| XtNsensitive         | Boolean          | TRUE                | GIO           |
| XtNtranslations      | XtTranslations   | NULL                | SGI           |
| XtNwidth             | Dimension        | (calculated)        | SGI           |
| XtNx                 | Position         | 0                   | SGI           |
| XtNy                 | Position         | 0                   | SGI           |

*Table 10-2* TextEdit Primitive Resources

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>            | <b>Access</b> |
|--------------------|----------------|---------------------------|---------------|
| XtNaccelerator     | String         | NULL                      | SGI           |
| XtNacceleratorText | String         | NULL                      | n/a           |
| XtNconsumeEvent    | XtCallbackList | NULL                      | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont             | SGID          |
| XtNfontColor       | Pixel          | XtDefaultForeground       | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground       | SGID          |
| XtNinputFocusColor | Pixel          | (calculated; see page 27) | SGID          |
| XtNmnemonic        | Unsigned char  | '\0'                      | n/a           |
| XtNreferenceName   | String         | NULL                      | GI            |

*TextEdit Widget*

*Table 10-2* TextEdit Primitive Resources (Continued)

| <b>Name</b>        | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------------|-------------|----------------|---------------|
| XtNreferenceWidget | Widget      | NULL           | GI            |
| XtNscale           | Int         | 12             | SGI           |
| XtNtextFormat      | OlStrRep    | OL_SB_STR_REP  | GI            |
| XtNtraversalOn     | Boolean     | TRUE           | SGI           |
| XtNuserData        | XtPointer   | NULL           | SGI           |

*Table 10-3* TextEdit Resources

| <b>Name</b>           | <b>Type</b>      | <b>Default</b> | <b>Access</b> |
|-----------------------|------------------|----------------|---------------|
| XtNblinkRate          | long             | 666            | SGI           |
| XtNbottomMargin       | Dimension        | 4              | SGI           |
| XtNbuttons            | XtCallbackList   | NULL           | SGIO          |
| XtNcharsVisible       | Int              | 50             | GI            |
| XtNcopyLabel          | OlStr            | “Copy”         | GI            |
| XtNcopyMnemonic       | unsigned char    | ‘\0’           | GI            |
| XtNcursorPosition     | TextPosition     | 0              | SGI           |
| XtNcutLabel           | OlStr            | “Cut”          | GI            |
| XtNcutMnemonic        | Unsigned char    | ‘\0’           | GI            |
| XtNdeleteLabel        | OlStr            | “Delete”       | GI            |
| XtNdeleteMnemonic     | unsigned char    | ‘\0’           | GI            |
| XtNdisplayPosition    | TextPosition     | 0              | SGI           |
| XtNeditType           | OlEditMode       | OL_TEXT_EDIT   | SGI           |
| XtNgrowMode           | OlDefine         | OL_GROW_OFF    | SGI           |
| XtNimPreeditStyle     | OlImPreeditStyle | OL_NO_PREEDIT  | GI            |
| XtNinsertTab          | Boolean          | TRUE           | SGI           |
| XtNkeys               | XtCallbackList   | NULL           | SGIO          |
| XtNleftMargin         | Dimension        | 4              | SGI           |
| XtNlinesVisible       | Int              | 16             | GI            |
| XtNmargin             | XtCallbackList   | NULL           | SGIO          |
| XtNmenuTitle          | OlStr            | “Edit”         | GI            |
| XtNmodifyVerification | XtCallbackList   | NULL           | SGIO          |
| XtNmotionVerification | XtCallbackList   | NULL           | SGIO          |
| XtNpasteLabel         | OlStr            | “Paste”        | GI            |
| XtNpasteMnemonic      | Unsigned char    | ‘\0’           | GI            |



Table 10-3 TextEdit Resources (Continued)

| Name                      | Type           | Default             | Access |
|---------------------------|----------------|---------------------|--------|
| XtNpostModifyNotification | XtCallbackList | NULL                | SGIO   |
| XtNrightMargin            | Dimension      | 4                   | SGI    |
| XtNselectEnd              | TextPosition   | 0                   | SGI    |
| XtNselectStart            | TextPosition   | 0                   | SGI    |
| XtNsource                 | OlStr          | NULL                | SGI    |
| XtNsourceType             | OlSourceType   | OL_STRING_SOURCE    | SGI    |
| XtNtabTable               | TabTable       | NULL                | SGI    |
| XtNtopMargin              | Dimension      | 4                   | SGI    |
| XtNundoLabel              | OlStr          | "Undo"              | GI     |
| XtNundoMnemonic           | Unsigned char  | '\0'                | GI     |
| XtNwrapMode               | OlWrapMode     | OL_WRAP_WHITE_SPACE | SGI    |

***XtNblinkRate***

| Class        | Type | Default | Access |
|--------------|------|---------|--------|
| XtCblinkRate | Long | 666     | SGI    |

Synopsis: The rate that the active input caret blinks.

Values: The number of milliseconds between blinks. Setting this value to zero turns off the blink effect.

***XtNbottomMargin***

| Class     | Type      | Default | Access |
|-----------|-----------|---------|--------|
| XtCmargin | Dimension | 4       | SGI    |

Synopsis: The number of pixels used for the height of the bottom margin.

***XtNbuttons***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCcallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when a mouse button press is made in the pane.

*TextEdit Widget*

The *call\_data* parameter is a pointer to an `OlInputCallData` structure:

```
typedef struct {
 Boolean consumed;
 XEvent *event;
 KeySym *keysym;
 char **buffer;
 int *length;
 OlInputEvent ol_event;
} OlInputCallData, *OlInputCallDataPointer;

typedef enum {
 OL_UNKNOWN_INPUT, OL_SELECT, OL_ADJUST, OL_MENU,
 OL_CONSTRRAIN, OL_DUPLICATE, OL_PAN, OL_UNKNOWN_KEY_INPUT,
 OL_CUT, OL_COPY, OL_PASTE, OL_HELP, OL_CANCEL, OL_PROP,
 OL_STOP, OL_UNDO, OL_NEXT_FIELD, OL_PREV_FIELD, OL_CHARFWD,
 OL_CHARBAK, OL_ROWDOWN, OL_ROWUP, OL_WORDFWD, OL_WORDBAK,
 OL_LINESTART, OL_LINEEND, OL_DOCSTART, OL_DOCEND, OL_PANESTART,
 OL_PANEEND, OL_DELCHARFWD, OL_DELCHARBAK, OL_DELWORDFWD,
 OL_DELWORDBAK, OL_DELLINEFWD, OL_DELLINEBAK, OL_DELLINE,
 OL_SELCHARFWD, OL_SELCHARBAK, OL_SELWORDFWD, OL_SELWORDBAK,
 OL_SELLINEFWD, OL_SELLINEBAK, OL_SELLINE, OL_SELFIPENDS,
 OL_REDRAW, OL_RETURN, OL_PAGEUP, OL_PAGEDOWN, OL_HOME, OL_END,
 OL_SCROLLUP, OL_SCROLLDOWN, OL_SCROLLLEFT, OL_SCROLLRIGHT,
 OL_SCROLLLEFTEDGE, OL_SCROLLRIGHTEDGE, OL_PGM_GOTO
} OlInputEvent;
```

This callback can be used to remap or consume button events; for example, to override the default Edit menu. To do so, the application would add a `XtNbuttons` callback that checks the *ol\_event* member of the `OlInputCallData` structure. If the value of the *ol\_event* member is `OL_MENU`, the callback posts the application-defined menu and sets the *consumed* member of the `OlInputCallData` structure to `TRUE`.

***XtNcharsVisible***

| Class                        | Type             | Default         | Access          |
|------------------------------|------------------|-----------------|-----------------|
| <code>XtCCharsVisible</code> | <code>int</code> | <code>50</code> | <code>GI</code> |

Synopsis: The initial width of the text in terms of characters.

Values: `0 ≤ XtNcharsVisible`

This resource overrides the `XtNwidth` resource setting. The `XtNwidth` is recalculated to be the value of `XtNcharsVisible` multiplied by the width of the “n” (en) character in the font plus the values for the left and right margins.

*TextEdit Widget*

The value of this resource changes to reflect the effects of geometry changes imposed by the widget tree and the user. Calls to `XtSetValues()` for this resource are ignored.

***XtNcopyLabel/  
XtNcutLabel/  
XtNdeleteLabel/  
XtNpasteLabel/  
XtNundoLabel***

| <b>Class</b>   | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|----------------|-------------|----------------|---------------|
| XtCCopyLabel   | OlStr       | "Copy"         | GI            |
| XtCCutLabel    | OlStr       | "Cut"          | GI            |
| XtCDeleteLabel | OlStr       | "Delete"       | GI            |
| XtCPasteLabel  | OlStr       | "Paste"        | GI            |
| XtCUndoLabel   | OlStr       | "Undo"         | GI            |

**Synopsis:** The label for the Copy, Cut, Delete, Paste, and Undo buttons in the Edit menu, respectively.

**Values:** Any `OlStr` value valid in the current locale.

***XtNcopyMnemonic/  
XtNcutMnemonic/  
XtNdeleteMnemonic/  
XtNpasteMnemonic/  
XtNundoMnemonic***

| <b>Class</b>      | <b>Type</b>   | <b>Default</b> | <b>Access</b> |
|-------------------|---------------|----------------|---------------|
| XtCCopyMnemonic   | unsigned char | '\0'           | GI            |
| XtCCutMnemonic    | unsigned char | '\0'           | GI            |
| XtCDeleteMnemonic | unsigned char | '\0'           | GI            |
| XtCPasteMnemonic  | unsigned char | '\0'           | GI            |
| XtCUndoMnemonic   | unsigned char | '\0'           | GI            |

**Synopsis:** The mnemonic for the Copy, Cut, Delete, Paste, and Undo buttons in the Edit menu, respectively.

**Values:** Any ASCII character.

TextEdit Widget

**XtNcursorPosition**

| Class           | Type         | Default | Access |
|-----------------|--------------|---------|--------|
| XtCTextPosition | TextPosition | 0       | SGI    |

Synopsis: The relative character position in the text of the insert cursor.

Values:  $0 \leq \text{XtNcursorPosition} < \text{number-of-chars-in-buffer}$

Changing the value of this resource may affect the XtNdisplayPosition resource if the XtNcursorPosition value is not visible in the pane.

**XtNdisplayPosition**

| Class           | Type         | Default | Access |
|-----------------|--------------|---------|--------|
| XtCTextPosition | TextPosition | 0       | SGI    |

Synopsis: The character position in the text that will be displayed at the top of the pane.

Values:  $0 \leq \text{XtNdisplayPosition} < \text{number-of-chars-in-textbuffer}$

A value of 0 indicates the beginning of the text source. When the value provided is near the end of the buffer, this position is recalculated to ensure that the last line in the buffer appears as the last line in the pane.

**XtNeditType**

| Class       | Type       | Default      | Access |
|-------------|------------|--------------|--------|
| XtCEditType | OLEditMode | OL_TEXT_EDIT | SGI    |

Synopsis: The edit state of the widget.

Values: OL\_TEXT\_READ/"textread" - The contents are read-only; the user cannot edit any text.  
 OL\_TEXT\_EDIT/"textedit" - The text is fully editable.

**XtNgrowMode**

| Class       | Type     | Default     | Access |
|-------------|----------|-------------|--------|
| XtCGrowMode | OLDefine | OL_GROW_OFF | SGI    |

Synopsis: The resize policy of the widget (in conjunction with XtNwrapmode).

Values: OL\_GROW\_OFF/"grow\_off" - The widget will not grow either horizontally or vertically when text exceeds window boundaries.  
 OL\_GROW\_HORIZONTAL/"grow\_horizontal" - The widget will grow horizontally, if the text exceeds the width of the window. This setting is valid only when XtNwrapMode is OL\_WRAP\_OFF.

*TextEdit Widget*

OL\_GROW\_VERTICAL/"grow\_vertical" - The widget will grow vertically, if the text exceeds the height of the window.  
 OL\_GROW\_BOTH/"grow\_both" - The widget will grow both horizontally and vertically as required, if the text exceeds the window width or height. This setting is valid only when XtNwrapMode is OL\_WRAP\_OFF.

***XtNimPreeditStyle***

| Class             | Type            | Default       | Access |
|-------------------|-----------------|---------------|--------|
| XtCImPreeditStyle | OImPreeditStyle | OL_NO_PREEDIT | GI     |

Synopsis: The pre-edit style (in conjunction with the shell's XtNimStatusStyle resource). If the pre-edit style is not supported by the input method, the ability to pre-edit is lost.

Values: OL\_ON\_THE\_SPOT/"onTheSpot" - The pre-edit data is displayed at the insertion point in the application window. The preexisting user data is shifted and the pre-edit data is inserted at the point of insertion.  
 OL\_OVER\_THE\_SPOT/"overTheSpot" - The pre-edit data is displayed in the application window, starting at the insertion point. As the user types the pre-edit data, the preexisting user data is obscured by the pre-edit data.  
 OL\_ROOT\_WINDOW/"rootWindow" - The pre-edit data is displayed in a child of the root window, away from the point of insertion.  
 OL\_NO\_PREEDIT/"none" - No pre-edit data is displayed.

See "XtNimStatusStyle" on page 44 and "Setting the Input Method Pre-Edit and Status Styles (Asian Locales Only)" on page 82.

***XtNinsertTab***

| Class        | Type    | Default | Access |
|--------------|---------|---------|--------|
| XtCInsertTab | Boolean | TRUE    | SGI    |

Synopsis: Determines whether a tab character is insertable.

Values: TRUE/"true" - Tabs are insertable but not used for forward traversal. Forward traversal can still be accomplished with Control-Tab.  
 FALSE/"false" - Tabs are used for forward traversal and are not insertable.

**XtNkeys**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when a key press is made in the pane.

The *call\_data* parameter is a pointer to an `OlInputCallData` structure, as shown in “XtNbuttons” on page 631.

This callback can be used to remap or consume key events, such as to post a property window for the text. To do so, the application would add a `XtNkeys` callback that checks the *ol\_event* member of the `OlInputCallData` structure. If the value of the *ol\_event* member is `OL_PROP`, the callback posts the property window (or raises it if it is already mapped) and sets the *consumed* member of the `OlInputCallData` structure to `TRUE`.

**XtNleftMargin**

| Class     | Type      | Default | Access |
|-----------|-----------|---------|--------|
| XtCMargin | Dimension | 4       | SGL    |

Synopsis: The number of pixels used for the width of the left margin.

**XtNlinesVisible**

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCLinesVisible | int  | 16      | GI     |

Synopsis: The initial height of the text in terms of display lines.

Values:  $1 \leq \text{XtNlinesVisible}$

This resource overrides the `XtNheight` resource setting. The `XtNheight` is recalculated to be the value of `XtNlinesVisible` multiplied by the height of the font plus the values for the top and bottom margins.

The value of `XtNlinesVisible` changes to reflect the effects of geometry changes imposed by the widget tree and the user. Calls to `XtSetValues()` for this resource are ignored.

**XtNmargin**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the pane is redisplayed.

**TextEdit Widget**

The *call\_data* parameter is a pointer to an `OlTextMarginCallData` structure:

```
typedef struct {
 OlTextMarginHint hint;
 XRectangle *rect;
} OlTextMarginCallData, *OlTextMarginCallDataPointer;

typedef enum {
 OL_MARGIN_EXPOSED,
 OL_MARGIN_CALCULATED
} OlTextMarginHint;
```

The *hint* member indicates whether the area to be redrawn was explicitly known because of an exposure event (`OL_MARGIN_EXPOSED`) or if the rectangle was calculated relative to the textual display (`OL_MARGIN_CALCULATED`). The margin callback should respond to the `OL_MARGIN_EXPOSED` hint by repainting the area defined by the *rect* member.

The margin callback may wish to calculate its own rectangle in the `OL_MARGIN_CALCULATED` case. It can use the rectangle structure passed with the *call\_data* for this purpose. This callback can be used to repair the margins for the text, such as to display line numbers for the text in the left margin.

**XtNmenuTitle**

| Class        | Type  | Default | Access |
|--------------|-------|---------|--------|
| XtCMenuTitle | OlStr | "Edit"  | GI     |

Synopsis: The title of the TextEdit menu.

Values: Any `OlStr` value valid in the current locale.

**XtNmodifyVerification**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when a modification of the text is attempted.

The *call\_data* parameter is a pointer to an `OlTextModifyCallData` structure:

```
typedef struct {
 Boolean ok;
 TextPosition current_cursor;
 TextPosition select_start;
 TextPosition select_end;
 TextPosition new_cursor;
 TextPosition new_select_start;
```

*TextEdit Widget*

```

 TextPosition new_select_end;
 char *text;
 int text_length;
 } OlTextModifyCallData, *OlTextModifyCallDataPointer;

```

All of the fields in this structure, with the exception of the *ok* flag, are treated as read-only information. The application can return without changing the value of *ok* (initially TRUE), in which case the update will occur. The application can also set the *ok* flag to FALSE, perform any other operations it desires, and return, in which case the update will not be performed.

***XtNmotionVerification***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked whenever the cursor position moves within the widget.

The *call\_data* parameter is a pointer to an OlTextMotionCallData structure:

```

typedef struct {
 Boolean ok;
 TextPosition current_cursor;
 TextPosition new_cursor;
 TextPosition select_start;
 TextPosition select_end;
} OlTextMotionCallData, *OlTextMotionCallDataPointer;

```

This callback list is used whenever the cursor position changes due to cursor movement operations or by modification of the text.

The application can distinguish between a simple cursor movement and a modify operation by comparing the *current\_cursor* and *new\_cursor* values. When these values are equal, the callback is the result of a modify operation. In this case the *ok* flag is ignored and the application should not attempt to perform updates to the text or its display during this callback.

If the values of *current\_cursor* and *new\_cursor* are different, the application is guaranteed that the operation is the result of a cursor movement. In this mode all of the fields in this structure, with the exception of the *ok* flag, are treated as read-only information.



*TextEdit Widget*

The application can return without changing the value of the *ok* flag (initially TRUE), in which case the movement will occur. The application can also set *ok* to FALSE, perform any other operations it desires and return, in which case the movement will not be performed.

***XtNpostModifyNotification***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked after a text update has completed.

The *call\_data* parameter is a pointer to an `OlTextPostModifyCallData` structure:

```
typedef struct {
 Boolean requestor;
 TextPosition new_cursor;
 TextPosition new_select_start;
 TextPosition new_select_end;
 char *inserted;
 char *deleted;
 TextLocation delete_start;
 TextLocation delete_end;
 TextLocation insert_start;
 TextLocation insert_end;
 TextLocation cursor_position;
} OlTextPostModifyCallData, *OlTextPostModifyCallDataPointer;
```

This callback synchronizes the application with the text once a modify operation is completed. For example, the application may record successful edit operations in an undo buffer to provide multilevel undo functionality. The data provided in this callback is considered read-only and volatile (the application should copy what it needs from this structure before returning).

***XtNrightMargin***

| Class     | Type      | Default | Access |
|-----------|-----------|---------|--------|
| XtCMargin | Dimension | 4       | SGI    |

Synopsis: The size of the right margin, in pixels.

*TextEdit Widget*

***XtNselectEnd /  
XtNselectStart***

| <b>Class</b>    | <b>Type</b>  | <b>Default</b> | <b>Access</b> |
|-----------------|--------------|----------------|---------------|
| XtCTextPosition | TextPosition | 0              | SGI           |
| XtCTextPosition | TextPosition | 0              | SGI           |

Synopsis: The character position of the end/start of the current text selection.

This resource is used with `XtNselectStart` and `XtNcursorPosition` to specify a selection. To be effective, the following conditions must be true:

```

XtNselectStart ≤ XtNselectEnd
 and either XtNcursorPosition == XtNselectStart
 or
 XtNcursorPosition == XtNselectEnd

```

If either of these tests fails, then `XtNselectStart` and `XtNselectEnd` are set to the value of `XtNcursorPosition`.

***XtNsource***

| <b>Class</b> | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|--------------|-------------|----------------|---------------|
| XtCSource    | OlStr       | NULL           | SGI           |

Synopsis: The source of the text to display.

Values: The datatype of the value depends on the value of `XtNsourceType`:

| <b>XtNsourceType</b>    | <b>XtNsource Contents</b>                                                                |
|-------------------------|------------------------------------------------------------------------------------------|
| OL_STRING_SOURCE        | Any <code>OlStr</code> value valid in the current locale.                                |
| OL_DISK_SOURCE          | A string that represents the pathname of a file containing the source text.              |
| OL_TEXT_BUFFER_SOURCE   | A pointer to a <code>TextBuffer</code> containing the source text in single-byte format. |
| OL_OLTEXT_BUFFER_SOURCE | A pointer to a <code>TextBuffer</code> containing the source text in multi-byte format.  |

The `TextEdit` widget does not guarantee that an `XtGetValues()` on this resource will return the correct data and datatype. Internally, the passed-in values are overwritten. To get at the current text contents, use the `TextEdit` function `OlTextEditCopyBuffer()` (see page 660).

**XtNsourceType**

| Class         | Type         | Default          | Access |
|---------------|--------------|------------------|--------|
| XtCSourceType | OlSourceType | OL_STRING_SOURCE | SGI    |

Synopsis: The interpretation of the XtNsource resource value.

Values: OL\_STRING\_SOURCE/"stringsource" - The XtNsource value is interpreted as the string to be used as the source.  
 OL\_DISK\_SOURCE/"disksource" - The XtNsource value is interpreted as the name of the file containing the source.  
 OL\_TEXT\_BUFFER\_SOURCE - The XtNsource value is interpreted as a pointer to a previously initialized single-byte format TextBuffer (see "Text Buffer Functions" on page 163 for a description of TextBuffers). This value can only be set programmatically; there is no corresponding resource file string.  
 OL\_OLTEXT\_BUFFER\_SOURCE - The XtNsource value is interpreted as a pointer to a previously initialized multi-byte format TextBuffer (see "Text Buffer Functions" on page 163 for a description of TextBuffers). This value can only be set programmatically; there is no corresponding resource file string.

**XtNtabTable**

| Class       | Type     | Default | Access |
|-------------|----------|---------|--------|
| XtCTabTable | TabTable | NULL    | SGI    |

Synopsis: The pointer to an array of tab positions.

The tab positions are specified in terms of pixels and must be terminated by a zero (0) entry. The widget calculates tabs by finding the next tab table entry that exceeds the current x offset for the line. If no such entry exists in the table or if the pointer to the tab table is NULL, the tab is set to the next greater multiple of 8 times the size of the "n" (en) character in the font.

**XtNtopMargin**

| Class     | Type      | Default | Access |
|-----------|-----------|---------|--------|
| XtCMargin | Dimension | 4       | SGI    |

Synopsis: The number of pixels used for the top margin.

**XtNwrapMode**

| Class       | Type       | Default             | Access |
|-------------|------------|---------------------|--------|
| XtCWrapMode | OIWrapMode | OL_WRAP_WHITE_SPACE | SGI    |

Synopsis: The wrapping of the text in the pane.

Values: OL\_WRAP\_ANY/"wrapany" - Lines are wrapped at the last character before the right margin.  
 OL\_WRAP\_WHITE\_SPACE/"wrapwhitespace" - Lines are wrapped at the last white space before the right margin or at the last character before the right margin if the line does not contain any white space.  
 OL\_WRAP\_OFF/"wrapoff" - Lines are not wrapped and the pane may scroll horizontally.

**Activation Types**

The following table lists the activation types used by the TextEdit.

Table 10-4 TextEdit Activation Types

| Activation Type  | Semantics      | Resource Name       |
|------------------|----------------|---------------------|
| OL_ADJUST        | ADJUST         | XtNadjustBtn        |
| OL_ADJUSTKEY     | ADJUST         | XtNadjustKey        |
| OL_CANCEL        | CANCEL         | XtNcancelKey        |
| OL_CHARBAK       | LEFT           | XtNleftKey          |
| OL_CHARFWD       | RIGHT          | XtNrightKey         |
| OL_COPY          | COPY           | XtNcopyBtn          |
| OL_CUT           | CUT            | XtNcutBtn           |
| OL_DEFAULTACTION | DEFAULTACTION  | XtNdefaultActionKey |
| OL_DELCHARBAK    | DELETEBACKWARD | XtNdelCharBakFwd    |
| OL_DELCHARFWD    | DELETEFORWARD  | XtNdelCharFwdKey    |
| OL_DELLINE       | DELETELIN      | XtNdelLineKey       |
| OL_DELLINEBAK    | DELLINEBAK     | XtNdelLineBakKey    |
| OL_DELLINEFWD    | DELLINEFWD     | XtNdelLineFwdKey    |
| OL_DELWORDBAK    | DELWORDBAK     | XtNdelWordBakKey    |
| OL_DELWORDFWD    | DELWORDFWD     | XtNdelWordFwdKey    |
| OL_DOCEND        | DATAEND        | XtNdocEndKey        |
| OL_DOCSTART      | DATASTART      | XtNdocStartKey      |

Table 10-4 TextEdit Activation Types (Continued)

| Activation Type    | Semantics       | Resource Name         |
|--------------------|-----------------|-----------------------|
| OL_HELP            | HELP            | XtNhelpKey            |
| OL_LINEEND         | ROWEND          | XtNlineEndKey         |
| OL_LINESTART       | ROWSTART        | XtNlineStartKey       |
| OL_MENU            | MENU            | XtNmenuBtn            |
| OL_MENUKEY         | MENU            | XtNmenuKey            |
| OL_MOVEDOWN        | MOVEDOWN        | XtNdownKey            |
| OL_MOVELEFT        | MOVELEFT        | XtNleftKey            |
| OL_MOVERIGHT       | MOVERIGHT       | XtNrightKey           |
| OL_MOVEUP          | MOVEUP          | XtNupKey              |
| OL_NEXTFIELD       | NEXTFIELD       | XtNnextFieldKey       |
| OL_PAGEDOWN        | PAGEDOWN        | XtNpageDownKey        |
| OL_PAGELEFT        | PAGELEFT        | XtNpageLeftKey        |
| OL_PAGERIGHT       | PAGERIGHT       | XtNpageRightKey       |
| OL_PAGEUP          | PAGEUP          | XtNpageUpKey          |
| OL_PANEEND         | PANEDOWN        | XtNpaneEndKey         |
| OL_PANESTART       | PANEUP          | XtNpaneStartKey       |
| OL_PASTE           | PASTE           | XtNpasteBtn           |
| OL_PREVFIELD       | PREVFIELD       | XtNprevFieldKey       |
| OL_ROWDOWN         | DOWN            | XtNdownKey            |
| OL_ROWUP           | UP              | XtNupKey              |
| OL_SCROLLDOWN      | SCROLLDOWN      | XtNscrollDownKey      |
| OL_SCROLLLEFT      | SCROLLLEFT      | XtNscrollLeftKey      |
| OL_SCROLLLEFTEDGE  | SCROLLLEFTEDGE  | XtNscrollLeftEdgeKey  |
| OL_SCROLLRIGHT     | SCROLLRIGHT     | XtNscrollRightKey     |
| OL_SCROLLRIGHTEDGE | SCROLLRIGHTEDGE | XtNscrollRightEdgeKey |
| OL_SCROLLUP        | SCROLLUP        | XtNscrollUpKey        |
| OL_SELCHARBAK      | SELCHARBAK      | XtNselCharBakKey      |
| OL_SELCHARFWD      | SELCHARFWD      | XtNselCharFwdKey      |
| OL_SELECT          | SELECT          | XtNselectBtn          |
| OL_SELECTKEY       | SELECT          | XtNselectKey          |
| OL_SELFLIPENDS     | SELFLIPENDS     | XtNselFlipEndsKey     |
| OL_SELLINE         | SELLINE         | XtNselLineKey         |
| OL_SELLINEBAK      | SELLINEBAK      | XtNselLineBakKey      |

Table 10-4 TextEdit Activation Types (Continued)

| Activation Type  | Semantics     | Resource Name       |
|------------------|---------------|---------------------|
| OL_SELLINEFWD    | SELLINEFWD    | XtNselLineFwdKey    |
| OL_SELWORDBAK    | SELWORDBAK    | XtNselWordBakKey    |
| OL_SELWORDFWD    | SELWORDFWD    | XtNselWordFwdKey    |
| OL_TOGGLEPUSHPIN | TOGGLEPUSHPIN | XtNtogglePushpinKey |
| OL_UNDO          | UNDO          | XtNundoKey          |
| OL_WORDBAK       | JUMPLEFT      | XtNwordBakKey       |
| OL_WORDFWD       | JUMPRIGHT     | XtNwordFwdKey       |

Activation types not described in the following list are described in “Common Activation Types” on page 68.

**OL\_ADJUST**

The OL\_ADJUST activation type first calls the XtNbuttons callback list with the appropriate OlInputCallData structure. If there is no XtNbuttons callback or if the *consumed* field of the OlInputCallData structure is TRUE, then processing continues. The OL\_ADJUST activation type calls the XtNmotionVerification callback list with the appropriate OlTextMotionCallData structure containing the new cursor position, selection start, and selection end. If a callback is not registered or the OlTextMotionCallData’s *ok* field is TRUE, the OL\_ADJUST activation type will modify the XtNcursorPosition, XtNselectStart, and XtNselectEnd resources to reflect the cursor position and the resulting selection.

**OL\_CHARBAK**

The OL\_CHARBAK activation type first calls the XtNkeys callback list with the appropriate OlInputCallData structure. If there is no XtNkeys callback or if the *consumed* field of the OlInputCallData structure is TRUE, then processing continues. The activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the cursor position moved one character before the current cursor position. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the cursor will be moved. When the cursor is positioned before the first character on a line, the

OL\_CHARBAK activation type will cause the previous line to be brought into view and the `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_CHARFWD***

The OL\_CHARFWD activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is TRUE, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved one character after the current cursor position. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is TRUE, then the cursor will be moved. When the cursor is positioned after the last character on a line, the activation type will cause the next line to be brought into view and the `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_COPY***

The OL\_COPY activation type first calls the `XtNbuttons` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNbuttons` callback or if the *consumed* field of the `OlInputCallData` structure is TRUE, then processing continues. The activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure containing the currently selected items. If the `OlTextModifyCallData` *ok* field is TRUE upon return, then the selected text will be copied to the CLIPBOARD.

### ***OL\_CUT***

The OL\_CUT activation type first calls the `XtNbuttons` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNbuttons` callback or if the *consumed* field of the `OlInputCallData` structure is TRUE, then processing continues. The activation type applies only to `TextEdit` widgets that have `XtNeditType` of `OL_TEXT_EDIT`. The `XtNmodifyVerification` callback list will be called with an `OlTextModifyCallData` structure containing the currently selected items. If the `OlTextModifyCallData` *ok* field is TRUE upon return, then the selected text will be copied to the CLIPBOARD.

***OL\_DELCHARBAK***

The *OL\_DELCHARBAK* activation type first calls the *XtNkeys* callback list with the appropriate *OlInputCallData* structure. If there is no *XtNkeys* callback or if the *consumed* field of the *OlInputCallData* structure is *TRUE*, then processing continues. The activation type calls the *XtNmodifyVerification* callback list with an *OlTextModifyCallData* structure that represents the selection, cursor position, and text after the character to the left of the cursor has been deleted. If there is no *XtNmodifyVerification* callback or if the *ok* field of the *OlTextModifyCallData* structure is *TRUE*, then the text will be updated. Then the activation type calls the *XtNmotionVerification* callback list with an *OlTextMotionCallData* structure that represents the selection and cursor position after the character has been deleted. If there is no *XtNmotionVerification* callback or if the *ok* field of the *OlTextMotionCallData* structure is *TRUE*, then the cursor and selection will be updated. The *XtNmargin* callback list will be called with the appropriate *OlTextMarginCallData* structure. Finally, the *XtNpostModifyNotification* callback list will be called with the appropriate *OlTextPostModifyCallData* structure.

***OL\_DELCHARFWD***

The *OL\_DELCHARFWD* activation type first calls the *XtNkeys* callback list with the appropriate *OlInputCallData* structure. If there is no *XtNkeys* callback or if the *consumed* field of the *OlInputCallData* structure is *TRUE*, then processing continues. The activation type calls the *XtNmodifyVerification* callback list with an *OlTextModifyCallData* structure that represents the selection, cursor position, and text after the character to the right of the cursor has been deleted. If there is no *XtNmodifyVerification* callback or if the *ok* field of the *OlTextModifyCallData* structure is *TRUE*, then the text will be updated. Then the activation type calls the *XtNmotionVerification* callback list with an *OlTextMotionCallData* structure that represents the selection and cursor position after the character has been deleted. If there is no *XtNmotionVerification* callback or if the *ok* field of the *OlTextMotionCallData* structure is *TRUE*, then the cursor and selection will be updated. The *XtNmargin* callback list will be called with the appropriate *OlTextMarginCallData* structure. Finally, the *XtNpostModifyNotification* callback list will be called with the appropriate *OlTextPostModifyCallData* structure.



### ***OL\_DELLINE***

The `OL_DELLINE` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the current line has been deleted. If there is no `XtNmodifyVerification` callback or if the *ok* field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the line has been deleted. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

### ***OL\_DELLINEBAK***

The `OL_DELLINEBAK` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the line to the left of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the *ok* field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the line has been deleted. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELLINEFWD***

The `OL_DELLINEFWD` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the line to the right of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the *ok* field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the line has been deleted. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELWORDBAK***

The `OL_DELWORDBAK` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the word to the left of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the *ok* field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the word has been deleted. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELWORDFWD***

The `OL_DELWORDFWD` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the word to the right of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the *ok* field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the word has been deleted. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DOCEND***

The `OL_DOCEND` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved to after the last character on the last line of the `TextBuffer`. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved and the text may be scrolled to bring the last line into view. If the text is scrolled to bring the last line into view, the `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

***OL\_DOCSTART***

The `OL_DOCSTART` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved to before the first character on the first line of the `TextBuffer`.

If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved and the text may be scrolled to bring the first line into view. If the text is scrolled to bring the first line into view, the `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_LINEEND***

The `OL_LINEEND` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the `consumed` field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved to after the last character on the current line. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

### ***OL\_LINESTART***

The `OL_LINESTART` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the `consumed` field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved to before the first character on the current line. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

### ***OL\_MENU/ OL\_MENUKEY***

The `OL_MENU` activation type first calls the `XtNbuttons` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNbuttons` callback or if the `consumed` field of the `OlInputCallData` structure is `TRUE`, then processing continues. The `OL_MENUKEY` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the `consumed` field of the `OlInputCallData` structure is `TRUE`, then processing continues. The `OL_MENU` and `OL_MENUKEY` activation types will pop up the `TextEdit` menu.

***OL\_PAGEDOWN***

The `OL_PAGEDOWN` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the page after the current view into the view.

***OL\_PAGELEFT***

The `OL_PAGELEFT` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The `OL_PAGELEFT` activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the page to the left of the current view into the view.

***OL\_PAGERIGHT***

The `OL_PAGERIGHT` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The `OL_PAGERIGHT` activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the page to the right of the current view into the view.

***OL\_PAGEUP***

The `OL_PAGEUP` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The `OL_PAGEUP` activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the page before the current view into the view.

***OL\_PANEEND***

The `OL_PANEEND` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor

position moved to after the last character on the last line in the current view. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

### ***OL\_PANESTART***

The `OL_PANESTART` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the `consumed` field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved to before the first character on the first line in the current view. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

### ***OL\_PASTE***

The `OL_PASTE` activation type first calls the `XtNbuttons` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNbuttons` callback or if the `consumed` field of the `OlInputCallData` structure is `TRUE`, then processing continues. The `OL_PASTE` activation type applies only to `TextEdit` widgets that have `XtNeditType` of `OL_TEXT_EDIT`. When the widget has focus, then the `OL_PASTE` activation type will insert the contents of the `CLIPBOARD` at the current insert position by calling the `XtNpostModifyVerification` callback list. The `XtNmotionVerification` callback list will be called with an `OlTextMotionCallData` structure that represents the new cursor and selection position.

### ***OL\_ROWDOWN***

The `OL_ROWDOWN` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the `consumed` field of the `OlInputCallData` structure is `TRUE`, then processing continues. The `OL_ROWDOWN` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved one line after the current cursor position. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved and the text may be scrolled to bring the new line into view. When a new line is brought into the view, the `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

***OL\_ROWUP***

The `OL_ROWUP` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved one line previous to the current cursor position. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved and the text may be scrolled to bring the new line into view. When a new line is brought into the view, the `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

***OL\_SCROLLDOWN***

The `OL_SCROLLDOWN` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the line after the current view into the view.

***OL\_SCROLLLEFT***

The `OL_SCROLLLEFT` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the character left of the current view into the view.

***OL\_SCROLLLEFTEDGE***

The `OL_SCROLLLEFTEDGE` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the left-most character of the document into the view.

***OL\_SCROLLRIGHT***

The `OL_SCROLLRIGHT` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the character right of the current view into the view.

***OL\_SCROLLRIGHTEDGE***

The `OL_SCROLLRIGHTEDGE` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the rightmost character of the document into the view.

***OL\_SCROLLUP***

The `OL_SCROLLUP` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the line before the current view into the view.

***OL\_SELCHARBAK***

The `OL_SELCHARBAK` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended one character to the left of the cursor position and the cursor position moved to before the selection. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, the cursor will be moved, and the text may be scrolled to bring the previous line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.



### **OL\_SELCHARFWD**

The OL\_SELCHARFWD activation type first calls the XtNkeys callback list with the appropriate OlInputCallData structure. If there is no XtNkeys callback or if the *consumed* field of the OlInputCallData structure is TRUE, then processing continues. The activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the selection extended one character to the right of the cursor position and the cursor position moved to after the selection. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the selection will be extended, the cursor will be moved, and the text may be scrolled to bring the next line into view. The XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure.

### **OL\_SELECT**

The OL\_SELECT activation type first calls the XtNbuttons callback list with the appropriate OlInputCallData structure. If there is no XtNbuttons callback or if the *consumed* field of the OlInputCallData structure is TRUE, then processing continues. The activation type calls the XtNmotionVerification callback list with the appropriate OlTextMotionCallData structure containing the new cursor position, selection start, and selection end. If a callback is not registered or the OlTextMotionCallData's *ok* field is TRUE, the OL\_SELECT activation type will modify the XtNcursorPosition, XtNselectStart, and XtNselectEnd resources to reflect the cursor position and the resulting selection.

### **OL\_SELFLIPENDS**

The OL\_SELFLIPENDS activation type first calls the XtNkeys callback list with the appropriate OlInputCallData structure. If there is no XtNkeys callback or if the *consumed* field of the OlInputCallData structure is TRUE, then processing continues. The activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the cursor position moved to the opposite end of the current selection. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the cursor will be moved. If the text is scrolled to bring the cursor into view, the XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure.

***OL\_SELLINE***

The `OL_SELLINE` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the beginning of the current line and to the end of the current line, and the cursor position moved to before the selection. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

***OL\_SELLINEBAK***

The `OL_SELLINEBAK` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the beginning of the line to the left of the current cursor position and the cursor position moved to before the selection. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, the cursor will be moved, and the text may be scrolled to bring the previous line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

***OL\_SELLINEFWD***

The `OL_SELLINEFWD` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the end of the line to the right of the current cursor position and the cursor position moved to after the selection. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be

extended, the cursor will be moved, and the text may be scrolled to bring the next line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_SELWORDBAK***

The `OL_SELWORDBAK` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the beginning of the word to the left of the current cursor position and the cursor position moved to before the selection. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, the cursor will be moved, and the text may be scrolled to bring the previous line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_SELWORDFWD***

The `OL_SELWORDFWD` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the end of the word to the right of the current cursor position and the cursor position moved to after the selection. If there is no `XtNmotionVerification` callback or if the *ok* field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, the cursor will be moved, and the text may be scrolled to bring the next line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_UNDO***

The `OL_UNDO` activation type first calls the `XtNkeys` callback list with the appropriate `OlInputCallData` structure. If there is no `XtNkeys` callback or if the *consumed* field of the `OlInputCallData` structure is `TRUE`, then processing continues. The activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text before the last modification. If there is no

XtNmodifyVerification callback or if the *ok* field of the OlTextModifyCallData structure is TRUE, then the text will be updated. Then the activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the selection and cursor position before the last modification. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the cursor and selection will be updated. The XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure. Finally, the XtNpostModifyNotification callback list will be called with the appropriate OlTextPostModifyCallData structure.

### **OL\_WORDBAK**

The OL\_WORDBAK activation type first calls the XtNkeys callback list with the appropriate OlInputCallData structure. If there is no XtNkeys callback or if the *consumed* field of the OlInputCallData structure is TRUE, then processing continues. The activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the cursor position moved one word before the current cursor position. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the cursor will be moved. When the cursor is positioned before the first word on a line, the previous line will be brought into view and the XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure.

### **OL\_WORDFWD**

The OL\_WORDFWD activation type first calls the XtNkeys callback list with the appropriate OlInputCallData structure. If there is no XtNkeys callback or if the *consumed* field of the OlInputCallData structure is TRUE, then processing continues. The activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the cursor position moved one word after the current cursor position. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the cursor will be moved. When the cursor is positioned after the last word on a line, the next line will be brought into view and the XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure.

*See Also*

“StaticText Widget” on page 600,  
“TextEdit Functions” on page 660,  
“TextField Widget” on page 665,  
“TextField Functions” on page 686,  
“TextLine Widget” on page 688,  
“TextLine Functions” on page 708,  
“Text Selection Operations” on page 204.

***TextEdit Functions***

---

The following functions assist in manipulating TextEdit widgets.

***OlTextEditClearBuffer***

```
#include <Xol/textbuff.h>
Boolean OlTextEditClearBuffer(
 TextEditWidget ctx);
```

`OlTextEditClearBuffer()` deletes all of the text associated with the TextEdit widget *ctx*. It returns FALSE if the widget supplied is not a TextEdit widget or if the clear operation fails; otherwise, it returns TRUE.

***OlTextEditReadSubString***

```
#include <Xol/textbuff.h>
Boolean OlTextEditReadSubString(
 TextEditWidget ctx,
 char **buffer,
 TextPosition start,
 TextPosition end);
```

`OlTextEditReadSubString()` retrieves a copy of a substring from the TextBuffer associated with the TextEdit widget. The storage required for the copy is allocated by this routine; it is the responsibility of the caller to free this storage when appropriate. It returns FALSE if the widget supplied is not a TextEdit widget or if the operation fails; otherwise, it returns TRUE.

***OlTextEditCopyBuffer***

```
#include <Xol/textbuff.h>
Boolean OlTextEditCopyBuffer(
 TextEditWidget ctx,
 char *buffer);
```

`OlTextEditCopyBuffer()` retrieves a copy of the contents of the TextBuffer associated with the TextEdit widget *ctx*. The storage required for the copy is allocated by this routine; it is the responsibility of the caller to free this storage when appropriate. The function returns FALSE if the widget supplied is not a TextEdit widget or if the buffer cannot be read; otherwise, it returns TRUE.

---

## TextEdit Functions

When `OlTextEditCopyBuffer()` returns, *buffer* contains the text in the `TextBuffer` of the widget in an (`OlStr *`); the `char **` shown in the synopsis is for binary compatibility with previous versions.

### *OlTextEditCopySelection*

```
#include <Xol/textbuff.h>
Boolean OlTextEditCopySelection(
 TextEditWidget ctx,
 int delete);
```

`OlTextEditCopySelection()` Copies or Cuts the current selection in the `TextEdit` widget *ctx*. If no selection exists, or if the `TextEdit` cannot acquire the `CLIPBOARD`, or if the widget supplied is not a `TextEdit` widget, `FALSE` is returned. Otherwise, the selection is copied to the `CLIPBOARD` then, if the *delete* flag is nonzero, the text is then deleted from the `TextBuffer` associated with the `TextEdit` widget (i.e., a Cut operation is performed). Finally, `TRUE` is returned.

### *OlTextEditRedraw*

```
#include <Xol/textbuff.h>
Boolean OlTextEditRedraw(
 TextEditWidget ctx);
```

`OlTextEditRedraw()` forces a complete refresh of the `TextEdit` widget display. It returns `FALSE` if the widget supplied is not a `TextEdit` widget or if the widget is not realized or if the update state is `FALSE`; otherwise, it returns `TRUE`.

### *OlTextEditGetCursorPosition*

```
#include <Xol/textbuff.h>
Boolean OlTextEditGetCursorPosition(
 TextEditWidget ctx,
 TextPosition *start,
 TextPosition *end,
 TextPosition *cursorPosition);
```

`OlTextEditGetCursorPosition()` retrieves the current selection *start*, *end*, and *cursorPosition*. The *start* value will represent the position of the first character in the selection; the *end* value will be the position of the character

after the last character in the selection. (For example, if the `TextBuffer` contains `abc` and the selection is `ab`, `start` will return as 0 and `end` as 2.) If there is no current selection, `start` and `end` will both be equal to `cursorPosition`. The function returns `FALSE` if the widget supplied is not a `TextEdit` widget; otherwise, it returns `TRUE`.

### *OlTextEditSetCursorPosition*

```
#include <Xol/textbuff.h>
Boolean OlTextEditSetCursorPosition(
 TextEditWidget ctx,
 TextPosition start,
 TextPosition end,
 TextPosition cursorPosition);
```

`OlTextEditSetCursorPosition()` changes the current selection `start` and `end` and `cursorPosition`. For efficiency, the function does not check the validity of the positions. If invalid values are given, results are unpredictable. The function attempts to ensure that the `cursorPosition` is visible by scrolling the display. It returns `FALSE` if the widget supplied is not a `TextEdit` widget; otherwise, it returns `TRUE`.

### *OlTextEditGetLastPosition*

```
#include <Xol/textbuff.h>
Boolean OlTextEditGetLastPosition(
 TextEditWidget ctx,
 TextPosition *position);
```

`OlTextEditGetLastPosition()` retrieves the `TextEdit` widget `ctx`. It returns `FALSE` if the widget supplied is not a `TextEdit` widget; otherwise, it returns `TRUE`.

### *OlTextEditMoveDisplayPosition*

```
#include <Xol/textbuff.h>
void OlTextEditMoveDisplayPosition(
 TextEditWidget ctx,
 OlInputEvent move_type);
```



---

## TextEdit Functions

`OlTextEditMoveDisplayPosition()` moves the display position and performs scroll updates in the `TextEdit` widget. It is recommended that this function be used instead of `XtSetValues()` to move the display position. Using `XtSetValues()` may result in a screen flicker.

The *move\_type* parameter can have one of the following values:

|                            |                                           |
|----------------------------|-------------------------------------------|
| <code>OL_SCROLLUP</code>   | Scroll up a line                          |
| <code>OL_SCROLLDOWN</code> | Scroll down a line                        |
| <code>OL_PAGEUP</code>     | Scroll a page up                          |
| <code>OL_PAGEDOWN</code>   | Scroll a page down                        |
| <code>OL_HOME</code>       | Scroll to the beginning of the text(home) |
| <code>OL_END</code>        | Scroll to the end                         |

### *OlTextEditTextBuffer*

```
#include <Xol/textbuff.h>
TextBuffer *OlTextEditTextBuffer(
 TextEditWidget ctx);
```

`OlTextEditTextBuffer()` retrieves the `TextBuffer` pointer associated with the `TextEdit` widget *ctx*. This pointer can be used to access the facilities provided by the functions in “Text Buffer Functions” on page 163 and “Text Buffer Functions for Internationalization” on page 176. The function returns `NULL` if the text format of the *ctx* widget is not single-byte.

### *OlTextEditOlTextBuffer*

```
#include <Oltextbuff.h>
OlTextBufferPtr OlTextEditOlTextBuffer(
 TextEditWidget ctx);
```

`OlTextEditOlTextBuffer()` retrieves the `OlTextBufferPtr` associated with the `TextEdit` widget *ctx*. `OlTextBufferPtr` is an opaque pointer that points to a `TextBuffer` that is capable of handling multibyte and wide character data. This type of text buffer is only associated with a multibyte (`OL_MB_STR_REP`) or wide character (`OL_WC_STR_REP`) `TextEdit` widget.

The function returns `NULL` if the text format of the *ctx* widget is single-byte. It returns an `OlTextBufferPtr` if the text format of *ctx* is multibyte or wide character. For a single-byte (`OL_SB_STR_REP`) `TextEdit` widget, use `OlTextEditTextBuffer()`.

### *OlTextEditInsert*

```
#include <Xol/textbuff.h>
Boolean OlTextEditInsert(
 TextEditWidget ctx,
 String buffer,
 int length);
```

`OlTextEditInsert()` inserts a NULL-terminated *buffer* containing *length* bytes in the `TextBuffer` associated with the `TextEdit` widget *ctx*. The inserted text replaces the current (if any) selection. The value of *length* is not used internally, but is passed on as the `length` field in the `XtNmodifyVerification` callback. The function returns `FALSE` if the widget supplied is not a `TextEdit` widget or if the insert operation fails; otherwise, it returns `TRUE`.

### *OlTextEditUpdate*

```
#include <Xol/textbuff.h>
Boolean OlTextEditUpdate(
 TextEditWidget ctx,
 Boolean state);
```

`OlTextEditUpdate()` sets the *updateState* of a `TextEdit` widget. Setting the state to `FALSE` turns screen update off; setting the state to `TRUE` turns screen updates on and refreshes the display. The function returns `FALSE` if the widget supplied is not a `TextEdit` widget; otherwise, it returns `TRUE`.

### *OlTextEditPaste*

```
#include <Xol/textbuff.h>
Boolean OlTextEditPaste(
 TextEditWidget ctx);
```

`OlTextEditPaste()` pastes the contents of the `CLIPBOARD` into the `TextEdit` widget *ctx*. The current (if any) selection is replaced by the contents of the `CLIPBOARD`. The function returns `FALSE` if the widget supplied is not a `TextEdit` widget; otherwise, it returns `TRUE`.

### *See Also*

Regular Expression Functions on page 161,  
“Text Buffer Functions” on page 163,  
“Text Buffer Functions for Internationalization” on page 176.

## TextField Widget

**Note** – The TextField widget is obsolete but remains in the toolkit for backward compatibility. Its functionality has been superseded by the TextLine widget, which provides a more efficient implementation of a one-line text field. See “TextLine Widget” on page 688 for more information.

### Class

**Class Name:** TextField  
**Class Pointer:** textFieldWidgetClass

### Ancestry

Core-Composite-Constraint-Manager-TextField

### Required Header Files

```
#include <Xol/OpenLook>
#include <Xol/TextField.h>
```

### Description

#### Components

A TextField widget is a one-line input field for text data, as shown in the following diagram.

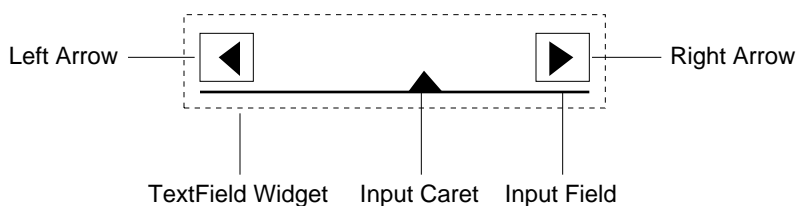


Figure 10-3 TextField Components

---

***TextField Widget***

The input field and input caret are always present; however, the left and right arrows only appear when the value of the `TextField` is larger than the input field can display. These arrows provide the user the ability to scroll the `TextField` value.

***Subwidget***

The `TextField` widget creates a one-line `TextEdit` widget to handle the text editing. If the application needs to configure certain `TextEdit` resources (i.e., `XtNcursorPosition`), it can access the handle to the `TextEdit` subwidget through the `XtNtextEditWidget` resource. For more information on the `TextEdit` widget, see “`TextEdit Widget`” on page 623.

***Keyboard Input***

Once the input focus has been moved to the Input Field, keyboard entry is allowed. The `TextField` widget does not validate the input, leaving that up to the application.

***Coloration***

For 3D and 2D, `XtNfontColor` is used to draw the `TextField`'s text and `XtNinputFocusColor` is used to draw the active caret.

For 3D, the `TextField` underline and scrollbar coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.” `XtNbackground` is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, `XtNbackground` and `XtNforeground` are used to render the `TextField`'s underline and scrollbars as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, “Controls.”

***Keyboard Traversal***

The default value of the `XtNtraversalOn` resource is `TRUE`.

The `TextField` widget responds to the following Activation Types for keyboard navigation. For more information on these Activation Types and their key bindings, see Table 10-9 on page 675.

*TextField Widget*

|              |              |              |
|--------------|--------------|--------------|
| OL_CHARBAK   | OL_MENU      | OL_PREVFIELD |
| OL_CHARFWD   | OL_MENUKEY   | OL_WORDBAK   |
| OL_LINEEND   | OL_NEXTFIELD | OL_WORDFWD   |
| OL_LINESTART |              |              |

***Keyboard Mnemonic Display***

The TextField does not display the mnemonic. If the TextField is the child of a Caption widget, the Caption widget can be used to display the mnemonic.

***Keyboard Accelerator Display***

The TextField does not respond to a keyboard accelerator because clicking the SELECT button on a TextField activates depending on the pointer position. So, the TextField does not display a keyboard accelerator.

***Text Selection***

The TextField widget responds to the following Activation Types for selection. For more information on these Activation Types and their key bindings, see Table 10-9 on page 675.

|              |                |               |
|--------------|----------------|---------------|
| OL_ADJUST    | OL_SELCHARBAK  | OL_SELLINE    |
| OL_ADJUSTKEY | OL_SELCHARFWD  | OL_SELLINEBAK |
| OL_COPY      | OL_SELECT      | OL_SELLINEFWD |
| OL_CUT       | OL_SELECTKEY   | OL_SELWORDBAK |
| OL_PASTE     | OL_SELFLIPENDS | OL_SELWORDFWD |

***Scrolling***

The TextField widget responds to the following Activation Types for scrolling. For more information on these Activation Types and their key bindings, see Table 10-9 on page 675.

|                    |               |                   |
|--------------------|---------------|-------------------|
| OL_SCROLLRIGHT     | OL_SCROLLLEFT | OL_SCROLLLEFTEDGE |
| OL_SCROLLRIGHTEDGE |               |                   |

***Editing***

The TextField widget responds to the following Activation Types for editing. For more information on these Activation Types and their key bindings, see Table 10-9 on page 675.

*TextField Widget*

|               |               |               |
|---------------|---------------|---------------|
| OL_DELCHARBAK | OL_DELLINEBAK | OL_DELWORDFWD |
| OL_DELCHARFWD | OL_DELLINEFWD | OL_RETURN     |
| OL_DELLINE    | OL_DELWORDBAK | OL_UNDO       |

*Scrolling Long Text Input*

If an input value exceeds the length of the Input Field, the Left Arrow and/or Right Arrow appear and the input value is visually truncated on the left and/or the right to show only as many characters as can fit in the Input Field. The truncation is at a character boundary. Since the Arrows take up space that would otherwise be used for the input, the truncation is more severe than would be necessary if they were not visible. An Arrow is present only if characters are hidden in the direction expressed by the arrow.

The user can scroll to show the hidden parts of the input by:

- Clicking or pressing SELECT on the Left or Right Arrow.
- Clicking SELECT on the Left Arrow scrolls the input one character to the right to show the next character that was hidden to the left.
- Clicking SELECT on the Right Arrow scrolls the input one character to the left to show the next character that was hidden to the right.
- Pressing SELECT scrolls continuously, with a user-adjustable wait between changes.

The text does not scroll beyond its limits, so that the left-most character never moves beyond the right edge of the TextField widget and the rightmost character never moves beyond the left edge.

- If the user attempts to scroll beyond the limits by clicking SELECT, the system beeps.
- If the user is pressing SELECT when the limit is reached, the text stops scrolling but the system does not beep.
- If the user releases SELECT and then presses it again to exceed the scrolling limit, the system beeps once regardless of how long SELECT is pressed.

*Input Validation*

A validation callback list can be used to perform limited per-field validation, such as when the user presses the RETURN, PREVFIELD, or NEXTFIELD keys. It is not called if the user moves the focus to another input area using the mouse.

### Caret Position

As characters are entered from the keyboard, the input caret moves to the right until it reaches the right end of the input field. As additional characters are typed the text scrolls to the left (the left arrow appears as discussed above) and the input caret moves relative to the text but remains stationary on the screen.

### Selecting and Operating on the Input Field

The TextField widget allows text to be copied or moved to and from the input field. See “Text Selection Operations” on page 204 for the description of these operations.

## Resources

Table 10-5 TextField Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent's)          | SGI    |
| XtNdepth             | int              | (parent's)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | (calculated)        | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent's)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | (calculated)        | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

*TextField Widget*

*Table 10-6 TextField Composite Resources*

| <b>Name</b>       | <b>Type</b> | <b>Default</b> | <b>Access</b> |
|-------------------|-------------|----------------|---------------|
| XtNchildren       | WidgetList  | NULL           | G             |
| XtNinsertPosition | XtOrderProc | NULL           | SGI           |
| XtNnumChildren    | Cardinal    | 0              | G             |

*Table 10-7 TextField Manager Resources*

| <b>Name</b>          | <b>Type</b>    | <b>Default</b> | <b>Access</b> |
|----------------------|----------------|----------------|---------------|
| XtNconsumeEvent      | XtCallbackList | NULL           | SGIO          |
| XtNinputFocusColor   | Pixel          | Red            | SGID          |
| XtNreferenceName     | String         | NULL           | GI            |
| XtNreferenceWidget   | Widget         | NULL           | GI            |
| XtNtraversalOn       | Boolean        | TRUE           | SGI           |
| XtNunrealizeCallback | XtCallbackList | NULL           | SGIO          |
| XtNuserData          | XtPointer      | NULL           | SGI           |

*Table 10-8 TextField Resources*

| <b>Name</b>       | <b>Type</b>     | <b>Default</b>      | <b>Access</b> |
|-------------------|-----------------|---------------------|---------------|
| XtNcharsVisible   | int             | (calculated)        | GI            |
| XtNeditType       | OIEditType      | OL_TEXT_EDIT        | SGI           |
| XtNfont           | OIFont          | XtDefaultFont       | SGI           |
| XtNfontColor      | Pixel           | XtDefaultForeground | SGID          |
| XtNimPreeditStyle | OImPreeditStyle | OL_NO_PREEDIT       | GI            |
| XtNinitialDelay   | int             | 500                 | SGI           |
| XtNinsertTab      | Boolean         | FALSE               | SGI           |
| XtNmaximumSize    | int             | 0                   | SGI           |
| XtNrepeatRate     | int             | 100                 | SGI           |
| XtNscale          | int             | 12                  | SGI           |
| XtNstring         | OIStr           | NULL                | SGI           |
| XtNtextEditWidget | Widget          | NULL                | G             |
| XtNtextFormat     | OIStrRep        | OL_SB_STR_REP       | GI            |
| XtNverification   | XtCallbackList  | NULL                | SGIO          |



**XtNcharsVisible**

| Class           | Type | Default      | Access |
|-----------------|------|--------------|--------|
| XtCCharsVisible | int  | (calculated) | GI     |

Synopsis: The initial width of the list in terms of characters.

This resource overrides the `XtNwidth` resource setting. The `XtNwidth` is recalculated to be the value of the average font width plus the values for the internal left and right margins. The value of this resource changes to reflect the effects of geometry changes imposed by the widget tree and the user. Calls to `XtSetValues()` for this resource are ignored.

**XtNeditType**

| Class       | Type       | Default      | Access |
|-------------|------------|--------------|--------|
| XtCEditType | OLEditMode | OL_TEXT_EDIT | SGI    |

Synopsis: The edit state of the source.

Values: `OL_TEXT_READ/"textread"` - The source is read-only; the user cannot edit it.  
`OL_TEXT_EDIT/"textedit"` - The source is fully editable.

**XtNfont**

| Class   | Type   | Default       | Access |
|---------|--------|---------------|--------|
| XtCFont | OLFont | XtDefaultFont | SGI    |

The TextField widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfont" on page 26.

**XtNfontColor**

| Class        | Type  | Default             | Access |
|--------------|-------|---------------------|--------|
| XtCFontColor | Pixel | XtDefaultForeground | SGID   |

The TextField widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See "XtNfontColor" on page 27.

**XtNimPreeditStyle**

| Class             | Type             | Default       | Access |
|-------------------|------------------|---------------|--------|
| XtCImPreeditStyle | OLImPreeditStyle | OL_NO_PREEDIT | GI     |

Synopsis: The pre-edit style (in conjunction with the shell's

*TextField Widget*

XtNimStatusStyle resource). If the pre-edit style is not supported by the input method, the ability to pre-edit is lost.

Values:

- OL\_ON\_THE\_SPOT/"onTheSpot" - The pre-edit data is displayed at the insertion point in the application window. The preexisting user data is shifted and the pre-edit data is inserted at the point of insertion.
- OL\_OVER\_THE\_SPOT/"overTheSpot" - The pre-edit data is displayed in the application window, starting at the insertion point. As the user types the pre-edit data, the preexisting user data is obscured by the pre-edit data.
- OL\_ROOT\_WINDOW/"rootWindow" - The pre-edit data is displayed in a child of the root window, away from the point of insertion.
- OL\_NO\_PREEDIT/"none" - No pre-edit data is displayed.

See "XtNimStatusStyle" on page 44 and "Setting the Input Method Pre-Edit and Status Styles (Asian Locales Only)" on page 82.

***XtNinitialDelay***

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCInitialDelay | int  | 500     | SGI    |

Synopsis: The number of milliseconds from the time the scrolling arrows are pressed until the repeated scrolling starts.

***XtNinsertTab***

| Class        | Type    | Default | Access |
|--------------|---------|---------|--------|
| XtCInsertTab | Boolean | FALSE   | SGI    |

Synopsis: Determines whether a tab character is insertable into the text.

Values:

- TRUE/"true" - Tabs are insertable but not used for forward traversal. Forward traversal can still be accomplished with Control-Tab.
- FALSE/"false" - A tab is not insertable.

Setting this resource to FALSE (the default) makes traversal of the controls easier if the tab key is bound as OL\_NEXTFIELD.

***XtNmaximumSize***

| Class     | Type | Default | Access |
|-----------|------|---------|--------|
| XtCLength | int  | 0       | SGI    |

Synopsis: The maximum number of characters that can be entered into the internal buffer.

Values:  $0 \leq \text{XtNmaximumSize}$

If this value is not set or is zero, the internal buffer will increase its size as needed, limited only by memory limitations of the process.

***XtNrepeatRate***

| Class         | Type | Default | Access |
|---------------|------|---------|--------|
| XtCRepeatRate | int  | 100     | SGI    |

Synopsis: The time in milliseconds between repeats when the scrolling arrows are pressed for more than `XtNinitialDelay` milliseconds.

***XtNscale***

| Class    | Type | Default | Access |
|----------|------|---------|--------|
| XtCScale | int  | 12      | SGI    |

The TextField widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNscale” on page 29.

***XtNstring***

| Class     | Type  | Default | Access |
|-----------|-------|---------|--------|
| XtCString | OlStr | NULL    | SGI    |

Synopsis: The content of the Input Field. On being set, a copy of the value is made in an internal buffer.

Values: Any `OlStr` value valid in the current locale.

Using `XtGetValues()` on this resource gets a new copy that the application is responsible for freeing when no longer needed.

The TextField function `OlTextFieldGetOlString()` can also be used to get a copy of the `XtNstring` value; see page 686.

*TextField* Widget

***XtNtextEditWidget***

| Class             | Type   | Default | Access |
|-------------------|--------|---------|--------|
| XtCTextEditWidget | Widget | NULL    | G      |

Synopsis: The TextEdit widget managed by the TextField.

This value can be used to directly access the underlying TextEdit widget (and its TextBuffer) used to manage the textual display.

***XtNtextFormat***

| Class         | Type     | Default       | Access |
|---------------|----------|---------------|--------|
| XtCTextFormat | OlStrRep | OL_SB_STR_REP | GI     |

The TextField widget supports this resource in the same manner as a widget that would inherit it from the Primitive class. See “XtNtextFormat” on page 29.

***XtNverification***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the user presses the RETURN, PREVFIELD, or NEXTFIELD keys out of the TextField widget.

The *call\_data* parameter is a pointer to a OlTextFieldVerify structure:

```
typedef struct _OlTextFieldVerify {
 String string;
 Boolean ok;
 OlTextVerifyReason reason;
} OlTextFieldVerify;
```

**string** A pointer to the content of the text field. It is not a copy but a pointer to an internal buffer. The application should copy the buffer if it needs to keep the data intact longer than the duration of the callback.

**ok** Currently unused.

**reason** One of the following constants:

|                     |                   |
|---------------------|-------------------|
| OlTextFieldReturn   | RETURN entered    |
| OlTextFieldPrevious | PREVFIELD entered |
| OlTextFieldNext     | NEXTFIELD entered |

## Activation Types

The following table lists the activation types used by the TextField.

Table 10-9 TextField Activation Types

| Activation Type   | Semantics       | Resource Name        |
|-------------------|-----------------|----------------------|
| OL_ADJUST         | ADJUST          | XtNadjustBtn         |
| OL_ADJUSTKEY      | ADJUST          | XtNadjustKey         |
| OL_CANCEL         | CANCEL          | XtNcancelKey         |
| OL_CHARBAK        | LEFT            | XtNleftKey           |
| OL_CHARFWD        | RIGHT           | XtNrightKey          |
| OL_COPY           | COPY            | XtNcopyBtn           |
| OL_CUT            | CUT             | XtNcutBtn            |
| OL_DEFAULTACTION  | DEFAULTACTION   | XtNdefaultActionKey  |
| OL_DELCHARBAK     | DELETE BACKWARD | XtNdelCharBakFwd     |
| OL_DELCHARFWD     | DELETE FORWARD  | XtNdelCharFwdKey     |
| OL_DELLINE        | DELETE LINE     | XtNdelLineKey        |
| OL_DELLINEBAK     | DELLINEBAK      | XtNdelLineBakKey     |
| OL_DELLINEFWD     | DELLINEFWD      | XtNdelLineFwdKey     |
| OL_DELWORDBAK     | DELWORDBAK      | XtNdelWordBakKey     |
| OL_DELWORDFWD     | DELWORDFWD      | XtNdelWordFwdKey     |
| OL_HELP           | HELP            | XtNhelpKey           |
| OL_LINEEND        | ROW END         | XtNlineEndKey        |
| OL_LINESTART      | ROW START       | XtNlineStartKey      |
| OL_MENU           | MENU            | XtNmenuBtn           |
| OL_MENUKEY        | MENU            | XtNmenuKey           |
| OL_MOVEDOWN       | MOVEDOWN        | XtNdownKey           |
| OL_MOVELEFT       | MOVELEFT        | XtNleftKey           |
| OL_MOVERIGHT      | MOVERIGHT       | XtNrightKey          |
| OL_MOVEUP         | MOVEUP          | XtNupKey             |
| OL_NEXTFIELD      | NEXTFIELD       | XtNnextFieldKey      |
| OL_PASTE          | PASTE           | XtNpasteBtn          |
| OL_PREVFIELD      | PREVFIELD       | XtNprevFieldKey      |
| OL_RETURN         | RETURN          | XtNreturnKey         |
| OL_SCROLLLEFT     | SCROLLLEFT      | XtNscrollLeftKey     |
| OL_SCROLLLEFTEDGE | SCROLLLEFTEDGE  | XtNscrollLeftEdgeKey |

Table 10-9 TextField Activation Types (Continued)

| Activation Type    | Semantics       | Resource Name         |
|--------------------|-----------------|-----------------------|
| OL_SCROLLRIGHT     | SCROLLRIGHT     | XtNscrollRightKey     |
| OL_SCROLLRIGHTEDGE | SCROLLRIGHTEDGE | XtNscrollRightEdgeKey |
| OL_SELCHARBAK      | SELCHARBAK      | XtNselCharBakKey      |
| OL_SELCHARFWD      | SELCHARFWD      | XtNselCharFwdKey      |
| OL_SELECT          | SELECT          | XtNselectBtn          |
| OL_SELECTKEY       | SELECT          | XtNselectKey          |
| OL_SELFLIPENDS     | SELFLIPENDS     | XtNselFlipEndsKey     |
| OL_SELLINE         | SELLINE         | XtNselLineKey         |
| OL_SELLINEBAK      | SELLINEBAK      | XtNselLineBakKey      |
| OL_SELLINEFWD      | SELLINEFWD      | XtNselLineFwdKey      |
| OL_SELWORDBAK      | SELWORDBAK      | XtNselWordBakKey      |
| OL_SELWORDFWD      | SELWORDFWD      | XtNselWordFwdKey      |
| OL_TOGGLEPUSHPIN   | TOGGLEPUSHPIN   | XtNtogglePushpinKey   |
| OL_UNDO            | UNDO            | XtNundoKey            |
| OL_WORDBAK         | JUMP LEFT       | XtNwordBakKey         |
| OL_WORDFWD         | JUMP RIGHT      | XtNwordFwdKey         |

Activation types not described in the following table are described in “Common Activation Types” on page 68.

**OL\_ADJUST**

The OL\_ADJUST activation type calls the XtNmotionVerification callback list with the appropriate OlTextMotionCallData structure containing the new cursor position, selection start, and selection end. If a callback is not registered or the OlTextMotionCallData’s ok field is TRUE, the OL\_ADJUST activation type will modify the XtNcursorPosition, XtNselectStart, and XtNselectEnd resources to reflect the cursor position and the resulting selection.

**OL\_CHARBAK**

The OL\_CHARBAK activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the cursor position moved one character before the current cursor position. If there is no XtNmotionVerification callback or if the ok field of the OlTextMotionCallData structure is TRUE, then the cursor will be moved.

***OL\_CHARFWD***

The `OL_CHARFWD` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved one character after the current cursor position. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

***OL\_COPY***

The `OL_COPY` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure containing the currently selected items. If the `OlTextModifyCallData` `ok` field is `TRUE` upon return, then the selected text will be copied to the CLIPBOARD.

***OL\_CUT***

The `OL_CUT` activation type applies only to `TextField` widgets that have `XtNeditType` of `OL_TEXT_EDIT`. The `XtNmodifyVerification` callback list will be called with an `OlTextModifyCallData` structure containing the currently selected items. If the `OlTextModifyCallData` `ok` field is `TRUE` upon return, then the selected text will be copied to the CLIPBOARD.

***OL\_DELCHARBAK***

The `OL_DELCHARBAK` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the character to the left of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the character has been deleted. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELCHARFWD***

The `OL_DELCHARFWD` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the character to the right of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the character has been deleted. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELLINE***

The `OL_DELLINE` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the current line has been deleted. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the line has been deleted. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELLINEBAK***

The `OL_DELLINEBAK` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the line to the left of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification`



---

*TextField Widget*

callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the line has been deleted. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELLINEFWD***

The `OL_DELLINEFWD` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the line to the right of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the line has been deleted. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELWORDBAK***

The `OL_DELWORDBAK` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the word to the left of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the word has been deleted. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_DELWORDFWD***

The `OL_DELWORDFWD` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text after the word to the right of the cursor has been deleted. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position after the word has been deleted. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_LINEEND***

The `OL_LINEEND` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved to after the last character on the current line. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

***OL\_LINESTART***

The `OL_LINESTART` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved to before the first character on the current line. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

***OL\_MENU/  
OL\_MENUKEY***

The `OL_MENU` and `OL_MENUKEY` activation types will pop up the `TextField` menu.

***OL\_PASTE***

The `OL_PASTE` activation type applies only to `TextField` widgets that have `XtNeditType` of `OL_TEXT_EDIT`. When the widget has focus, then the `OL_PASTE` activation type will insert the contents of the `CLIPBOARD` at the current insert position by calling the `XtNpostModifyVerification` callback list. The `XtNmotionVerification` callback list will be called with an `OlTextMotionCallData` structure that represents the new cursor and selection position.

***OL\_RETURN***

This activation type calls the `XtNverification` callback list with the appropriate `OlTextFieldVerify` structure.

***OL\_SCROLLLEFT***

The `OL_SCROLLLEFT` activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the character left of the current view into the view.

***OL\_SCROLLLEFTEDGE***

The `OL_SCROLLLEFTEDGE` activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the leftmost character of the document into the view.

***OL\_SCROLLRIGHT***

The `OL_SCROLLRIGHT` activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the character right of the current view into the view.

***OL\_SCROLLRIGHTEDGE***

The `OL_SCROLLRIGHTEDGE` activation type calls the `XtNmargin` callback list with the appropriate `OlTextMarginCallData` structure to bring the rightmost character of the document into the view.

***OL\_SELCHARBAK***

The `OL_SELCHARBAK` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended one character to the left of the cursor position and the cursor position moved to before the selection. If there is no

XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the selection will be extended, cursor will be moved, and the text may be scrolled to bring the previous line into view. The XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure.

### ***OL\_SELCHARFWD***

The OL\_SELCHARFWD activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the selection extended one character to the right of the cursor position and the cursor position moved to after the selection. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the selection will be extended, cursor will be moved, and the text may be scrolled to bring the next line into view. The XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure.

### ***OL\_SELECT***

The OL\_SELECT activation type calls the XtNmotionVerification callback list with the appropriate OlTextMotionCallData structure containing the new cursor position, selection start, and selection end. If a callback is not registered or the OlTextMotionCallData's *ok* field is TRUE, the OL\_SELECT activation type will modify the XtNcursorPosition, XtNselectStart, and XtNselectEnd resources to reflect the cursor position and the resulting selection.

### ***OL\_SELFLIPENDS***

The OL\_SELFLIPENDS activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the cursor position moved to the opposite end of the current selection. If there is no XtNmotionVerification callback or if the *ok* field of the OlTextMotionCallData structure is TRUE, then the cursor will be moved. If the text is scrolled to bring the cursor into view, the XtNmargin callback list will be called with the appropriate OlTextMarginCallData structure.

### ***OL\_SELLINE***

The OL\_SELLINE activation type calls the XtNmotionVerification callback list with an OlTextMotionCallData structure that represents the selection extended to the beginning of the current line and to the end of the current line.

and the cursor position moved to before the selection. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_SELLINEBAK***

The `OL_SELLINEBAK` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the beginning of the line to the left of the current cursor position and the cursor position moved to before the selection. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, cursor will be moved, and the text may be scrolled to bring the previous line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_SELLINEFWD***

The `OL_SELLINEFWD` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the end of the line to the right of the current cursor position and the cursor position moved to after the selection. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, cursor will be moved, and the text may be scrolled to bring the next line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

### ***OL\_SELWORDBAK***

The `OL_SELWORDBAK` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the beginning of the word to the left of the current cursor position and the cursor position moved to before the selection. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, cursor will be moved, and the text may be scrolled to bring the previous line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

***OL\_SELWORDFWD***

The `OL_SELWORDFWD` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection extended to the end of the word to the right of the current cursor position and the cursor position moved to after the selection. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the selection will be extended, cursor will be moved, and the text may be scrolled to bring the next line into view. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure.

***OL\_UNDO***

The `OL_UNDO` activation type calls the `XtNmodifyVerification` callback list with an `OlTextModifyCallData` structure that represents the selection, cursor position, and text before the last modification. If there is no `XtNmodifyVerification` callback or if the `ok` field of the `OlTextModifyCallData` structure is `TRUE`, then the text will be updated. Then the activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the selection and cursor position before the last modification. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor and selection will be updated. The `XtNmargin` callback list will be called with the appropriate `OlTextMarginCallData` structure. Finally, the `XtNpostModifyNotification` callback list will be called with the appropriate `OlTextPostModifyCallData` structure.

***OL\_WORDBAK***

The `OL_WORDBAK` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved one word before the current cursor position. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

***OL\_WORDFWD***

The `OL_WORDFWD` activation type calls the `XtNmotionVerification` callback list with an `OlTextMotionCallData` structure that represents the cursor position moved one word after the current cursor position. If there is no `XtNmotionVerification` callback or if the `ok` field of the `OlTextMotionCallData` structure is `TRUE`, then the cursor will be moved.

*See Also*

“StaticText Widget” on page 600,  
“TextEdit Widget” on page 623,  
“TextEdit Functions” on page 660,  
“TextField Functions” on page 686,  
“Text Buffer Functions” on page 163,  
“Text Selection Operations” on page 204.

### TextField Functions

The following functions assist in manipulating TextField widgets.

#### *OlTextFieldCopyOlString*

```
int OlTextFieldCopyOlString(
 TextFieldWidget tfw,
 OlStr string);
```

`OlTextFieldCopyOlString()` copies the `OlStr` string associated with the `TextField` widget `tfw` into the user-supplied area pointed to by `string`. The function returns the length of this string in number of bytes (if the text format is single-byte or multibyte) or in the number of wide characters (if the text format is wide character).

#### *OlTextFieldCopyString*

```
#include <textbuff.h>

int OlTextFieldCopyString(
 TextFieldWidget tfw,
 char *string);
```

`OlTextFieldCopyString()` copies the string associated with the `TextField` widget `tfw` into the user supplied area pointed to by `string` and returns the length of this string.

---

**Note** – This function is superseded by `OlTextFieldCopyOlString()`. However, it is safe to use `OlTextFieldCopyString()` if the text format of the widget is single-byte.

---

#### *OlTextFieldGetOlString*

```
OlStr OlTextFieldGetOlString(
 TextFieldWidget tfw,
 int *size);
```

`OlTextFieldGetOlString()` retrieves a new copy of the `OlStr` associated with the `TextField` widget `tfw`. The function returns a pointer to the newly allocated `OlStr` copy.



---

## TextField Functions

Optionally, if *size* is not NULL, `OlTextFieldGetOlString()` returns in *size* the length of the string in number of bytes (if the text format is single-byte or multibyte) or in number of wide characters (if the text format is wide character).

### *OlTextFieldGetString*

```
#include <textbuff.h>
char *OlTextFieldGetString(
 TextFieldWidget tfw,
 int *size);
```

`OlTextFieldGetString()` retrieves a new copy of the string associated with the `TextField` widget *tfw* and returns a pointer to the newly allocated string copy. Optionally, if *size* is not NULL, the function returns in *size* the length of the string, including the null terminator.

---

**Note** – The storage for the copy is allocated by this routine. It is the responsibility of the caller to free this storage when it becomes dispensable.

---

**Note** – This function is superseded by `OlTextFieldGetString()`. However, it is safe to use `OlTextFieldGetString()` if the text format of the widget is single-byte.

---

### *See Also*

Regular Expression Functions on page 161.

## *TextLine Widget*

### *TextLine Widget*

#### *Class*

*Class Name:* TextLine  
*Class Pointer:* textLineWidgetClass

#### *Ancestry*

Core-Primitive-TextLine

#### *Required Header Files*

```
#include <Xol/OpenLook>
#include <Xol/TextLine.h>
```

#### *Description*

The TextLine widget is a one-line input field for text data. Once the input focus is moved into the widget, keyboard entry is allowed. If the input value exceeds the length of the input field, the scroll buttons appear. Hidden text can then be scrolled into view by pressing the scroll buttons. Pressing the buttons continuously scrolls the text repeatedly with a user-adjustable delay.

#### *Components*

The TextLine contains the following graphical elements:

- Right-justified bold label at the left of the TextLine
- Input field
- Input Caret (not present in ReadOnly mode)
- 1-point (for Mono) or chiseled underline (not present in ReadOnly mode)
- Optional scroll buttons

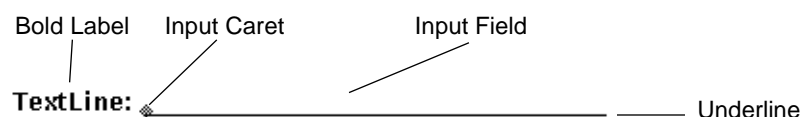


Figure 10-4 **TextLine Components**

### ***Keyboard Traversal***

The TextLine allows keyboard entry if it is sensitive and it has the keyboard focus. However, in ReadOnly mode, the widget is not traversable and it does not receive keyboard focus. Selection is also turned off during ReadOnly mode.

The TextLine responds to the following keyboard navigation keys:

- NEXTFIELD moves to the next traversable widget in the window
- PREVFIELD moves to the previous traversable widget in the window
- CHARFWD moves the caret forward one character
- CHARBAK moves the caret backward one character
- WORDFWD moves the caret forward one word
- WORDBAK moves the caret back one word
- LINESTART moves the caret to the beginning of the display
- LINEEND moves the caret to the end of the display
- MENUKEY posts the menu associated with the TextLine

The TextLine responds to the following edit keys:

- DELCHARFWD deletes the character to the right of the caret
- DELCHARBAK deletes the character to the left of the caret
- DELWORDFWD deletes the word to the right of the caret
- DELWORDBAK deletes the word to the left of the caret
- DELLINEFWD deletes to the end of the line from the caret
- DELLINEBAK deletes from the beginning of the line to the caret
- DELLINE deletes the line containing the caret
- UNDO undoes the last edit

### ***Keyboard Mnemonic Display***

The TextLine does not display any mnemonic. If the TextLine is the child of a Caption widget, the Caption can be used to display any mnemonic.

***Keyboard Accelerator Display***

The TextLine does not respond to any keyboard accelerators.

***Display of Text***

The TextLine displays its contents in the font specified by the `XtNfont` resource. If the length of the text exceeds the length of the input field, the widget sets up the Left or Right or both scroll buttons to indicate this. The text is then visually truncated at the boundaries to show only as many characters as can fit in the input field. The truncation is always at a character boundary. A scroll button is present only if characters are hidden in the direction indicated by that button. The user can scroll to show the hidden parts of the text by clicking or pressing the scroll buttons. Clicking SELECT on any scroll button will scroll the text one character in the direction indicated by that button. Pressing SELECT on any scroll button will repeat the scrolling with a user-adjustable delay between each scroll.

***Caret Position***

As characters are entered from the keyboard, the caret moves to the right until it reaches the right end of the input field. As additional characters are typed, the text jump-scrolls to the left by a specific amount. Note that the TextLine always keeps the cursor visible. Thus, the presence or absence of either of the scroll buttons is controlled by the current cursor position.

***Selection of Text***

Text selection can be done by the user by using the mouse or keyboard. The widget also provides a set of convenience functions to manipulate the selection programmatically; see “TextLine Functions” on page 708.

***Coloration***

For 3D and 2D, `XtNfontColor` is used to draw the TextLine’s text and `XtNinputFocusColor` is used to draw the active caret.

*TextLine Widget*

For 3D, the TextLine underline and scrollbar coloration is defined by the *OPEN LOOK GUI Functional Specification*, Chapter 9, “Color and Three-Dimensional Design.” XtNbackground is used for BG1, and the BG2 (pressed-in), BG3 (shadow), and Highlight colors are derived by the toolkit from BG1.

For 2D, XtNbackground and XtNforeground are used to render the TextLine’s underline and scrollbars as described by the *OPEN LOOK GUI Functional Specification*, Chapter 4, “Controls.”

**Known Deficiencies**

The TextLine widget currently does not support the *implicit commit* feature; see “Input Method” on page 80. This could be a deficiency in Asian locales. The workaround is to use the TextField widget (page 665), which does support it.

**Resources***Table 10-10*TextLine Core Resources

| Name                 | Type             | Default             | Access |
|----------------------|------------------|---------------------|--------|
| XtNaccelerators      | AcceleratorTable | NULL                | SGI    |
| XtNancestorSensitive | Boolean          | TRUE                | G      |
| XtNbackground        | Pixel            | XtDefaultBackground | SGID   |
| XtNbackgroundPixmap  | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderColor       | Pixel            | XtDefaultForeground | SGID   |
| XtNborderPixmap      | Pixmap           | XtUnspecifiedPixmap | SGI    |
| XtNborderWidth       | Dimension        | 1                   | SGI    |
| XtNcolormap          | Colormap         | (parent’s)          | SGI    |
| XtNdepth             | int              | (parent’s)          | GI     |
| XtNdestroyCallback   | XtCallbackList   | NULL                | SGIO   |
| XtNheight            | Dimension        | (calculated)        | SGI    |
| XtNmappedWhenManaged | Boolean          | TRUE                | SGI    |
| XtNscreen            | Screen *         | (parent’s)          | G      |
| XtNsensitive         | Boolean          | TRUE                | GIO    |
| XtNtranslations      | XtTranslations   | NULL                | SGI    |
| XtNwidth             | Dimension        | (calculated)        | SGI    |
| XtNx                 | Position         | 0                   | SGI    |
| XtNy                 | Position         | 0                   | SGI    |

*TextLine Widget*

*Table 10-11*TextLine Primitive Resources

| <b>Name</b>        | <b>Type</b>    | <b>Default</b>            | <b>Access</b> |
|--------------------|----------------|---------------------------|---------------|
| XtNaccelerator     | String         | NULL                      | n/a           |
| XtNacceleratorText | String         | NULL                      | n/a           |
| XtNconsumeEvent    | XtCallbackList | NULL                      | SGIO          |
| XtNfont            | OlFont         | XtDefaultFont             | SGI           |
| XtNfontColor       | Pixel          | XtDefaultForeground       | SGID          |
| XtNforeground      | Pixel          | XtDefaultForeground       | SGID          |
| XtNinputFocusColor | Pixel          | (calculated; see page 27) | SGID          |
| XtNmnemonic        | unsigned char  | '\0'                      | n/a           |
| XtNreferenceName   | String         | NULL                      | GI            |
| XtNreferenceWidget | Widget         | NULL                      | GI            |
| XtNscale           | int            | 12                        | SGI           |
| XtNtextFormat      | OlStrRep       | OL_SB_STR_REP             | GI            |
| XtNtraversalOn     | Boolean        | TRUE                      | SGI           |
| XtNuserData        | XtPointer      | NULL                      | SGI           |

*Table 10-12*TextLine Resources

| <b>Name</b>                      | <b>Type</b>      | <b>Default</b>    | <b>Access</b> |
|----------------------------------|------------------|-------------------|---------------|
| XtNblinkRate                     | int              | 1000              | SGI           |
| XtNcaptionAlignment <sup>1</sup> | OlDefine         | OL_CENTER         | G             |
| XtNcaptionFont <sup>1</sup>      | OlFont           | OlDefaultBoldFont | SI            |
| XtNcaptionLabel <sup>1</sup>     | OlStr            | NULL              | SGI           |
| XtNcaptionPosition <sup>1</sup>  | OlDefine         | OL_LEFT           | G             |
| XtNcaptionSpace <sup>1</sup>     | Dimension        | 4                 | G             |
| XtNcaptionWidth <sup>1</sup>     | Dimension        | 0                 | G             |
| XtNcharsVisible                  | int              | 0                 | GI            |
| XtNcommitCallback                | XtCallbackList   | NULL              | SGIO          |
| XtNcursorPosition                | int              | 0                 | SGI           |
| XtNeditType                      | OlDefine         | OL_TEXT_EDIT      | SGI           |
| XtNimPreeditStyle                | OlImPreeditStyle | OL_NO_PREEDIT     | GI            |
| XtNinitialDelay                  | int              | 500               | SGI           |
| XtNinsertTab                     | Boolean          | FALSE             | SGI           |

Table 10-12 TextLine Resources (Continued)

| Name                  | Type           | Default   | Access |
|-----------------------|----------------|-----------|--------|
| XtNmaximumChars       | int            | 0         | GI     |
| XtNmenu               | Widget         | (special) | GI     |
| XtNmotionCallback     | XtCallbackList | NULL      | SGIO   |
| XtNpostModifyCallback | XtCallbackList | NULL      | SGIO   |
| XtNpreModifyCallback  | XtCallbackList | NULL      | SGIO   |
| XtNrepeatRate         | int            | 100       | SGI    |
| XtNstring             | OlStr          | NULL      | SGI    |
| XtNunderline          | Boolean        | TRUE      | SGI    |
| XtNupdateDisplay      | Boolean        | TRUE      | SGI    |

1. These resources are provided to support captions. They are subject to change in a future OLIT release. Note that they cannot be set.

**XtNblinkRate**

| Class        | Type | Default | Access |
|--------------|------|---------|--------|
| XtCblinkRate | int  | 1000    | SGI    |

Synopsis: The blink rate of the active cursor in terms of milliseconds. A value of zero turns blinking off.

Values: Any integer

**XtNcaptionAlignment**

| Class               | Type     | Default   | Access |
|---------------------|----------|-----------|--------|
| XtCCaptionAlignment | OlDefine | OL_CENTER | G      |

Synopsis: The alignment of the caption with respect to the text area.

**Note** – This resource cannot be set and is subject to change in future revisions.

**XtNcaptionFont**

| Class          | Type   | Default           | Access |
|----------------|--------|-------------------|--------|
| XtCCaptionFont | OlFont | OlDefaultBoldFont | SI     |

Synopsis: The font for the caption label.

Values: Any font valid in the current locale

**Note** – This resource cannot be set and is subject to change in future revisions.

*TextLine Widget*

***XtNcaptionLabel***

| Class           | Type  | Default | Access |
|-----------------|-------|---------|--------|
| XtCCaptionLabel | O1Str | NULL    | SGI    |

Synopsis: The Label for the TextLine.

Values: Any O1Str valid in the current locale.

---

**Note** – This resource cannot be set and is subject to change in future revisions.

---

***XtNcaptionPosition***

| Class              | Type     | Default | Access |
|--------------------|----------|---------|--------|
| XtCCaptionPosition | O1Define | OL_LEFT | G      |

Synopsis: The position of the caption with respect to the text area.

---

**Note** – This resource cannot be set and is subject to change in future revisions.

---

***XtNcaptionSpace***

| Class           | Type      | Default | Access |
|-----------------|-----------|---------|--------|
| XtCCaptionSpace | Dimension | 4       | G      |

Synopsis: The separation between the caption and the text area in pixels.

---

**Note** – This resource cannot be set and is subject to change in future revisions.

---

***XtNcaptionWidth***

| Class           | Type      | Default | Access |
|-----------------|-----------|---------|--------|
| XtCCaptionWidth | Dimension | 0       | G      |

Synopsis: The width of caption text in pixels

---

**Note** – This resource cannot be set and is subject to change in future revisions.

---



**XtNcharsVisible**

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCCharsVisible | int  | 0       | GI     |

Synopsis: If nonzero, the initial width of text in terms of characters.

Values: Any integer

This resource overrides the `XtNwidth` setting. `XtNwidth` is then calculated as:

$$\text{XtNwidth} = \text{XtNcharsVisible} \times \text{max\_char\_width over the given FontSet.}$$

The actual number of characters visible could be more than the value of this resource since `XtNwidth` is computed based on the `max_char_width` of the given `FontSet`. The resource value changes with Geometry changes.

If `XtNcharsVisible` is zero, the width of the text is determined by `XtNmaximumChars`. See page 697.

**XtNcommitCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when a <Tab> or <Return> is inserted into the TextLine.

The `call_data` structure is:

```
typedef struct {
 int reason;
 XEvent *event;
 Boolean valid;
 OlStr buffer;
 int length;
} OlTLCommitCallbackStruct;
```

Fields in the `call_data` structure are:

**reason**      OL\_REASON\_COMMIT

**event**        A pointer to the corresponding `XEvent` structure

**valid**        A field to be set by the callback to indicate the validity of the contents

**buffer**       A pointer to the `XtNstring` resource. This pointer is `ReadOnly` and `valid` only within the callback.

**length**       Length of the text in characters, not including any terminating `NULL` character

*TextLine Widget*

This callback is invoked when the user hits the RETURN, NEXTFIELD, or PREVFIELD keys. If the callback sets *valid* to TRUE, the widget:

- Transfers focus to the next traversable widget if Mouseless mode is enabled.
- Transfers focus and insert point to the next TextLine or TextEdit if Mouseless is disabled.

If the callback sets *valid* to FALSE, the widget maintains focus and insert point in the current TextLine. The application also can provide additional feedback within this callback, such as popping up a Notice, clearing the field, or resetting the cursor position.

***XtNcursorPosition***

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCTextPosition | int  | 0       | SGI    |

Synopsis: The position of the cursor.

Values:  $0 \leq \text{XtNcursorPosition} \leq \text{total number-of-characters}$

Setting this resource will cause scrolling if the new position is beyond the visual area.

***XtNeditType***

| Class       | Type     | Default      | Access |
|-------------|----------|--------------|--------|
| XtCEditType | OlDefine | OL_TEXT_EDIT | SGI    |

Synopsis: The edit mode.

Values: OL\_TEXT\_EDIT/"text\_edit" - The text is editable.  
 OL\_TEXT\_READ/"text\_read" - The text is read-only. In read-only mode, the input-caret is disabled and the widget is not traversable or selectable.

***XtNimPreeditStyle***

| Class             | Type             | Default       | Access |
|-------------------|------------------|---------------|--------|
| XtCImPreeditStyle | OIImPreeditStyle | OL_NO_PREEDIT | GI     |

Synopsis: The pre-edit style (in conjunction with the shell's XtNimStatusStyle resource). If the pre-edit style is not supported by the input method, the ability to pre-edit is lost.

Values: OL\_ON\_THE\_SPOT/"onTheSpot" - The pre-edit data is displayed at the insertion point in the application window. The preexisting user data is shifted and the pre-edit data is inserted at the point of insertion.

*TextLine Widget*

OL\_OVER\_THE\_SPOT/"overTheSpot" - The pre-edit data is displayed in the application window, starting at the insertion point. As the user types the pre-edit data, the preexisting user data is obscured by the pre-edit data.

OL\_ROOT\_WINDOW/"rootWindow" - The pre-edit data is displayed in a child of the root window, away from the point of insertion.

OL\_NO\_PREEDIT/"none" - No pre-edit data is displayed.

See "XtNimStatusStyle" on page 44 and "Setting the Input Method Pre-Edit and Status Styles (Asian Locales Only)" on page 82.

***XtNinitialDelay***

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCInitialDelay | int  | 500     | SGI    |

Synopsis: The time in milliseconds of the initial repeat delay to be used when the scrolling arrows are pressed.

Values: Any integer

***XtNinsertTab***

| Class        | Type    | Default | Access |
|--------------|---------|---------|--------|
| XtCInsertTab | Boolean | FALSE   | SGI    |

Synopsis: Determines whether tabs are insertable.

Values: TRUE/"true" - Tabs are insertable but are not used for forward traversal. Forward traversal can still be accomplished with Control-Tab.

FALSE/"false" - Tabs are not insertable and act as NEXTFIELD.

***XtNmaximumChars***

| Class           | Type | Default | Access |
|-----------------|------|---------|--------|
| XtCMaximumChars | int  | 0       | GI     |

Synopsis: The maximum internal buffer size in terms of characters.

Values: Any integer

If this resource is zero, then the internal buffer will increase in size dynamically.

If XtNcharsVisible is zero (its default value), it is set up as follows:

*TextLine Widget*

```
if (charsVisible == 0)
 charsVisible = (maximumChars ? maximumChars : 20)
```

**XtNmenu**

| Class       | Type   | Default   | Access |
|-------------|--------|-----------|--------|
| XtCReadOnly | Widget | (special) | GI     |

Synopsis: The handle to the MenuShell widget that is popped up when the user presses the MENU key over the TextLine. By default, this menu contains the OPEN LOOK specified default elements UNDO, CUT, COPY, PASTE, and DELETE.

The application can augment this menu by creating more items as children of the MenuShell's XtNmenuPane widget. It can also change attributes of the MenuShell or the default items within. However, none of the default items should be removed from the MenuShell.

The application can also install its own Menu by setting this resource while creating the TextLine widget. In this case, the application's menu will be popped up instead of the widget's built-in menu. In fact, the widget does not even create its own menu. Therefore, an application can share menus among multiple TextLine widgets and avoid the overhead of having multiple MenuShells.

**XtNmotionCallback**

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked when the cursor position changes.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 XEvent *event;
 Boolean valid;
 int current_cursor;
 int new_cursor;
} OlTLMotionCallbackStruct;
```

The fields in the *call\_data* structure are:

|                       |                                                                                                                                                                                                              |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i>         | OL_REASON_MOTION – Indicates that the callback was invoked due to non-programmatic cursor-movement.<br>OL_REASON_PROG_MOTION – Indicates that this callback was invoked due to programmatic cursor-movement. |
| <i>event</i>          | A pointer to the corresponding XEvent structure                                                                                                                                                              |
| <i>valid</i>          | If TRUE, the widget performs the Cursor motion. Otherwise, it does not.                                                                                                                                      |
| <i>current_cursor</i> | Current cursor position                                                                                                                                                                                      |
| <i>new_cursor</i>     | New cursor position                                                                                                                                                                                          |

### ***XtNpreModifyCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked before a modification of the buffer is attempted.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 XEvent *event;
 Boolean valid;
 int current_cursor;
 int new_cursor;
 OlStr buffer;
 int start;
 int replace_length;
 OlStr insert_buffer;
 int insert_length;
} OlTLPreModifyCallbackStruct;
```

The fields in the *call\_data* structure are:

|               |                                                                                                                                                                                                              |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i> | OL_REASON_PRE_MODIFICATION – Indicates that the callback was invoked due to non-programmatic edits.<br>OL_REASON_PROG_PRE_MODIFICATION – Indicates that this callback was invoked due to programmatic edits. |
| <i>event</i>  | A pointer to the corresponding XEvent structure                                                                                                                                                              |
| <i>valid</i>  | If TRUE, the widget performs the modification; otherwise, it does not                                                                                                                                        |

*TextLine Widget*

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>current_cursor</i> | Current position of the cursor                                                                                                                |
| <i>new_cursor</i>     | Cursor position after modification                                                                                                            |
| <i>buffer</i>         | A pointer to the <code>XtNstring</code> resource. Note that this pointer is <code>ReadOnly</code> and will be valid only within the callback. |
| <i>start</i>          | Start of text to be deleted or replaced                                                                                                       |
| <i>replace_length</i> | Length in characters of the text to be deleted or replaced, not including any terminating <code>NULL</code> character.                        |
| <i>insert_buffer</i>  | The new text to be inserted                                                                                                                   |
| <i>insert_length</i>  | Length in characters of the text to be inserted, not including any terminating <code>NULL</code> character.                                   |

All fields except *valid* are `ReadOnly`.

***XtNpostModifyCallback***

| Class       | Type           | Default | Access |
|-------------|----------------|---------|--------|
| XtCCallback | XtCallbackList | NULL    | SGIO   |

Synopsis: The callback list invoked after modification of the buffer is done.

The *call\_data* structure is:

```
typedef struct {
 int reason;
 XEvent *event;
 int cursor;
 OlStr buffer;
} OlTLPostModifyCallbackStruct;
```

Fields in the *call\_data* structure are:

|               |                                                                                                                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>reason</i> | OL_REASON_POST_MODIFICATION – Indicates that the callback was invoked due to non-programmatic edits.<br>OL_REASON_PROG_POST_MODIFICATION – Indicates that this callback was invoked due to programmatic edits. |
| <i>event</i>  | A pointer to the corresponding <code>XEvent</code> structure                                                                                                                                                   |
| <i>cursor</i> | Current position of cursor                                                                                                                                                                                     |
| <i>buffer</i> | A pointer to the <code>XtNstring</code> resource. This pointer is <code>ReadOnly</code> and will be valid only within the callback.                                                                            |

All fields are `ReadOnly`.

**XtNrepeatRate**

| Class         | Type | Default | Access |
|---------------|------|---------|--------|
| XtCRepeatRate | int  | 100     | SGI    |

Synopsis: The time in milliseconds of the repeat delay to be used when the scrolling arrows are pressed.

Values: Any integer

**XtNstring**

| Class     | Type  | Default | Access |
|-----------|-------|---------|--------|
| XtCString | OlStr | NULL    | SGI    |

Synopsis: The contents of the TextLine buffer.

Values: Any OlStr valid in the current locale.

On being set, the string is inserted into the widget's internal buffer.

XtGetValues() on this resource returns a pointer to the current data. The widget should treat the pointed-to data as ReadOnly and the pointed-to data is guaranteed to be valid only until the next Intrinsics call. Thus, if the application needs the data longer, it should make a copy of it.

**XtNunderline**

| Class        | Type    | Default | Access |
|--------------|---------|---------|--------|
| XtCUnderline | Boolean | TRUE    | SGI    |

Synopsis: The presence of the underline.

Values: TRUE/"true" - The underline is present.

FALSE/"false" - The underline is absent. Note that the *OPEN LOOK GUI Functional Specification* states that the underline should be removed if the widget is in read-only mode.

**XtNupdateDisplay**

| Class            | Type    | Default | Access |
|------------------|---------|---------|--------|
| XtCUpdateDisplay | Boolean | TRUE    | SGI    |

Synopsis: The redisplay of the screen.

Values: TRUE/"true" - Redisplay the widget in the current state.

FALSE/"false" - Screen redisplay is stopped until it is set back to TRUE.

This resource is useful during incremental programmatic edits.

*TextLine Widget*

*Activation Types*

The following table lists the activation types used by the TextLine.

*Table 10-13*TextLine Activation Types

| <b>Activation Type</b> | <b>Semantics</b> | <b>Resource Name</b> |
|------------------------|------------------|----------------------|
| OL_CANCEL              | CANCEL           | XtNcancelKey         |
| OL_CHARBAK             | LEFT             | XtNleftKey           |
| OL_CHARFWD             | RIGHT            | XtNrightKey          |
| OL_COPY                | COPY             | XtNcopyBtn           |
| OL_CUT                 | CUT              | XtNcutBtn            |
| OL_DEFAULTACTION       | DEFAULTACTION    | XtNdefaultActionKey  |
| OL_DELCHARBAK          | DELETE BACKWARD  | XtNdelCharBakFwd     |
| OL_DELCHARFWD          | DELETE FORWARD   | XtNdelCharFwdKey     |
| OL_DELLINE             | DELETE LINE      | XtNdelLineKey        |
| OL_DELLINEBAK          | DELLINEBAK       | XtNdelLineBakKey     |
| OL_DELLINEFWD          | DELLINEFWD       | XtNdelLineFwdKey     |
| OL_DELWORDBAK          | DELWORDBAK       | XtNdelWordBakKey     |
| OL_DELWORDFWD          | DELWORDFWD       | XtNdelWordFwdKey     |
| OL_HELP                | HELP             | XtNhelpKey           |
| OL_LINEEND             | ROW END          | XtNlineEndKey        |
| OL_LINESTART           | ROW START        | XtNlineStartKey      |
| OL_MOVEDOWN            | MOVEDOWN         | XtNdownKey           |
| OL_MOVELEFT            | MOVELEFT         | XtNleftKey           |
| OL_MOVERIGHT           | MOVERIGHT        | XtNrightKey          |
| OL_MOVEUP              | MOVEUP           | XtNupKey             |
| OL_NEXTFIELD           | NEXTFIELD        | XtNnextFieldKey      |
| OL_PASTE               | PASTE            | XtNpasteBtn          |
| OL_PREVFIELD           | PREVFIELD        | XtNprevFieldKey      |
| OL_TOGGLEPUSHPIN       | TOGGLEPUSHPIN    | XtNtogglePushpinKey  |
| OL_UNDO                | UNDO             | XtNundoKey           |
| OL_WORDBAK             | JUMP LEFT        | XtNwordBakKey        |
| OL_WORDFWD             | JUMP RIGHT       | XtNwordFwdKey        |

Activation types not described in the following table are described in “Common Activation Types” on page 68.



***OL\_CHARBAK***

The cursor is moved backward by one character. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to `FALSE`.

***OL\_CHARFWD***

The cursor is moved forward by one character. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to `FALSE`.

***OL\_COPY***

This activation type copies the current selection from the widget to the CLIPBOARD.

***OL\_CUT***

This activation type copies the current selection from the widget to the CLIPBOARD and also deletes the selected text from the widget. The following callbacks are invoked:

|                                    |                                                                                                                                                                                                   |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>XtNpreModifyCallback</code>  | Invoked before the deletion occurs. The callback can prevent the deletion by setting the <i>valid</i> field in the <i>call_data</i> to <code>FALSE</code> .                                       |
| <code>XtNpostModifyCallback</code> | Invoked after the deletion occurs.                                                                                                                                                                |
| <code>XtNmotionCallback</code>     | Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to <code>FALSE</code> . |

***OL\_DELCHARBAK***

If there exists a selection in the widget, it is deleted. If there is no selection, the character before the insert point is deleted. The following callbacks are invoked:

|                                    |                                                                                                                                                             |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>XtNpreModifyCallback</code>  | Invoked before the deletion occurs. The callback can prevent the deletion by setting the <i>valid</i> field in the <i>call_data</i> to <code>FALSE</code> . |
| <code>XtNpostModifyCallback</code> | Invoked after the deletion occurs.                                                                                                                          |

*TextLine Widget*

XtNmotionCallback      Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***OL\_DELCHARFWD***

If there exists a selection in the widget, it is deleted. If there is no selection, the character after the insert point is deleted. The following callbacks are invoked:

XtNpreModifyCallback      Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.

XtNpostModifyCallback      Invoked after the deletion occurs.

XtNmotionCallback      Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***OL\_DELLINE***

The whole line is deleted. The following callbacks are invoked:

XtNpreModifyCallback      Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.

XtNpostModifyCallback      Invoked after the deletion occurs.

XtNmotionCallback      Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***OL\_DELLINEBAK***

If there exists a selection in the widget, it is deleted. If there is no selection, the segment of the line before the insert point The following callbacks are invoked:

XtNpreModifyCallback      Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.

XtNpostModifyCallback      Invoked after the deletion occurs.

---

*TextLine Widget*

XtNmotionCallback      Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***OL\_DELLINEFWD***

If there exists a selection in the widget, it is deleted. If there is no selection, the segment of the line after the insert point is deleted. The following callbacks are invoked:

XtNpreModifyCallback      Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.

XtNpostModifyCallback      Invoked after the deletion occurs.

XtNmotionCallback      Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***OL\_DELWORDBAK***

If there exists a selection in the widget, it is deleted. If there is no selection, the word before the insert point is deleted. The following callbacks are invoked:

XtNpreModifyCallback      Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.

XtNpostModifyCallback      Invoked after the deletion occurs.

XtNmotionCallback      Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***OL\_DELWORDFWD***

If there exists a selection in the widget, it is deleted. If there is no selection, the word after the insert point is deleted. The following callbacks are invoked:

XtNpreModifyCallback      Invoked before the deletion occurs. The callback can prevent the deletion by setting the *valid* field in the *call\_data* to FALSE.

## TextLine Widget

---

|                       |                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNpostModifyCallback | Invoked after the deletion occurs.                                                                                                                                                  |
| XtNmotionCallback     | Invoked before the cursor position is changed due to the deletion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

### **OL\_LINESTART**

The cursor is moved to the start of the line. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

### **OL\_LINEEND**

The cursor is moved to the end of the line. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

### **OL\_NEXTFIELD**

This activation type invokes the `XtNcommitCallback`. If the callback sets *valid* to TRUE and if `XtNmouseless` is TRUE, the widget transfers focus to the next traversable widget. If the callback sets *valid* to TRUE and if `XtNmouseless` is FALSE, the widget transfers focus and sets the insert point to the next `TextLine` or `TextEdit` widget. If the callback sets *valid* to FALSE, the widget maintains focus and insert point within itself.

### **OL\_PASTE**

This activation type inserts the contents of the CLIPBOARD into the widget at the current insert point. The following callbacks are invoked:

|                       |                                                                                                                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XtNpreModifyCallback  | Invoked before the insertion occurs. The callback can prevent the insertion by setting the <i>valid</i> field in the <i>call_data</i> to FALSE.                                      |
| XtNpostModifyCallback | Invoked after the insertion occurs.                                                                                                                                                  |
| XtNmotionCallback     | Invoked before the cursor position is changed due to the insertion. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

***OL\_PREVFIELD***

This activation type invokes the `XtNcommitCallback`. If the callback sets *valid* to TRUE and if `XtNmouseless` is TRUE, the widget transfers focus to the previous traversable widget. If the callback sets *valid* to TRUE and if `XtNmouseless` is FALSE, the widget transfers focus and sets the insert point to the previous `TextLine` or `TextEdit` widget. If the callback sets *valid* to FALSE, the widget maintains focus and insert point within itself.

***OL\_UNDO***

This activation type undoes the last modification to the widget's text buffer. The following callbacks are invoked:

|                                    |                                                                                                                                                                                         |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>XtNpreModifyCallback</code>  | Invoked before any modification occurs. The callback can prevent the modification by setting the <i>valid</i> field in the <i>call_data</i> to FALSE.                                   |
| <code>XtNpostModifyCallback</code> | Invoked after any modification occurs.                                                                                                                                                  |
| <code>XtNmotionCallback</code>     | Invoked before the cursor position is changed due to the modification. The callback can prevent the cursor movement by setting the <i>valid</i> field in the <i>call_data</i> to FALSE. |

***OL\_WORDBAK***

The cursor is moved backward by one word. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***OL\_WORDFWD***

The cursor is moved forward by one word. The `XtNmotionCallback` callback is invoked before the cursor position is changed. The callback can prevent the cursor movement by setting the *valid* field in the *call\_data* to FALSE.

***See Also***

“`NumericField` Widget” on page 443,  
“`TextField` Widget” on page 665,  
“`TextLine` Functions” on page 708.

### TextLine Functions

---

These functions manipulate the contents of the TextLine widget.

#### *OlTLGetPosition*

```
#include <TextLine.h>
int OlTLGetPosition(
 Widget w,
 int pos);
```

`OlTLGetPosition()` returns some key positions in the text line. Valid values for *pos* are:

|                                    |                                           |
|------------------------------------|-------------------------------------------|
| <code>OL_CURSORPOS</code>          | Return the cursor position.               |
| <code>OL_BEGIN_CURRENT_WORD</code> | Return the beginning of the current word. |
| <code>OL_END_CURRENT_WORD</code>   | Return the end of the current word.       |
| <code>OL_END_LINE</code>           | Return the end of the line.               |

`OlTLGetPosition()` returns the position corresponding to the specified value of *pos*. (No error information is available.)

#### *OlTLGetSelection*

```
#include <Xol/TextLine.h>
OlStr OlTLGetSelection(
 Widget w,
 int *start,
 int *length);
```

`OlTLGetSelection()` returns the current Selection as well as the Selection *start* and *length*. The returned string should be freed by the application, when no longer required, using `XtFree()`. `OlTLGetSelection()` returns NULL if no selection is active.

### *OlTlGetSubString*

```
#include <Xol/TextLine.h>
OlStr OlTlGetSubString(
 Widget w,
 int start,
 int length);
```

`OlTlGetSubString()` returns *length* characters beginning at *start*. The returned string should be freed by the application using `XtFree()`. `OlTlGetSubString()` returns NULL on failure. Failures include invalid *start* and *length* values.

### *OlTlOperateOnSelection*

```
#include <Xol/TextLine.h>
Boolean OlTlOperateOnSelection(
 Widget w,
 int mode);
```

`OlTlOperateOnSelection()` performs various operations on the primary or CLIPBOARD selections. Valid values for *mode* are:

- OL\_CUT** Copies the primary selection to the CLIPBOARD, then deletes the primary selection. This operation invokes the `XtNpreModifyCallback` and `XtNpostModifyCallback` callbacks.
- OL\_COPY** Copies the primary selection to the CLIPBOARD.
- OL\_PASTE** Inserts the CLIPBOARD selection at the destination cursor. If the destination cursor is inside the current selection, the CLIPBOARD selection replaces the selected text. This operation invokes the `XtNpreModifyCallback` and `XtNpostModifyCallback` callbacks.
- OL\_CANCEL** Clears the primary selection.

`OlTlOperateOnSelection()` returns FALSE if the primary selection is NULL, if the widget does not own the primary selection, or if the function is unable to gain ownership of the CLIPBOARD selection. Otherwise, it returns TRUE.

### *OLTLSetSelection*

```
#include <Xol/TextLine.h>
Boolean OLTLSetSelection(
 Widget w,
 int start,
 int length);
```

`OLTLSetSelection()` selects *length* characters beginning at *start*. It returns TRUE on success, FALSE on failure. Failures include invalid *start* and *length* values.

### *OLTLSetSubString*

```
#include <Xol/TextLine.h>
Boolean OLTLSetSubString(
 Widget w,
 int start,
 int length,
 OlStr buffer);
```

`OLTLSetSubString()` can be used to do insertion, replacement, and deletion of substrings. It replaces *length* characters, beginning at *start*, with the contents of *buffer*. It returns TRUE on success, FALSE on failure. Failures include invalid *start* and *length* values and memory allocation failures.

### *See Also*

“TextLine Widget” on page 688.



# Index

---

## Symbols

.Xdefaults file, 17  
font sets, 87

## Numerics

2D/3D resource, 26

## A

Abbreviated Scrollbars, 520

AbbrevMenuItem, 227

Activation Types, 233

Ancestry, 227

Coloration, 229

Components, 228

Core Resources, 230

Current Selection Widget, 228

Keyboard Accelerator Display, 230

Keyboard Mnemonic Display, 230

Keyboard Traversal, 229

OL\_MENU Activation Type, 233

OL\_MENUKEY Activation Type, 234

OL\_SELECT Activation Type, 234

OL\_SELECTKEY Activation Type,  
234

Primitive Resources, 231

Resources, 231

Subwidget, 228

Subwidget Resources, 232

XtNmenuItem resource, 232

XtNpreviewWidget resource, 232

accelerator string resource, 35

accelerators *See* keyboard accelerators

Access column of resource tables, 16

activating widgets, 103

Activation Types

AbbrevMenuItem widget, 233

BulletinBoard widget, 238

Caption widget, 245

CheckBox widget, 258

common types, 78

ControlArea widget, 267

DrawArea widget, 275

DropTarget widget, 286

Exclusives widget, 292

FileChooser widget, 320

FileChooserShell widget, 329

FlatCheckBox widget, 345

FlatExclusives widget, 355

FlatNonexclusives widget, 362

FontChooser widget, 384

Form widget, 404

Gauge widget, 411

MenuItem widget, 421

MenuShell widget, 437

Nonexclusives widget, 442

---

NoticeShell widget, 390, 452  
 NumericField widget, 467  
 OblongButton widget, 483  
 PopupWindowShell widget, 497  
 RectButton widget, 509  
 relationship to virtual events, 71  
 RubberTile widget, 517  
 Scrollbar widget, 535  
 ScrolledWindow widget, 553  
 ScrollingList widget, 584  
 Slider widget, 607  
 StaticText widget, 617  
 Stub widget, 631  
 table, 75  
 TextEdit widget, 652  
 TextField widget, 684  
 TextLine widget, 711  
 active pointer grabbing, 218  
 Aligning Captions, 263  
 AllocateBuffer(), 105  
 AllocateTextBuffer(), 173  
 Alt key, 23  
 anchor, 518  
 application resources, *See* OLIT Toolkit Resources, 17  
 ApplicationShell Resources, 61  
     XtNargc, 61  
     XtNargv, 61  
 application-specific data resource, 40, 58  
 arrow button in AbbreviatedMenuButton, 228  
 aspect ratios resource, 48  
 audible warnings resource, 18  
  
**B**  
 background color resource, 19, 28  
 background pixmap resource, 28  
 BackwardScanTextBuffer(), 174  
 base window  
     creating, 41  
     icon from window ID, 47  
     icon mask, 46  
     iconifying, 60  
     iconifying at startup, 47  
     image of icon, 46  
     initial state, 47  
     location, 47  
     of shell widget, 51  
     title, 49  
 beep resource, 18  
 beep volume resource, 18  
 BG0 - BG3, 19  
 border color resource, 29  
 border pixmap resource, 29  
 border width resource, 29  
 Buffer Functions, 105  
     AllocateBuffer(), 105  
     Buffer Macros, 105  
     CopyBuffer(), 106  
     FreeBuffer(), 106  
     GrowBuffer(), 106  
     InsertIntoBuffer(), 106  
     ReadFileIntoBuffer(), 107  
     ReadStringIntoBuffer(), 107  
     strclose(), 108  
     strgetc(), 108  
     stropen(), 108  
 Buffer Macros, 105  
 buffer object, 105  
 BulletinBoard, 235  
     Activation Types, 238  
     Ancestry, 235  
     Coloration, 236  
     Composite Resources, 237  
     Core Resources, 236  
     Keyboard Traversal, 235  
     Manager Resources, 237  
     Resources, 237  
     XtNlayout resource, 237  
 BulletinBoard Resources  
     DrawArea widget, 272  
 busy marking resource, 53  
 button events  
     remapping and consuming in Text, 642  
 ButtonPress event, 222

---

ButtonRelease event, 222

## C

callback

- Activation Types, 72
- destruction, 30
- dynamic, 150
- flat widget, 335
- popdown, 43
- popup, 43
- Preview and Animate, 122
- virtual events, 72
- XEvent consumption, 53

caps lock resource, 21

Caption, 239

- Activation Types, 245
- Ancestry, 239
- Child Constraints, 240
- Coloration, 240
- Components, 239
- Composite Resources, 242
- Core Resources, 241
- Keyboard Accelerator Display, 241
- Keyboard Mnemonic Display, 241
- Keyboard Traversal, 240
- Layout Control, 240
- Manager Resources, 242
- OL\_SELECT Activation Type, 246
- OL\_SELECTKEY Activation Type, 246
- Resources, 242
- Widget, 239
- XtNalignment resource, 243
- XtNfont resource, 243
- XtNfontColor resource, 243
- XtNlabel resource, 244
- XtNmnemonic resource, 244
- XtNposition resource, 244
- XtNrecomputeSize resource, 244
- XtNspace conflict with RubberTile, 516
- XtNspace resource, 245
- XtNtextFormat resource, 245

cascade menu, 414

changebars, 259

character encoding, 84

CheckBox, 247

- Activation Types, 258
- Ancestry, 247
- Appearance with Set/Default/Dim, 250
- Bounds on, 248
- Coloration, 249
- Components, 247
- Composite Resources, 252
- Core Resources, 251
- Keyboard Accelerator Display, 250
- Keyboard Mnemonic Display, 250
- Keyboard Traversal, 249
- Label and Check Box Appearance, 251
- Label Resource Interactions, 251
- Manager Resources, 252
- OL\_SELECT Activation Type, 258
- OL\_SELECTKEY Activation Type, 258
- Operations, 248
- Resources, 253
- SELECT, 248
- XtNaccelerator resource, 253
- XtNacceleratorText resource, 253
- XtNdim resource, 254
- XtNfont resource, 254
- XtNfontColor resource, 254
- XtNforeground resource, 254
- XtNlabel resource, 254
- XtNlabelImage resource, 255
- XtNlabelJustify resource, 255
- XtNlabelTile resource, 255
- XtNlabelType resource, 256
- XtNmnemonic resource, 256
- XtNposition resource, 256
- XtNrecomputeSize resource, 256
- XtNscale resource, 257
- XtNselect resource, 257
- XtNset resource, 257
- XtNtextFormat resource, 257
- XtNunselect resource, 257

children list resource, 33

---

click timing resource, 24  
 click-move-click menu, 426  
 click-move-click mode, 422  
 CLIPBOARD operations, 215  
 color  
     background resource, 19, 28  
     border resource, 29  
     foreground, 37  
     highlight, 19  
     shadow, 19  
     tuples resource, 19  
 colormap resource, 29  
 command line arguments, 62  
 committing input, 91  
 Composite Resources, 33  
     BulletinBoard widget, 237  
     Caption widget, 242  
     CheckBox widget, 252  
     ControlArea widget, 261  
     DrawArea widget, 271  
     Exclusives widget, 290  
     FileChooser widget, 296  
     FileChooserShell widget, 324  
     FontChooser widget, 371  
     FontChooserShell widget, 387  
     FooterPanel widget, 393  
     Form widget, 399  
     MenuShell widget, 432  
     Nonexclusives widget, 440  
     NoticeShell widget, 447  
     PopupWindowShell widget, 490  
     RubberTile widget, 514  
     ScrolledWindow widget, 544  
     ScrollingList widget, 565  
     TextField widget, 679  
     XtNchildren, 33  
     XtNinsertPosition, 33  
     XtNnumChildren, 34  
 Constraint Resources  
     RubberTile widget, 515  
 consume an XEvent resource, 36  
 control key resource, 20  
 ControlArea, 259  
     Activation Types, 267  
     Ancestry, 259  
     Coloration, 260  
     Components, 259  
     Composite Resources, 261  
     Core Resources, 260  
     Layout Control, 259  
     Manager Resources, 261  
     Padding Around Controls, 264  
     Resources, 261  
     Subwidget Resources, 262  
     XtNalignCaptions resource, 262  
     XtNallowChangeBars resource, 262  
     XtNcenter resource, 263  
     XtNchangeBar resource, 263  
     XtNhPad resource, 264  
     XtNhSpace resource, 264  
     XtNlayoutType resource, 265  
     XtNmeasure resource, 266  
     XtNsameSize resource, 266  
     XtNvPad resource, 264  
     XtNvSpace resource, 264  
 ControlArea Subwidget Resources  
     NoticeShell widget, 450  
     PopupWindowShell widget, 494  
 converting pixel dimensions, 168  
 CopyBuffer(), 106  
 copying text, 215  
 CopyTextBufferBlock(), 174  
 Core Resources, 27  
     AbbrevMenuButton widget, 230  
     BulletinBoard widget, 236  
     Caption widget, 241  
     CheckBox widget, 251  
     ControlArea widget, 260  
     DrawArea widget, 271  
     DropTarget widget, 277  
     Exclusives widget, 289  
     FileChooser widget, 295  
     FileChooserShell widget, 324  
     FlatCheckBox widget, 342  
     FlatExclusives widget, 349  
     FlatNonexclusives widget, 359  
     FontChooser widget, 370  
     FontChooserShell widget, 386  
     FooterPanel widget, 392

---

Form widget, 399  
 Gauge widget, 407  
 MenuButton widget, 417  
 MenuShell widget, 432  
 Nonexclusives widget, 440  
 NoticeShell widget, 446  
 NumericField widget, 456  
 OblongButton widget, 479  
 PopupWindowShell widget, 490  
 RectButton widget, 503  
 RubberTile widget, 514  
 Scrollbar widget, 525  
 ScrolledWindow widget, 544  
 ScrollingList widget, 564  
 Slider widget, 600  
 StaticText widget, 612  
 Stub widget, 621  
 TextEdit widget, 639  
 TextField widget, 678  
 TextLine widget, 700  
 XtNaccelerators, 27  
 XtNancestorSensitive, 28  
 XtNbackground, 28  
 XtNbackgroundPixmap, 28  
 XtNborderColor, 29  
 XtNborderPixmap, 29  
 XtNborderWidth, 29  
 XtNcolormap, 29  
 XtNdepth, 30  
 XtNdestroyCallback, 30  
 XtNheight, 30  
 XtNmappedWhenManaged, 31  
 XtNscreen, 31  
 XtNsensitive, 31  
 XtNtranslations, 32  
 XtNwidth, 32  
 XtNx, 32  
 XtNy, 32  
 corners, resize, 57  
 Cursor and Pixmap Functions, 109  
   GetOlBusyCursor(), 115  
   GetOlDuplicateCursor(), 116  
   GetOlMoveCursor(), 116  
   GetOlPanCursor(), 116  
   GetOlQuestionCursor(), 116  
   GetOlStandardCursor(), 116  
   GetOlTargetCursor(), 117  
   OlGet50PercentGrey(), 117  
   OlGet75PercentGrey(), 117  
   OlGetDataDupeDragCursor(), 109  
   OlGetDataDupeDropCursor(), 109  
   OlGetDataDupeInsertCursor(), 109  
   OlGetDataDupeNoDropCursor(), 110  
   OlGetDataMoveDragCursor(), 110  
   OlGetDataMoveDropCursor(), 110  
   OlGetDataMoveInsertCursor(), 110  
   OlGetDataMoveNoDropCursor(),  
     110  
   OlGetDocCursor(), 110  
   OlGetDocStackCursor(), 110  
   OlGetDropCursor(), 111  
   OlGetDupeDocCursor(), 111  
   OlGetDupeDocDragCursor(), 111  
   OlGetDupeDocDropCursor(), 111  
   OlGetDupeDocNoDropCursor(), 111  
   OlGetDupeStackCursor(), 111  
   OlGetDupeStackDragCursor(), 111  
   OlGetDupeStackDropCursor(), 112  
   OlGetDupeStackNoDropCursor(),  
     112  
   OlGetFolderCursor(), 112  
   OlGetFolderStackCursor(), 112  
   OlGetMoveDocCursor(), 112  
   OlGetMoveDocDragCursor(), 112  
   OlGetMoveDocDropCursor(), 112  
   OlGetMoveDocNoDropCursor(), 113  
   OlGetMoveStackCursor(), 113  
   OlGetMoveStackDragCursor(), 113  
   OlGetMoveStackDropCursor(), 113  
   OlGetMoveStackNoDropCursor(),  
     113  
   OlGetNoDropCursor(), 113  
   OlGetTextDupeDragCursor(), 113  
   OlGetTextDupeDropCursor(), 114  
   OlGetTextDupeInsertCursor(), 114  
   OlGetTextDupeNoDropCursor(), 114  
   OlGetTextMoveDragCursor(), 114  
   OlGetTextMoveDropCursor(), 114  
   OlGetTextMoveInsertCursor(), 114  
   OlGetTextMoveNoDropCursor(), 114  
   Other Version 2 Cursors, 115

---

    Pixmap Functions, 117  
    Version 2 Drag and Drop Cursors, 115  
    Version 3 Cursors, 109  
cutting text, 215

## D

data, application-specific, 40, 58  
default font resource, 24  
default label font resource, 36  
default text format, 86  
DeltaButtons, 454  
depth  
    specifying a visual, 164  
depth resource, 30  
destruction callback, 30  
Differences From OLIT Release 3.1, 11  
display  
    updating, 118  
display depth resource, 30  
Display Functions, 118  
    OIUpdateDisplay(), 118  
Drag and Drop Functions, 119  
    Begin Drag, 121  
    Cleanup, 124  
    client\_data, 124  
    Closing Handshake, 123  
    Common Arguments, 124  
    Data Structures, 127  
    Destination Functions, 138  
    Drop and Data Transfer, 122  
    Drop Rectangle, 119  
    Drop Site, 119  
    Drop Site Manipulation Functions,  
        125  
    Drop Target, 120  
    dropsiteid, 124  
    General Purpose Functions, 126  
    Handshake Functions, 126  
    Message Functions, 126  
    Notify Procedure Prototypes, 128  
    OIDnDAllocTransientAtom(), 132  
    OIDnDBeginSelectionTransaction(),  
        138

    OIDnDChangeDropSitePreview-  
        Hints(), 139  
    OIDnDClearDragState(), 133  
    OIDnDDeliverPreviewMessage(), 133  
    OIDnDDeliverTriggerMessage(), 133  
    OIDnDDestroyDropSite(), 139  
    OIDnDDisownSelection(), 134  
    OIDnDDragAndDrop(), 134  
    OIDnDDragNDropDone(), 139  
    OIDnDDropSiteID(), 127  
    OIDnDEndSelectionTransaction(),  
        140  
    OIDnDErrorDuringSelection-  
        Transaction(), 141  
    OIDnDFreeTransientAtom(), 135  
    OIDnDGetCurrentSelectionsFor-  
        Widget(), 141  
    OIDnDGetDropSitesOfWidget(), 142  
    OIDnDGetDropSitesOfWindow(),  
        142  
    OIDnDGetWidgetOfDropSite(), 143  
    OIDnDGetWindowOfDropSite(), 143  
    OIDnDInitializeDragState(), 136  
    OIDnDOWnSelection(), 136  
    OIDnDOWnSelectionIncremental(),  
        136  
    OIDnDPMNotifyProc(), 129  
    OIDnDPreviewAndAnimate(), 137  
    OIDnDPreviewAnimateCbP(), 128  
    OIDnDProtocolActionCbP(), 129  
    OIDnDQueryDropSiteInfo(), 144  
    OIDnDRegisterWidgetDropSite(),  
        145  
    OIDnDRegisterWindowDropSite(),  
        146  
    OIDnDSetDropSiteInterest(), 146  
    OIDnDSetInterestInWidgetHier(),  
        147  
    OIDnDSitePreviewHints(), 127  
    OIDnDSiteRect(), 127  
    OIDnDTMNotifyProc(), 131  
    OIDnDTransactionStateCallback(),  
        130  
    OIDnDUpdateDropSiteGeometry(),  
        147

---

- OldNdWidgetConfiguredInHier(), 147
- OlGrabDragPointer(), 137
- OlUngrabDragPointer(), 138
- Owner, 120
- Preview And Animate, 122
- Preview Message Notify Procedure, 120
- preview\_hints, 125
- Selection Functions, 126
- Setup, 121
- Source Functions, 132
- Trigger Message Notify Procedure, 120
- Drag Box Range of Movement, 605
- drag distance resource, 20
- drag distance, specifying, 23
- DrawArea, 269
  - Activation Types, 275
  - Ancestry, 269
  - BulletinBoard Resources, 272
  - Coloration, 270
  - Composite Resources, 271
  - Core Resources, 271
  - Keyboard Traversal, 270
  - Manager Resources, 271
  - Multiple Visuals Support, 270
  - Resources, 272
  - XtNexposeCallback resource, 272
  - XtNforeground resource, 273
  - XtNgraphicsExposeCallback resource, 273
  - XtNresizeCallback resource, 274
  - XtNvisual resource, 274
- Drop Target Coloration, 277
- DropTarget, 276
  - Activation Types, 286
  - Ancestry, 276
  - Coloration, 276
  - Core Resources, 277
  - OldDropTargetCallbackStruct structure, 279
  - Pixmap Resources, 278
  - Primitive Resources, 278
  - Resources, 278
  - XtNbusyPixmap resource, 281
  - XtNdndAcceptCursor resource, 281
  - XtNdndAnimateCallback resource, 282
  - XtNdndCopyCursor resource, 282
  - XtNdndMoveCursor resource, 282
  - XtNdndPreviewCallback resource, 283
  - XtNdndPreviewHints resource, 282
  - XtNdndRejectCursor resource, 283
  - XtNdndTriggerCallback resource, 284
  - XtNfull resource, 284
  - XtNownSelectionCallback resource, 284
  - XtNrecomputeSize resource, 285
  - XtNselectionAtom resource, 285
- dynamic callback list, 150
- Dynamic Resource Functions, 150
  - OlCallDynamicCallbacks(), 150
  - OlRegisterDynamicCallback(), 150
  - OlUnregisterDynamicCallback(), 151
- dynamic resources, 16

## E

- Editable ScrollingList, 558
- elevator, 518
- emanate widget, 443
- encoding
  - multibyte, 84
  - single-byte, 84
  - wide character, 84
- EndCurrentTextBufferWord(), 175
- EnterNotify event, 222
- Error Functions, 152
  - OlError(), 152
  - OlErrorHandler(), 154
  - OlSetErrorHandler(), 153
  - OlSetVaDisplayErrorMsg-Handler(), 154
  - OlSetVaDisplayWarningMsgHandler(), 154
  - OlSetWarningHandler(), 153

---

OIvaDisplayErrorMsg(), 152  
 OIvaDisplayErrorMsgHandler(), 154  
 OIvaDisplayWarningMsg(), 153  
 OIvaDisplayWarningMsgHandler(), 155  
 OIWarning(), 152  
 OIWarningHandler(), 154  
 Exclusive Buttons Example, 287  
 Exclusives, 287  
   Activation Types, 292  
   Ancestry, 287  
   Child Constraint, 288  
   Coloration, 288  
   Composite Resources, 290  
   Core Resources, 289  
   Grid Layout and Button Labels, 287  
   Keyboard Traversal, 289  
   Manager Resources, 290  
   Menu Use, 288  
   Resources, 290  
   Selection Control, 288  
   XtNlayoutType resource, 291  
   XtNmeasure resource, 291  
   XtNnoneSet resource, 291  
   XtNrecomputeSize resource, 292  
 expose event, 273  
 exposure events  
   processing, 118

**F**

FileChooser, 294  
   Accelerators, 313  
   Activation Types, 320  
   Ancestry, 294  
   Base Resources, 300  
   Coloration, 295  
   Component Access, 314  
   Composite Resources, 296  
   Core Resources, 295  
   Customization Resources, 311  
   Extensibility Callbacks, 316  
   Extensibility Resources, 314  
   Extension Container, 314  
   File Filtering, 307  
   GoTo Control, 309  
   Known Deficiencies, 295  
   Labels, 318  
   Manager Resources, 296  
   Mnemonics, 313  
   Pathname Processing, 312  
   Resources, 297  
   RubberTile Resources, 296  
   Sorting, 311  
   Sorting Styles, 312  
   Standard Callbacks, 302  
   XtNapplicationFolders resource, 309  
   XtNapplicationFoldersMaxCount resource, 309  
   XtNcancelAccelerator resource, 313  
   XtNcancelButtonWidget resource, 315  
   XtNcancelCallback resource, 316  
   XtNcancelLabel resource, 319  
   XtNcancelMnemonic resource, 313  
   XtNcommandButtonWidget resource, 315  
   XtNcomparisonFunc resource, 311  
   XtNcurrentFolder resource, 300  
   XtNcurrentFolderLabelString resource, 319  
   XtNcurrentFolderLabelWidget resource, 315  
   XtNcurrentFolderWidget resource, 315  
   XtNdefaultDocumentName resource, 319  
   XtNdefaultDocumentSuffix resource, 320  
   XtNdocumentListWidget resource, 315  
   XtNdocumentNameLabelWidget resource, 315  
   XtNdocumentNameTypeInWidget resource, 315  
   XtNexpandTilde resource, 312  
   XtNextensionClass resource, 314  
   XtNextensionName resource, 314  
   XtNextensionWidget resource, 314  
   XtNfilterProc resource, 307  
   XtNfilterString resource, 308



---

XtNfolderOpenedCallback resource, 316  
 XtNfolderPromptString resource, 318  
 XtNfollowSymlinks resource, 300  
 XtNfont resource, 300  
 XtNfontColor resource, 300  
 XtNforeground resource, 301  
 XtNgotoButtonWidget resource, 315  
 XtNgotoHomeAccelerator resource, 313  
 XtNgotoHomeButtonWidget resource, 315  
 XtNgotoHomeLabel resource, 319  
 XtNgotoHomeMnemonic resource, 313  
 XtNgotoLabel resource, 319  
 XtNgotoMenuWidget resource, 315  
 XtNgotoPromptString resource, 318  
 XtNgotoPromptWidget resource, 315  
 XtNgotoTypeInWidget resource, 315  
 XtNgoUpOneFolderLabel resource, 319  
 XtNhideDotFiles resource, 308  
 XtNhistoryFoldersMaxCount resource, 309  
 XtNhistoryFoldersMinCount resource, 309  
 XtNhomeFolder resource, 310  
 XtNincludeAccelerator resource, 313  
 XtNincludeLabel resource, 319  
 XtNincludeMnemonic resource, 313  
 XtNinputDocumentCallback resource, 303  
 XtNlastDocumentName resource, 301  
 XtNlistChoiceCallback resource, 317  
 XtNlistPromptWidget resource, 315  
 XtNlistVisibleItemCount resource, 301  
 XtNlistVisibleItemMinCount resource, 301  
 XtNopenAccelerator resource, 313  
 XtNopenButtonWidget resource, 315  
 XtNopenFolderAccelerator resource, 313  
 XtNopenFolderCallback resource, 305  
 XtNopenFolderLabel resource, 319  
 XtNopenFolderMnemonic resource, 313  
 XtNopenLabel resource, 319  
 XtNopenMnemonic resource, 313  
 XtNopenPromptString resource, 318  
 XtNoperation resource, 301  
 XtNoutputDocumentCallback resource, 306  
 XtNsaveAccelerator resource, 313  
 XtNsaveAsAccelerator resource, 313  
 XtNsaveAsLabel resource, 319  
 XtNsaveAsMnemonic resource, 313  
 XtNsaveLabel resource, 319  
 XtNsaveMnemonic resource, 313  
 XtNscale resource, 302  
 XtNshowGlyphs resource, 302  
 XtNshowInactive resource, 308  
 XtNsubstituteShellVariables resource, 312  
 XtNtextFormat resource, 302  
 XtNuserFolders resource, 310  
 XtNuserFoldersMaxCount resource, 311  
 FileChooserShell, 321  
   Activation Types, 329  
   Ancestry, 321  
   Coloration, 322  
   Composite Resources, 324  
   Core Resources, 324  
   Resources, 327  
   Shell Resources, 324  
   TransientShell Resources, 327  
   VendorShell Resources, 326  
   WMShell Resources, 325  
 XtNfileChooserWidget resource, 327  
 XtNoperation resource, 327  
 XtNpointerWarping resource, 328  
 XtNtextFormat resource, 328  
 XtNverifyCallback resource, 328  
 Flat, 331  
   Callbacks and Flat Widgets, 335  
   FlatExclusives Settings Example, 332

---

- Item Lists and Allowable Resources, 331
  - registering help for Flat widgets, 156
  - Registering Help on Items, 338
  - Setting/Getting the State of an Item, 336
  - Specifying Items, 334
- Flat Resources, 62
  - FlatCheckBox widget, 343
  - FlatExclusives widget, 351
  - FlatNonexclusives widget, 360
  - XtNgravity, 63
  - XtNhPad, 64
  - XtNhSpace, 64
  - XtNitemFields, 64
  - XtNitemGravity, 65
  - XtNitemMaxHeight, 65
  - XtNitemMaxWidth, 65
  - XtNitemMinHeight, 65
  - XtNitemMinWidth, 65
  - XtNitems, 66
  - XtNitemsTouched, 66
  - XtNlabel, 67
  - XtNlabelImage, 67
  - XtNlabelJustify, 67
  - XtNlabelTile, 67
  - XtNlayoutHeight, 68
  - XtNlayoutType, 68
  - XtNlayoutWidth, 68
  - XtNmeasure, 69
  - XtNnumItemFields, 69
  - XtNnumItems, 69
  - XtNsameHeight, 69
  - XtNsameWidth, 70
  - XtNvPad, 64
  - XtNvSpace, 64
- Flat Widget Functions, 364
  - OIFlatCallAcceptFocus(), 364
  - OIFlatGetFocusItem(), 364
  - OIFlatGetItemGeometry(), 365
  - OIFlatGetItemIndex(), 364
  - OIFlatGetValues(), 365
  - OIFlatSetValues(), 366
  - OIVaFlatGetValues(), 365
  - OIVaFlatSetValues(), 366
- FlatCheckBox, 339
  - Activation Types, 345
  - Ancestry, 339
  - Coloration, 340
  - Components, 340
  - Core Resources, 342
  - Flat Resources, 343
  - FlatExclusives Resources, 344
  - Keyboard Accelerator Display, 341
  - Keyboard Mnemonic Display, 341
  - Keyboard Traversal, 341
  - OL\_SELECT Activation Type, 346
  - OL\_SELECTKEY Activation Type, 346
  - Primitive Resources, 343
  - Resources, 345
  - XtNposition resource, 345
- FlatExclusives, 347
  - Activation Types, 355
  - Ancestry, 347
  - Coloration, 348
  - Core Resources, 349
  - Flat Resources, 351
  - Keyboard Traversal, 349
  - Menu Use, 348
  - OL\_MENU Activation Type, 355
  - OL\_MENUDEFAULT Activation Type, 355
  - OL\_MENUDEFAULTKEY Activation Type, 355
  - OL\_MENUEKEY Activation Type, 355
  - OL\_SELECT Activation Type, 356
  - OL\_SELECTKEY Activation Type, 356
  - Primitive Resources, 350
  - Resources, 352
  - Selection Control, 347
  - XtNclientData resource, 352
  - XtNdefault resource, 352
  - XtNdim resource, 353
  - XtNhSpace resource, 353
  - XtNnoneSet resource, 354
  - XtNselectProc resource, 354
  - XtNset resource, 354
  - XtNunselectProc resource, 354

---

- XtNvSpace resource, 353
- FlatExclusives Resources
  - FlatCheckBox widget, 344
  - FlatNonexclusives widget, 361
- FlatNonexclusives, 357
  - Activation Types, 362
  - Ancestry, 357
  - Coloration, 358
  - Core Resources, 359
  - Default Spacing, 357
  - Flat Resources, 360
  - FlatExclusives Resources, 361
  - Keyboard Traversal, 358
  - Menu Use, 357
  - OL\_MENU Activation Type, 362
  - OL\_MENUDEFAULT Activation Type, 363
  - OL\_MENUDEFAULTKEY Activation Type, 363
  - OL\_MENUKEY Activation Type, 362
  - OL\_SELECT Activation Type, 363
  - OL\_SELECTKEY Activation Type, 363
  - Primitive Resources, 360
- follower widgets, 103
- font
  - default font resource, 24
  - default label, 36
  - text color, 37
- font set
  - Fontset Definitions in OpenWindows.fs, 368
  - status feedback, 54
- font set handling, 87
- FontChooser, 367
  - Activation Types, 384
  - Ancestry, 367
  - Appearance, 368
  - Coloration, 368
  - Composite Resources, 371
  - Core Resources, 370
  - Default Font Family, 369
  - Default Font Scales, 369
  - Fontset Definitions in OpenWindows.fs, 368
  - Fontset Name Aliases, 369
  - Fontset Specifier, 368
  - Manager Resources, 371
  - OL\_PROPERTY, 370
  - propertiesKey, 370
  - Resources, 372
  - RubberTile Resources, 372
  - XtNapplyCallback resource, 373
  - XtNapplyLabel resource, 373
  - XtNattributeListHeight resource, 374
  - XtNcancelCallback resource, 374
  - XtNcancelLabel resource, 375
  - XtNchangedCallback resource, 375
  - XtNcharsetInfo resource, 376
  - XtNerrorCallback resource, 376
  - XtNextensionArea resource, 377
  - XtNfontSearchSpec resource, 377
  - XtNinitialFontName resource, 378
  - XtNmaximumPointSize resource, 378
  - XtNnoPreviewText, 378
  - XtNpreferredPointSizes resource, 379
  - XtNpreviewBackground resource, 379
  - XtNpreviewBorderWidth resource, 379
  - XtNpreviewFontColor resource, 379
  - XtNpreviewForeground resource, 379
  - XtNpreviewHeight resource, 380
  - XtNpreviewPresent resource, 380
  - XtNpreviewSwitchLabel, 380
  - XtNpreviewSwitchOffLabel resource, 380
  - XtNpreviewSwitchOnLabel resource, 381
  - XtNpreviewText resource, 381
  - XtNrevertCallback resource, 381
  - XtNrevertLabel resource, 382
  - XtNsizeLabel resource, 382
  - XtNstyleLabel resource, 383
  - XtNtextFormat resource, 383
  - XtNtypefaceLabel resource, 383
- FontChooserShell, 385
  - Ancestry, 385
  - Appearance, 386

---

- Coloration, 386
- Components, 385
- Composite Resources, 387
- Core Resources, 386
- Resources, 389
- Shell Resources, 387
- TransientShell Resources, 389
- VendorShell Resources, 388
- WMShell Resources, 387
- XtNfontChooserWidget resource, 389
- XtNtextFormat resource, 390
- footer
  - left string, 55
  - left visibility, 55
  - right string, 57
  - right visibility, 57
- footer area resource, 54
- FooterPanel, 391
  - Activations Types, 394
  - Ancestry, 391
  - Coloration, 392
  - Composite Resources, 393
  - Core Resources, 392
  - Limitations, 392
  - Manager Resources, 393
  - Sizing, 392
- foreground color resource, 37
- Form, 395
  - Activation Types, 404
  - Ancestry, 395
  - Automatic Resizing, 396
  - Child Management, 396
  - Coloration, 396
  - Composite Resources, 399
  - Constraint Resources, 400
  - Core Resources, 399
  - Form Geometry Management
    - Algorithm, 397
  - Horizontal Constraints, 395
  - Manager Resources, 399
  - Reference\_Tree, 397
  - resize algorithm, 397
  - Resources, 400
  - Spanning Constraints, 395
  - Terminology, 397
- Vertical Constraints, 396
- XtNxAddWidth resource, 401
- XtNxAttachOffset resource, 401
- XtNxAttachRight resource, 401
- XtNxOffset resource, 402
- XtNxRefName resource, 402
- XtNxRefWidget resource, 403
- XtNxResizable resource, 403
- XtNxVaryOffset resource, 404
- XtNyAddHeight resource, 401
- XtNyAttachBottom resource, 401
- XtNyAttachOffset resource, 401
- XtNyOffset resource, 402
- XtNyRefName resource, 402
- XtNyRefWidget resource, 403
- XtNyResizable resource, 403
- XtNyVaryOffset resource, 404
- ForwardScanTextBuffer(), 175
- FreeBuffer(), 106
- FreeTextBuffer(), 175

**G**

- gadgets
  - registering help, 156
- Gauge, 405
  - Activation Types, 411
  - Ancestry, 405
  - Application Notification, 406
  - Coloration, 406
  - Components, 406
  - Core Resources, 407
  - Horizontal Orientation, 406
  - Primitive Resources, 407
  - Resources, 408
  - XtNleftMargin resource, 408
  - XtNmaxLabel resource, 408
  - XtNminLabel resource, 409
  - XtNorientation resource, 409
  - XtNrecomputeSize resource, 410
  - XtNrightMargin resource, 410
  - XtNsliderMax resource, 410
  - XtNsliderMin resource, 410
  - XtNsliderValue resource, 410
  - XtNspan resource, 411

---

XtNticks resource, 411  
XtNtickUnit resource, 411  
Gauge Function, 412  
    OlSetGaugeValue(), 412  
geometry manager  
    example, 148  
geometry requests, 42  
GetOlBusyCursor(), 115  
GetOlDataDupeDragCursor(), 115  
GetOlDataDupeDropCursor(), 115  
GetOlDataDupeInsertCursor(), 115  
GetOlDataDupeNoDropCursor(), 115  
GetOlDataMoveDragCursor(), 115  
GetOlDataMoveDropCursor(), 115  
GetOlDataMoveInsertCursor(), 115  
GetOlDataMoveNoDropCursor(), 115  
GetOlDocCursor(), 115  
GetOlDocStackCursor(), 115  
GetOlDropCursor(), 115  
GetOlDupeDocCursor(), 115  
GetOlDupeDocDragCursor(), 115  
GetOlDupeDocDropCursor(), 115  
GetOlDupeDocNoDropCursor(), 115  
GetOlDupeStackCursor(), 115  
GetOlDupeStackDragCursor(), 115  
GetOlDupeStackDropCursor(), 115  
GetOlDupeStackNoDropCursor(), 115  
GetOlDuplicateCursor(), 116  
GetOlFolderCursor(), 115  
GetOlFolderStackCursor(), 115  
GetOlMoveCursor(), 116  
GetOlMoveDocCursor(), 115  
GetOlMoveDocDragCursor(), 115  
GetOlMoveDocDropCursor(), 115  
GetOlMoveDocNoDropCursor(), 115  
GetOlMoveStackCursor(), 115  
GetOlMoveStackDragCursor(), 115  
GetOlMoveStackDropCursor(), 115  
GetOlMoveStackNoDropCursor(), 115  
GetOlNoDropCursor(), 115

GetOlPanCursor(), 116  
GetOlQuestionCursor(), 116  
GetOlStandardCursor(), 116  
GetOlTargetCursor(), 117  
GetOlTextDupeDragCursor(), 115  
GetOlTextDupeDropCursor(), 115  
GetOlTextDupeInsertCursor(), 115  
GetOlTextDupeNoDropCursor(), 115  
GetOlTextMoveDragCursor(), 115  
GetOlTextMoveDropCursor(), 115  
GetOlTextMoveInsertCursor(), 115  
GetOlTextMoveNoDropCursor(), 115  
GetTextBufferBlock(), 176  
GetTextBufferBuffer(), 176  
GetTextBufferChar(), 176  
GetTextBufferLine(), 177  
GetTextBufferLocation(), 177  
grabbing pointer events, 135  
grabs, avoiding permanent toolkit, 430  
graphics expose event, 273  
gravity resource for flat widgets, 63  
GrowBuffer(), 106

## H

header resource, 58  
height resource, 30  
help facility  
    Activation Type, 78  
    Displaying the Help Message, 159  
    Format of Help, 157  
    Help for Flat Widgets, 156  
    Help Key Event, 158  
    internationalization, 89  
    OlRegisterHelp(), 156  
    Static Variables, 159  
Help Function, 156  
help model resource, 21  
highlight color, 19  
highlighting keyboard mnemonics  
    resource, 26  
Hiragana, 91

---

## I

- icon mask resource, 46
- icon name resource, 61
- iconifying the base window, 60
- IM *See* input method
- IncrementTextBufferLocation(), 177
- Initialization and Activation Functions, 102
  - OIActivateWidget(), 103
  - OIAssociateWidget(), 103
  - OIInitialize(), 102
  - OIToolkitInitialize(), 102
  - OIUnassociateWidget(), 104
- input events
  - receiving, 28, 31
- input focus
  - acquiring, 23
  - application behavior, 48
  - color resource, 37
  - feedback resource, 21
  - manipulating, 160
  - moving, 79
  - selecting a window, 54
  - sensitivity, 31
- Input Focus Functions, 160
  - OICallAcceptFocus(), 160
  - OICanAcceptFocus(), 160
  - OIGetCurrentFocusWidget(), 161
  - OIHasFocus(), 161
  - OIMoveFocus(), 162
  - OISetInputFocus(), 161
- input method, 90
  - committing input, 91
  - default, 53
  - implicit commit, 91
  - screen regions, 92
  - status feedback, 54
  - status style, 95
- input mode, 91
- InsertIntoBuffer(), 106
- international OLIT
  - definition, 82
  - supported languages, 82
- internationalization
  - character encoding, 84
  - example, 97
  - font sets, 87
  - help facility, 89
  - input mode, 91
  - locale, 81
  - setting default text format, 86
  - setting locales, 83
  - standards, 99
  - supported languages, 82
  - text formats, 84
- Internationalization Features, 81
- Internationalization Functions
  - OIAllocateTextBuffer(), 186
  - OIBackwardScanTextBuffer(), 187
  - OICopyTextBufferBlock(), 188
  - OIEndCurrentTextBufferWord(), 188
  - OIForwardScanTextBuffer(), 189
  - OIFreeTextBuffer(), 190
  - OIGetTextBufferBlock(), 190
  - OIGetTextBufferBuffer(), 191
  - OIGetTextBufferCharAtLoc(), 192
  - OIGetTextBufferFileName(), 192
  - OIGetTextBufferLine(), 193
  - OIGetTextUndoDeleteItem(), 193
  - OIGetTextUndoInsertItem(), 194
  - OIIncrementTextBufferLocation(), 194
  - OIIsTextBufferEmpty(), 195
  - OIIsTextBufferModified(), 195
  - OILastCharInTextBufferLine(), 195
  - OILastTextBufferLine(), 196
  - OILastTextBufferLocation(), 196
  - OILastTextBufferPosition(), 197
  - OILineOfPosition(), 197
  - OILinesInTextBuffer(), 197
  - OILocationOfPosition(), 198
  - OINextLocation(), 199
  - OINextTextBufferWord(), 199
  - OINumBytesInTextBufferLine(), 200
  - OINumCharsInTextBufferLine(), 200
  - OINumUnitsInTextBufferLine(), 201
  - OIPositionOfLine(), 201
  - OIPositionOfLocation(), 201

---

- OlPreviousLocation(), 202
- OlPreviousTextBufferWord(), 202
- OlReadFileIntoTextBuffer(), 203
- OlReadStringIntoTextBuffer(), 204
- OlRegisterAllTextBufferScan-  
Functions(), 204
- OlRegisterAllTextBufferWord-  
Definition, 205
- OlRegisterPerTextBufferScan-  
Functions(), 205
- OlRegisterPerTextBufferWord-  
Definition(), 206
- OlRegisterTextBufferUpdate(), 207
- OlReplaceBlockInTextBuffer(), 207
- OlReplaceCharInTextBuffer(), 209
- OlSaveTextBuffer(), 210
- OlSetTextUndoDeleteItem(), 210
- OlSetTextUndoInsertItem(), 211
- OlStartCurrentTextBufferWord(), 211
- OlTextEditOlTextBuffer(), 212
- OlUnitOffsetOfLocation(), 212
- OlUnregisterTextBufferUpdate(), 213
- internationalizing applications, 81
- item list resource for flat widgets, 64
- items
  - flat widget, 331

## J

- Japanese Feature Package, 82
- Japanese phonetic alphabets, 91

## K

- Kanji, 91
- Katakana, 91
- key bindings in resource file, 74
- keyboard accelerators
  - binding in resource file, 74
  - KeyPress event, 35
  - resource, 26
  - string resource, 35
  - translations resource, 27
- keyboard input focus *See* input focus
- keyboard mnemonics

- resource, 26, 38
- keyboard repeat count, 24
- keyboard traversal
  - preventing, 103
  - resource, 40
- KeyPress event, 222
- KeyPress event resource, 35

## L

- label
  - default font, 36
- label resource for flat widgets, 67
- LANG
  - setting locale, 84
- LastTextBufferLocation(), 178
- LastTextBufferPosition(), 178
- leader widgets, 103
- LeaveNotify event, 222
- LineOfPosition(), 178
- locale, 81
  - setting, 83
- localization, 81
- localized messages, 90
- LocationOfPosition(), 178
- lookup choice region, 91
- LookupOlColors(), 150
- LookupOlInputEvent(), 217

## M

- Manager Resources, 41
  - BulletinBoard widget, 237
  - Caption widget, 242
  - CheckBox widget, 252
  - ControlArea widget, 261
  - DrawArea widget, 271
  - Exclusives widget, 290
  - FileChooser widget, 296
  - FileChooser widget, 371
  - FooterPanel widget, 393
  - Form widget, 399
  - Nonexclusives widget, 440



---

- RubberTile widget, 514
- ScrolledWindow widget, 544
- ScrollingList widget, 565
- TextField widget, 679
- mapping
  - widget, 31
  - X events, 32
- menu
  - cascade, 414
  - click-move-click, 422, 426
  - pinned, 427
  - popup, creating, 227
  - press-drag-release, 422, 426
  - slow clicking, 426
- menu button
  - presence in shell window header, 55
- menu mark width resource, 22
- MenuButton, 413
  - Activation Types, 421
  - Ancestry, 413
  - Appearance, 413
  - Coloration, 415
  - Components, 413
  - Core Resources, 417
  - Label Appearance, 416
  - Menu Placement, 415
  - Menu Previewing, 414
  - OL\_MENU Activation Type, 422
  - OL\_MENUDEFAULT Activation Type, 422
  - OL\_MENUDEFAULTKEY Activation Type, 422
  - OL\_MENUKEY Activation Type, 423
  - OL\_SELECT Activation Type, 423
  - OL\_SELECTKEY Activation Type, 423
  - Popping Up the MenuShell Subwidget, 414
  - Primitive Resources, 417
  - Resources, 418
  - Selecting the Default Item, 415
  - Subwidget, 414
  - Subwidgets, 414
  - XtNdefault resource, 419
  - XtNlabel resource, 419
  - XtNlabelImage resource, 419
  - XtNlabelJustify resource, 420
  - XtNlabelType resource, 420
  - XtNmenuMark resource, 420
  - XtNmenuPane resource, 421
  - XtNrecomputeSize resource, 421
- menumark, 413
- menupane, 425
- MenuPane Subwidget Resources, 435
- MenuShell, 424
  - Activation Types, 437
  - Ancestry, 424
  - Associating a Menu with a Widget, 426
  - Avoiding Permanent Toolkit Grabs, 430
  - Coloration, 428
  - Components, 424
  - Composite Resources, 432
  - Converting Stay-up to Pop-up Menu, 427
  - Core Resources, 432
  - Highlighting Items, 427
  - Keyboard Traversal, 431
  - Menu Components, 425
  - menupane, 425
  - OlMenuPopdown(), 430
  - OlMenuPopup(), 428
  - OlMenuPost(), 430
  - OlMenuUnpost(), 430
  - Popup Control, 426
  - Popup Position, 427
  - Programmatic Menu Popup and Popdown, 428
  - pushpin usage, 427
  - Resources, 435
  - Selection Control, 426
  - Shell Resources, 432
  - Subwidget Resources, 418
  - Subwidgets, 425
  - The Default Item, 427
  - TransientShell Resources, 434
  - VendorShell Resources, 434
  - WMSHELL Resources, 433
  - XtNmenuAugment resource, 435



---

- XtNmenuPane resource, 436
- XtNpushpin resource, 436
- XtNpushpinDefault resource, 436
- messages
  - localized, 90
  - protocol, 58, 59
- Meta key, 23
- meta-key, 22
- millimeters
  - converting from pixels, 168
- mnemonic
  - character prefix resource, 22
  - keyboard, 38
- mnemonics *See* keyboard mnemonics
- Mod5 key, 23
- ModeSwitch key, 23
- Modifier1 - Modifier5 key resources, 23
- MotionNotify event, 222
- mouse damping factor resource, 23
- mouseless operations resource, 23
- multibyte character text, 39
- multibyte encoding, 84
- multi-click timing resource, 24
- Multiple Visual Functions, 164
  - OlColormapOfObject(), 165
  - OlDepthOfObject(), 165
  - OlInternAtom(), 165
  - OlVisualOfObject(), 166
- multi-visual application, 164

## N

- NextLocation(), 179
- NextTextBufferWord(), 179
- Nonexclusives, 438
  - Activation Types, 442
  - Ancestry, 438
  - Buttons Example, 438
  - Coloration, 439
  - Composite Resources, 440
  - Core Resources, 440
  - Keyboard Traversal, 439
  - Manager Resources, 440

- menu use, 439
- menupane usage, 439
- Resources, 441
- Restrictions on Children, 439
- XtNlayoutType resource, 441
- XtNmeasure resource, 441
- XtNrecomputeSize resource, 442
- NoticeShell, 443
  - Activation Types, 390, 452
  - Ancestry, 443
  - Coloration, 446
  - Components, 443
  - Composite Resources, 447
  - ControlArea Subwidget Resources, 450
  - Core Resources, 446
  - emanate widget, 443
  - Keyboard Traversal, 445
  - Popping the Notice Up and Down, 444
  - Resources, 449
  - Shell Resources, 447
  - StaticText Subwidget Resources, 449
  - Subwidgets, 444
  - Text and ControlAreas, 445
  - TransientShell Resources, 449
  - VendorShell Resources, 448
  - WMShell Resources, 447
  - XtNcontrolArea resource, 450
  - XtNemanateWidget resource, 450
  - XtNpointerWarping resource, 451
  - XtNtextArea resource, 451
  - XtNtextFormat resource, 451
- numeric values
  - displaying graphically, 405
- NumericField, 453
  - Activation Types, 467
  - Ancestry, 453
  - Caret Position, 455
  - Coloration, 456
  - Components, 453, 454
  - Core Resources, 456
  - DeltaButtons, 454, 456
  - Display of Text, 455
  - Implementing New Datatypes, 465

---

Keyboard Accelerator Display, 455  
 Keyboard Mnemonic Display, 455  
 Keyboard Traversal, 454  
 OL\_CHARBAK Activation Type, 468  
 OL\_CHARFWD Activation Type, 468  
 OL\_COPY Activation Type, 468  
 OL\_CUT Activation Type, 468  
 OL\_DELCHARBAK Activation Type, 469  
 OL\_DELCHARFWD Activation Type, 469  
 OL\_DELLINE Activation Type, 470  
 OL\_DELLINEBAK Activation Type, 470  
 OL\_DELLINEFWD Activation Type, 470  
 OL\_DELWORDBAK Activation Type, 471  
 OL\_DELWORDFWD Activation Type, 471  
 OL\_LINEEND Activation Type, 471  
 OL\_LINESTART Activation Type, 471  
 OL\_NEXTFIELD Activation Type, 472  
 OL\_PASTE Activation Type, 472  
 OL\_PREVFIELD Activation Type, 472  
 OL\_UNDO Activation Type, 472  
 OL\_WORDBAK Activation Type, 473  
 OL\_WORDFWD Activation Type, 473  
 Per-Key and Per-Field Validation, 464  
 Primitive Resources, 457  
 Resources, 458  
 Selection Of Text, 455  
 TextLine Resources, 458  
 validation, 464  
 XtNconvertProc resource, 459  
 XtNdelta resource, 459  
 XtNdeltaCallback resource, 460  
 XtNdeltaState resource, 460  
 XtNmaxValue resource, 461  
 XtNminValue resource, 461  
 XtNsizeOf resource, 462  
 XtNtype resource, 462  
 XtNvalidateCallback resource, 462  
 XtNvalue resource, 463  
 NumLock key, 23

**O**

OblongButton, 474  
   Activation Types, 483  
   Ancestry, 474  
   Busy Indication During Callbacks, 474  
   Coloration, 476  
   Components, 474  
   Core Resources, 479  
   in popup menus, 475  
   in PopupWindowShells, 496  
   Keyboard Accelerator Display, 477  
   Keyboard Mnemonic Display, 477  
   Keyboard Traversal, 476  
   Label Appearance, 478  
   OblongButton Gadgets, 475  
   OblongButtons Not In Popup Menus, 475  
   OL\_MENU Activation Type, 484  
   OL\_MENUDEFAULT Activation Type, 484  
   OL\_MENUDEFAULTKEY Activation Type, 484  
   OL\_MENUEXKEY Activation Type, 484  
   OL\_SELECT Activation Type, 484  
   OL\_SELECTKEY Activation Type, 484  
   Primitive Resources, 479  
   Resources, 480  
   XtNbusy resource, 480  
   XtNdefault resource, 481  
   XtNlabel resource, 481  
   XtNlabelImage resource, 481  
   XtNlabelJustify resource, 482  
   XtNlabelTile resource, 482  
   XtNlabelType resource, 482  
   XtNrecomputeSize resource, 482  
   XtNselect resource, 483  
 OL\_ADJUST  
   ScrollingList widget, 585  
   StaticText widget, 618  
   TextEdit widget, 654  
   TextField widget, 685

---

OL\_ADJUSTKEY  
     ScrollingList widget, 585

OL\_CHARBAK  
     NumericField widget, 468  
     TextEdit widget, 654  
     TextField widget, 685  
     TextLine widget, 712

OL\_CHARFWD  
     NumericField widget, 468  
     TextEdit widget, 654  
     TextField widget, 686  
     TextLine widget, 712

OL\_COPY  
     NumericField widget, 468  
     ScrollingList widget, 585  
     StaticText widget, 618  
     TextEdit widget, 655  
     TextField widget, 686  
     TextLine widget, 712

OL\_CORE\_IE, 223

OL\_CUT  
     NumericField widget, 468  
     ScrollingList widget, 585  
     TextEdit widget, 655  
     TextField widget, 686  
     TextLine widget, 712

OL\_DEFAULT\_IE, 223

OL\_DELCHARBAK  
     NumericField widget, 469  
     TextEdit widget, 655  
     TextField widget, 686  
     TextLine widget, 712

OL\_DELCHARFWD  
     NumericField widget, 469  
     TextEdit widget, 656  
     TextField widget, 687  
     TextLine widget, 713

OL\_DELLINE  
     NumericField widget, 470  
     TextEdit widget, 656  
     TextField widget, 687  
     TextLine widget, 713

OL\_DELLINEBAK  
     NumericField widget, 470  
     TextEdit widget, 657  
     TextField widget, 687  
     TextLine widget, 713

OL\_DELLINEFWD  
     NumericField widget, 470  
     TextEdit widget, 657  
     TextField widget, 688  
     TextLine widget, 714

OL\_DELWORDBAK  
     NumericField widget, 471  
     TextEdit widget, 658  
     TextField widget, 688  
     TextLine widget, 714

OL\_DELWORDFWD  
     NumericField widget, 471  
     TextEdit widget, 658  
     TextField widget, 689  
     TextLine widget, 714

OL\_DOCEND  
     TextEdit widget, 659

OL\_DOCSTART  
     TextEdit widget, 659

OL\_HSBMENU  
     Scrollbar widget, 536  
     ScrolledWindow widget, 554

OL\_LINEEND  
     NumericField widget, 471  
     TextEdit widget, 659  
     TextField widget, 689  
     TextLine widget, 715

OL\_LINESTART  
     NumericField widget, 471  
     TextEdit widget, 660  
     TextField widget, 689  
     TextLine widget, 715

OL\_MENU  
     AbbrevMenuButton widget, 233  
     FlatExclusives widget, 355  
     FlatNonexclusives widget, 362  
     MenuButton widget, 422  
     OblongButton widget, 484  
     RectButton widget, 510  
     Scrollbar widget, 536  
     ScrollingList widget, 586

---

TextEdit widget, 660  
 TextField widget, 689  
 OL\_MENUEFAULT  
   FlatExclusives widget, 355  
   FlatNonexclusives widget, 363  
   MenuButton widget, 422  
   OblongButton widget, 484  
   RectButton widget, 510  
 OL\_MENUEFAULTKEY  
   FlatExclusives widget, 355  
   FlatNonexclusives widget, 363  
   MenuButton widget, 422  
   OblongButton widget, 484  
   RectButton widget, 510  
 OL\_MENUKEY  
   AbbrevMenuButton widget, 234  
   FlatExclusives widget, 355  
   FlatNonexclusives widget, 362  
   MenuButton widget, 423  
   OblongButton widget, 484  
   RectButton widget, 510  
   Scrollbar widget, 536  
   ScrollingList widget, 586  
   TextEdit widget, 660  
   TextField widget, 689  
 OL\_MMToPixel, 168  
 OL\_MULTIDOWN  
   ScrollingList widget, 586  
 OL\_MULTIUP  
   ScrollingList widget, 586  
 OL\_NEXTFIELD  
   NumericField widget, 472  
   TextLine widget, 715  
 OL\_PAGEDOWN  
   Scrollbar widget, 536  
   ScrolledWindow widget, 555  
   ScrollingList widget, 586  
   TextEdit widget, 660  
 OL\_PAGELEFT  
   Scrollbar widget, 536  
   ScrolledWindow widget, 555  
   TextEdit widget, 660  
 OL\_PAGERIGHT  
   Scrollbar widget, 536  
   ScrolledWindow widget, 555  
   TextEdit widget, 661  
 OL\_PAGEUP  
   Scrollbar widget, 537  
   ScrolledWindow widget, 555  
   ScrollingList widget, 586  
   TextEdit widget, 661  
 OL\_PANEEND  
   ScrollingList widget, 586  
   TextEdit widget, 661  
 OL\_PANESTART  
   ScrollingList widget, 586  
   TextEdit widget, 661  
 OL\_PASTE  
   NumericField widget, 472  
   ScrollingList widget, 586  
   TextEdit widget, 662  
   TextField widget, 690  
   TextLine widget, 715  
 OL\_PixelToMM, 168  
 OL\_PixelToPoint, 168  
 OL\_PointToPixel, 168  
 OL\_PREVFIELD  
   NumericField widget, 472  
   TextLine widget, 716  
 OL\_RETURN  
   TextField widget, 690  
 OL\_ROWDOWN  
   TextEdit widget, 662  
 OL\_ROWUP  
   TextEdit widget, 662  
 OL\_SB\_STR\_REP, 87  
 OL\_ScreenMMToPixel, 168  
 OL\_ScreenPixelToMM, 168  
 OL\_ScreenPixelToPoint, 168  
 OL\_ScreenPointToPixel, 168  
 OL\_SCROLLBOTTOM  
   Scrollbar widget, 537  
   ScrolledWindow widget, 555  
   ScrollingList widget, 586  
   Slider widget, 608  
 OL\_SCROLLDOWN  
   Scrollbar, 537

---

- Scrollbar widget, 537
- ScrolledWindow widget, 555
- ScrollingList widget, 586
- Slider widget, 608
- TextEdit widget, 663
- OL\_SCROLLLEFT
  - ScrolledWindow widget, 555
  - Slider widget, 608
  - TextEdit widget, 663
  - TextField widget, 690
- OL\_SCROLLLEFTEDGE
  - Scrollbar widget, 537
  - ScrolledWindow widget, 556
  - Slider widget, 608
  - TextEdit widget, 663
  - TextField widget, 690
- OL\_SCROLLRIGHT
  - Scrollbar widget, 538
  - ScrolledWindow widget, 556
  - Slider widget, 609
  - TextEdit widget, 663
  - TextField widget, 690
- OL\_SCROLLRIGHTEDGE
  - Scrollbar widget, 537
  - ScrolledWindow widget, 556
  - Slider widget, 609
  - TextEdit widget, 663
  - TextField widget, 690
- OL\_SCROLLTOP
  - Scrollbar widget, 537
  - ScrolledWindow widget, 556
  - ScrollingList widget, 587
  - Slider widget, 609
- OL\_SCROLLUP
  - Scrollbar widget, 538
  - ScrolledWindow widget, 556
  - ScrollingList widget, 587
  - Slider widget, 609
  - TextEdit widget, 664
- OL\_SELCHARBAK
  - TextEdit widget, 664
  - TextField widget, 690
- OL\_SELCHARFWD
  - TextEdit widget, 664
- TextField widget, 691
- OL\_SELECT
  - AbbrevMenuButton widget, 234
  - Caption widget, 246
  - CheckBox widget, 258
  - FlatCheckBox widget, 346
  - FlatExclusives widget, 356
  - FlatNonexclusives widget, 363
  - MenuButton widget, 423
  - OblongButton widget, 484
  - RectButton widget, 511
  - Scrollbar widget, 538
  - ScrollingList widget, 587
  - Slider widget, 609
  - StaticText widget, 618
  - TextEdit widget, 665
  - TextField widget, 691
- OL\_SELECTKEY
  - AbbrevMenuButton widget, 234
  - Caption widget, 246
  - CheckBox widget, 258
  - FlatCheckBox widget, 346
  - FlatExclusives widget, 356
  - FlatNonexclusives widget, 363
  - MenuButton widget, 423
  - OblongButton widget, 484
  - RectButton widget, 511
  - ScrollingList widget, 587
- OL\_SELFLIPENDS
  - TextEdit widget, 665
  - TextField widget, 691
- OL\_SELLINE
  - TextEdit widget, 665
  - TextField widget, 691
- OL\_SELLINEBAK
  - TextEdit widget, 666
  - TextField widget, 692
- OL\_SELLINEFWD
  - TextEdit widget, 666
  - TextField widget, 692
- OL\_SELWORDBAK
  - TextEdit widget, 666
  - TextField widget, 692
- OL\_SELWORDFWD

---

TextEdit widget, 667  
 TextField widget, 693  
 OL\_TEXT\_IE, 223  
 OL\_UNDO  
   NumericField widget, 472  
   TextEdit widget, 667  
   TextField widget, 693  
   TextLine widget, 716  
 OL\_VSBMENU  
   Scrollbar widget, 536  
   ScrolledWindow widget, 556  
 OL\_WORDBAK  
   NumericField widget, 473  
   TextEdit widget, 668  
   TextField widget, 693  
   TextLine widget, 716  
 OL\_WORDFWD  
   NumericField widget, 473  
   TextEdit widget, 668  
   TextField widget, 693  
   TextLine widget, 716  
 OIActivateWidget(), 103  
 OIAddCallback(), 36, 59  
 OIAllocateTextBuffer(), 186  
 OIAppAddItemProc, 567  
 OIAppDeleteItemProc, 568  
 OIAppEditCloseProc, 569  
 OIAppEditOpenProc, 570  
 OIAppTouchItemProc, 571  
 OIAppUpdateViewProc, 571  
 OIAppViewItemProc, 572  
 OIAssociateWidget(), 73, 103  
 OIBackwardScanTextBuffer(), 187  
 OIBlackPixel, 164  
 OICallAcceptFocus(), 160  
 OICallDynamicCallbacks(), 150  
 OICanAcceptFocus(), 160  
 OIClassSearchIEDB(), 219  
 OIClassSearchTextDB(), 220  
 OIColormapOfObject(), 165  
 OICopyTextBufferBlock(), 188  
 OICreateInputEventDB(), 221  
 OICreatePackedWidgetList(), 166  
 OIDefaultDisplay, 168  
 OIDefaultScreen, 168  
 OIDepthOfObject(), 165  
 OIDetermineMouseAction(), 217  
 OIDnDAllocateTransientAtom(), 121  
 OIDnDAllocTransientAtom(), 132  
 OIDnDBeginSelectionTransaction(), 138  
 OIDnDChangeDropSitePreviewHints(),  
   139  
 OIDnDClearDragState(), 133  
 OIDnDDeliverPreviewMessage(), 133  
 OIDnDDeliverTriggerMessage(), 133  
 OIDnDDestroyDropSite(), 139  
 OIDnDDisownSelection(), 134  
 OIDnDDragAndDrop(), 134  
 OIDnDDragNDropDone(), 139  
 OIDnDDragNDropInfo structure, 135  
 OIDnDDropSiteID(), 127  
 OIDnDEndSelectionTransaction(), 140  
 OIDnDErrorDuringSelection-  
   Transaction(), 141  
 OIDnDFreeTransientAtom(), 135  
 OIDnDGetCurrentSelectionsForWidget(),  
   141  
 OIDnDGetDropSitesOfWidget(), 142  
 OIDnDGetDropSitesOfWindow(), 142  
 OIDnDGetWidgetOfDropSite(), 143  
 OIDnDGetWindowOfDropSite(), 143  
 OIDnDInitializeDragState(), 136  
 OIDnDOwnSelection(), 136  
 OIDnDOwnSelectionIncremental(), 136  
 OIDnDPMNotifyProc(), 129  
 OIDnDPreviewAndAnimate(), 137  
 OIDnDPreviewAnimateCbP(), 128  
 OIDnDProtocolActionCbP(), 129  
 OIDnDQueryDropSiteInfo(), 144  
 OIDnDRegisterWidgetDropSite(), 145  
 OIDnDRegisterWindowDropSite(), 146  
 OIDnDSetDropSiteInterest(), 146

---

OIdnDSetInterestInWidgetHier(), 147  
 OIdnDSitePreviewBoth, 128  
 OIdnDSitePreviewDefaultSite, 128  
 OIdnDSitePreviewEnterLeave, 127  
 OIdnDSitePreviewForwarded, 128  
 OIdnDSitePreviewHints(), 127  
 OIdnDSitePreviewInsensitive, 128  
 OIdnDSitePreviewMotion, 128  
 OIdnDSitePreviewNone, 127  
 OIdnDSiteRect(), 127  
 OIdnDTMNotifyProc(), 131  
 OIdnDTransactionStateCallback(), 130  
 OIdnDUpdateDropSiteGeometry(), 147  
 OIdnDWidgetConfiguredInHier(), 147  
 OIDrawAreaCallbackStruct structure, 273  
 OI DropTargetCallbackStruct Field  
     Validity, 280  
 OI DropTargetCallbackStruct structure,  
     279  
 OIEndCurrentTextBufferWord(), 188  
 OIError(), 152  
 OIErrorHandler(), 154  
 OIFCApplyCallbackStruct structure, 373  
 OIFCCancelCallbackStruct structure, 374  
 OIFCChangedCallbackStruct structure,  
     375  
 OIFCErrorCallbackStruct structure, 376  
 OIFCRevertCallbackStruct structure, 381  
 OIFiChFilterCallbackStruct structure, 307  
 OIFileChDocumentCallbackStruct  
     structure, 304, 306  
 OIFileChFolderCallbackStruct structure,  
     305  
 OIFileChGenericCallbackStruct  
     structure, 302  
 OIFileChListChoiceCallbackStruct  
     structure, 317  
 OIFileChShVerifyCallbackStruct structure,  
     329  
 OIFlatCallAcceptFocus(), 364  
 OIFlatCallData structure, 335  
 OIFlatGetFocusItem(), 364  
 OIFlatGetItemGeometry(), 365  
 OIFlatGetItemIndex(), 364  
 OIFlatGetValues(), 365  
 OIFlatHelpId structure, 156, 338  
 OIFlatSetValues(), 366  
 OIFNavNode structure type, 304  
 OIFolderList, 309  
 OIFont, 87  
 OIForwardScanTextBuffer(), 189  
 OIFreeTextBuffer(), 190  
 OIGet50PercentGrey(), 117  
 OIGet75PercentGrey(), 117  
 OIGetApplicationResources(), 150  
 OIGetApplicationValues(), 17, 216  
 OIGetCurrentFocusWidget(), 161  
 OIGetDataDupeDragCursor(), 109  
 OIGetDataDupeDropCursor(), 109  
 OIGetDataDupeInsertCursor(), 109  
 OIGetDataDupeNoDropCursor(), 110  
 OIGetDataMoveDragCursor(), 110  
 OIGetDataMoveDropCursor(), 110  
 OIGetDataMoveInsertCursor(), 110  
 OIGetDataMoveNoDropCursor(), 110  
 OIGetDefaultFont(), 88  
 OIGetDocCursor(), 110  
 OIGetDocStackCursor(), 110  
 OIGetDropCursor(), 111  
 OIGetDupeDocCursor(), 111  
 OIGetDupeDocDragCursor(), 111  
 OIGetDupeDocDropCursor(), 111  
 OIGetDupeDocNoDropCursor(), 111  
 OIGetDupeStackCursor(), 111  
 OIGetDupeStackDragCursor(), 111  
 OIGetDupeStackDropCursor(), 112  
 OIGetDupeStackNoDropCursor(), 112  
 OIGetFolderCursor(), 112  
 OIGetFolderStackCursor(), 112  
 OIGetMoveDocCursor(), 112  
 OIGetMoveDocDragCursor(), 112



---

OIGetMoveDocDropCursor(), 112  
 OIGetMoveDocNoDropCursor(), 113  
 OIGetMoveStackCursor(), 113  
 OIGetMoveStackDragCursor(), 113  
 OIGetMoveStackDropCursor(), 113  
 OIGetMoveStackNoDropCursor(), 113  
 OIGetNoDropCursor(), 113  
 OIGetTextBufferBlock(), 190  
 OIGetTextBufferBuffer(), 191  
 OIGetTextBufferCharAtLoc(), 192  
 OIGetTextBufferFileName(), 192  
 OIGetTextBufferLine(), 193  
 OIGetTextDupeDragCursor(), 113  
 OIGetTextDupeDropCursor(), 114  
 OIGetTextDupeInsertCursor(), 114  
 OIGetTextDupeNoDropCursor(), 114  
 OIGetTextMoveDragCursor(), 114  
 OIGetTextMoveDropCursor(), 114  
 OIGetTextMoveInsertCursor(), 114  
 OIGetTextMoveNoDropCursor(), 114  
 OIGetTextUndoDeleteItem(), 193  
 OIGetTextUndoInsertItem(), 194  
 OIGrabDragPointer(), 121, 137  
 OIHasFocus(), 161  
 OIIncrementTextBufferLocation(), 194  
 OIInitialize(), 102  
 OIInputCallData structure, 642  
 OIInputEvent enumerated type, 642  
 OIInternAtom(), 165  
 OIIsTextBufferEmpty(), 195  
 OIIsTextBufferModified(), 195  
 OLIT Release 3.2 differences, 11  
 OLIT Toolkit Resources, 17  
     XtNbeep, 18  
     XtNbeepVolume, 19  
     XtNcolorTupleList, 19  
     XtNcontrolName, 20  
     XtNdragRightDistance, 20  
     XtNgrabPointer, 20  
     XtNhelpModel, 21  
     XtNinputFocusFeedback, 21  
     XtNlockName, 21  
     XtNmenuMarkRegion, 22  
     XtNmnemonicPrefix, 22  
     XtNmod1Name, 23  
     XtNmod2Name, 23  
     XtNmod3Name, 23  
     XtNmod4Name, 23  
     XtNmod5Name, 23  
     XtNmouseDampingFactor, 23  
     XtNmouseless, 23  
     XtNmultiClickTimeout, 24  
     XtNmultiObjectCount, 24  
     XtNolDefaultFont, 24  
     XtNscale, 25  
     XtNselectDoesPreview, 25  
     XtNshiftName, 25  
     XtNshowAccelerators, 26  
     XtNshowMnemonics, 26  
     XtNthreeD, 26  
 OIKeyOrBtnRec structure, 221  
 OILastCharInTextBufferLine(), 195  
 OILastTextBufferLine(), 196  
 OILastTextBufferLocation(), 196  
 OILastTextBufferPosition(), 197  
 OILineOfPosition(), 197  
 OILinesInTextBuffer(), 197  
 OIListDelete structure, 578  
 OIListItem structure, 583  
 OILocationOfPosition(), 198  
 OILookupInputEvent(), 222  
 OIMMToPixel, 168  
 OIMoveFocus(), 162  
 OINextLocation(), 199  
 OINextTextBufferWord(), 199  
 OINFDeltaCallbackStruct structure, 460  
 OINFFValidateCallbackStruct structure, 463  
 OINumBytesInTextBufferLine(), 200  
 OINumCharsInTextBufferLine(), 200  
 OINumUnitsInTextBufferLine(), 201  
 OIPackedWidget structure, 167  
 OIPixelToMM, 168



---

OIPixelToPoint, 168  
 OIPointToPixel, 168  
 OIPositionOfLine(), 201  
 OIPositionOfLocation(), 201  
 OIPreviousLocation(), 202  
 OIPreviousTextBufferWord(), 202  
 OIReadFileIntoTextBuffer(), 203  
 OIReadStringIntoTextBuffer(), 204  
 OIRegisterAllTextBufferScanFunctions(), 204  
 OIRegisterDynamicCallback(), 150  
 OIRegisterHelp(), 89, 156  
 OIRegisterPerTextBufferScanFunctions(), 205  
 OIRegisterPerTextBufferWordDefinition(), 206  
 OIRegisterTextBufferUpdate(), 207  
 OIReplaceBlockInTextBuffer(), 207  
 OIReplaceCharInTextBuffer(), 209  
 OIReplayBtnEvent(), 219  
 OISaveTextBuffer(), 210  
 OIScreenMMToPixel, 168  
 OIScreenPixelToMM, 168  
 OIScreenPixelToPoint, 168  
 OIScreenPointToPixel, 168  
 OIScrollbarVerify structure, 533  
 OISetApplicationValues(), 17, 216  
 OISetDefaultTextFormat(), 40, 86  
 OISetErrorHandler(), 153  
 OISetGaugeValue(), 412  
 OISetInputFocus(), 161  
 OISetTextUndoDeleteItem(), 210  
 OISetTextUndoInsertItem(), 211  
 OISetVaDisplayErrorMsgHandler(), 154  
 OISetVaDisplayWarningMsgHandler(), 154  
 OISetWarningHandler(), 153  
 OISliderVerify structure, 606  
 OIListAddItem(), 588  
 OIListCallbackStruct structure, 581  
 OIListDeleteAllItems(), 588  
 OIListDeleteItem(), 589  
 OIListDeleteItems(), 589  
 OIListEditItem(), 589  
 OIListEndEdit(), 589  
 OIListFirstViewableItem(), 590  
 OIListGetItemAttrs(), 590  
 OIListGetItemImage(), 590  
 OIListGetItemLabel(), 590  
 OIListGetItemSensitivity(), 591  
 OIListGetItemType(), 591  
 OIListGetItemUserData(), 591  
 OIListGetMode(), 591  
 OIListGetNextItem(), 592  
 OIListGetPrevItem(), 592  
 OIListIsItemCurrent(), 592  
 OIListIsValidItem(), 593  
 OIListItemAttrs structure, 582  
 OIListLastViewableItem(), 593  
 OIListMakeAllItemsNotCurrent(), 593  
 OIListMakeItemCurrent(), 593  
 OIListMakeItemNotCurrent(), 594  
 OIListSetItemAttrs(), 594  
 OIListTouchItem(), 594  
 OIListUpdateView(), 595  
 OIListUserDeleteCallbackStruct structure, 578  
 OIListViewItem(), 595  
 OIStartCurrentTextBufferWord(), 211  
 OIStr type, 85  
 OIStrScanDefFunc(), 205, 206  
 OIStrWordDefFunc(), 205, 206  
 OISWGeometries structure, 547  
 OITextBufferPtr type, 672  
 OITextEditClearBuffer(), 669  
 OITextEditCopyBuffer(), 669  
 OITextEditCopySelection(), 670  
 OITextEditGetCursorPosition(), 670  
 OITextEditGetLastPosition(), 671  
 OITextEditInsert(), 673  
 OITextEditMoveDisplayPosition(), 671

---

OITextEditOITextBuffer(), 212, 672  
 OITextEditPaste(), 673  
 OITextEditReadSubString(), 669  
 OITextEditRedraw(), 670  
 OITextEditSetCursorPosition(), 671  
 OITextEditTextBuffer(), 672  
 OITextEditUpdate(), 673  
 OITextFieldCopyOIString(), 695  
 OITextFieldCopyString(), 695  
 OITextFieldGetOIString(), 695  
 OITextFieldGetString(), 696  
 OITextFieldVerify structure, 683  
 OITextMarginCallData structure, 647  
 OITextMarginHint enumerated type, 647  
 OITextModifyCallData structure, 647  
 OITextMotionCallData structure, 648  
 OITextPostModifyCallData structure, 649  
 OITLCommitCallbackStruct structure, 704  
 OITLGetPosition(), 717  
 OITLGetSelection(), 717  
 OITLGetSubString(), 718  
 OITLMotionCallbackStruct structure, 707  
 OITLOperateOnSelection(), 718  
 OITLPostModifyCallbackStruct structure, 709  
 OITLPreModifyCallbackStruct structure, 708  
 OITLSetSelection(), 719  
 OITLSetSubString(), 719  
 OIToolkitInitialize(), 41, 102  
 OIUnassociateWidget(), 104  
 OIUngrabDragPointer(), 138  
 OIUnitOffsetOfLocation(), 212  
 OIUnregisterDynamicCallback(), 151  
 OIUnregisterTextBufferUpdate(), 213  
 OIUpdateDisplay(), 118  
 OIVaDisplayErrorMsg(), 152  
 OIVaDisplayErrorMsgHandler(), 154  
 OIVaDisplayWarningMsg(), 153  
 OIVaDisplayWarningMsgHandler(), 155  
 OIVaFlatGetValues(), 365  
 OIVaFlatSetValues(), 366  
 OIVirtualEventRec, 36  
 OIVirtualEventRec structure, 223  
 OIVirtualEventTable, 221  
 OIVisualOfObject(), 166  
 OIWarning(), 152  
 OIWarningHandler(), 154  
 OIWhitePixel, 166  
 OIWidgetSearchIEDB(), 225  
 OIWidgetSearchTextDB(), 226  
 OIWMProtocolAction(), 170  
 OIWMProtocolVerify structure, 59, 170  
 OPEN LOOK TEXT database, 220

**P**

Packed Widget Function, 166  
 Packed Widget Functions  
   OICreatePackedWidgetList(), 166  
 pasting text, 215  
 pinned menu, 427  
 pixel  
   number of bits per, 30  
 Pixel Conversion Functions, 168  
   Screen Selection, 169  
 pixmap  
   background resource, 28  
   border resource, 29  
 pixmap functions, *See* Cursor and Pixmap Functions, 109  
 Pixmap Resources  
   DropTarget widget, 278  
 point size resource, 39  
 point size scaling resource, 25  
 pointer grabbing, 218  
   resource, 20  
 points  
   converting from pixels, 168  
 popdown callbacks, 43  
 popup

---

- callback, 43
- function resource, 42
- position, 43
- window title, 49
- PopupWindowShell, 485
  - Activation Types, 497
  - Ancestry, 485
  - Automatic Addition of Buttons,
    - Settings Menu, 487
  - Coloration, 489
  - Components, 486
  - Composite Resources, 490
  - Control Areas, 488
  - ControlArea Subwidget Resources,
    - 494
  - Core Resources, 490
  - Default Window Decorations, 485
  - Keyboard Traversal, 489
  - Popping the Window Up and Down,
    - 488
  - Resources, 493
  - Shell Resources, 491
  - Subwidgets, 487
  - TransientShell Resources, 493
  - Traversable Components, 489
  - VendorShell Resources, 492
  - WMSHELL Resources, 491
  - XtNapply resource, 494
  - XtNapplyLabel resource, 495
  - XtNapplyMnemonic resource, 495
  - XtNfooterPanel resource, 495
  - XtNlowerControlArea resource, 496
  - XtNmenuTitle resource, 496
  - XtNpointerWarping resource, 496
  - XtNreset resource, 494
  - XtNresetFactory resource, 494
  - XtNresetFactoryLabel resource, 495
  - XtNresetFactoryMnemonic resource,
    - 495
  - XtNresetLabel resource, 495
  - XtNresetMnemonic resource, 495
  - XtNsetDefaults resource, 494
  - XtNsetDefaultsLabel resource, 495
  - XtNsetDefaultsMnemonic resource,
    - 495
  - XtNupperControlArea resource, 496
  - XtNverify resource, 496
- position resource, 32
- PositionOfLine(), 179
- PositionOfLocation(), 179
- pre-edit region, 91
- press-drag-release menu, 426
- press-drag-release mode, 422
- previewing submenus, 25
- previewing the default menu item, 229
- PreviousLocation(), 180
- PreviousTextBufferWord(), 180
- Primitive Resources, 34
  - AbbrevMenuButton widget, 231
  - DropTarget widget, 278
  - FlatCheckBox widget, 343
  - FlatExclusives widget, 350
  - FlatNonexclusives widget, 360
  - Gauge widget, 407
  - MenuButton widget, 417
  - NumericField widget, 457
  - OblongButton widget, 479
  - RectButton widget, 503
  - Scrollbar widget, 525
  - Slider widget, 600
  - StaticText widget, 612
  - Stub widget, 622
  - TextEdit widget, 639
  - TextLine widget, 701
  - XtNaccelerator, 35
  - XtNacceleratorText, 35
  - XtNconsumeEvent, 36
  - XtNfont, 36
  - XtNfontColor, 37
  - XtNforeground, 37
  - XtNinputFocusColor, 37
  - XtNmnemonic, 38
  - XtNreferenceName, 38
  - XtNreferenceWidget, 39
  - XtNscale, 39
  - XtNtextFormat, 39
  - XtNtraversalOn, 40
  - XtNuserData, 40
- Protocol Function, 170

---

Protocol Functions  
     OIWMPProtocolAction(), 170  
 protocol messages, 58  
     selecting, 59  
 pushpin  
     toggling, 79  
 pushpin resource, 56

**R**

ReadFileIntoBuffer(), 107  
 ReadFileIntoTextBuffer(), 180  
 ReadStringIntoBuffer(), 107  
 ReadStringIntoTextBuffer(), 181  
 RectButton, 499  
     Activation Types, 509  
     Ancestry, 499  
     Border Resource Interactions, 508  
     Borders, 508  
     Coloration, 501  
     Components, 500  
     Core Resources, 503  
     Keyboard Accelerator Display, 502  
     Keyboard Mnemonic Display, 502  
     Keyboard Traversal, 501  
     Label Appearance, 509  
     Label Resource Interactions, 508  
     Non-Popup Menu RectButtons, 501  
     OL\_MENU Activation Type, 510  
     OL\_MENUDEFAULT Activation Type, 510  
     OL\_MENUDEFAULTKEY Activation Type, 510  
     OL\_MENUKEY Activation Type, 510  
     OL\_SELECT Activation Type, 511  
     OL\_SELECTKEY Activation Type, 511  
     Popup Menu RectButtons, 500  
     Primitive Resources, 503  
     Rectangular Buttons, 500  
     Resources, 504  
     XtNdefault resource, 504  
     XtNdim resource, 505  
     XtNlabel resource, 505  
     XtNlabelImage resource, 505  
     XtNlabelJustify resource, 506  
     XtNlabelTile resource, 506  
     XtNlabelType resource, 506  
     XtNrecomputeSize resource, 506  
     XtNselect resource, 507  
     XtNset resource, 507  
     XtNunselect resource, 507  
 RegisterTextBufferScanFunctions(), 181  
 RegisterTextBufferUpdate(), 182  
 RegisterTextBufferWordDefinition(), 181  
 Regular Expression Functions, 171  
     streexp(), 171  
     strexpl(), 171  
     strexpr(), 172  
 Regular Expression Notation, 171  
 release differences, 11  
 remapping button events, 642  
 repeat count resource, 24  
 ReplaceBlockInTextBuffer(), 182  
 ReplaceCharInTextBuffer(), 184  
 resize corners, 57  
 resize policy of flat widgets, 68  
 resizing increment resource, 50  
 resizing Shell widgets resource, 43  
 resizing shells, 42  
 resource database  
     detecting dynamic changes, 150  
 resource file  
     bindings, 17  
     key bindings, 74  
 RESOURCE\_MANAGER, 150  
 resources  
     dynamic updating, 16  
 Resources Summary Tables, 16  
 resources summary tables, 16  
 rgb.txt file, 28, 29, 37, 38, 273, 549  
 Romaji, 91  
 RubberTile, 512  
     Activation Types, 517  
     Ancestry, 512  
     Coloration, 513  
     Composite Resources, 514

---

Constraint Resources, 515  
 Core Resources, 514  
 Manager Resources, 514  
 Resources, 515  
 XtNorientation resource, 515  
 XtNrefName resource, 516  
 XtNrefWidget resource, 516  
 XtNspace conflict with Caption, 516  
 XtNspace resource, 516  
 XtNweight resource, 517  
 RubberTile Resources  
   FileChooser widget, 296  
   FontChooser widget, 372

**S**

SaveTextBuffer(), 184  
 scale resource, 25, 39  
 ScrollingList Functions  
   known deficiencies, 595  
 screen resource, 31  
 Scrollbar, 518  
   Abbreviated Scrollbar, 520  
   Activation Types, 535  
   Ancestry, 518  
   Coloration, 524  
   Components, 518  
   Core Resources, 525  
   elevator, 518  
   Elevator at Limits, 522  
   Elevator Motion, 521  
   Horizontal Orientation, 519  
   Indicating View Proportion, 523  
   Keyboard Accelerator Display, 524  
   Keyboard Mnemonic Display, 524  
   Keyboard Traversal, 524  
   OL\_HSBMENU Activation Type, 536  
   OL\_MENU Activation Type, 536  
   OL\_MENUKEY Activation Type, 536  
   OL\_PAGEDOWN Activation Type, 536  
   OL\_PAGELEFT Activation Type, 536  
   OL\_PAGERIGHT Activation Type, 536  
   OL\_PAGEUP Activation Type, 537  
   OL\_SCROLLBOTTOM Activation Type, 537  
   OL\_SCROLLDOWN Activation Type, 537  
   OL\_SCROLLLEFTEDGE Activation Type, 537  
   OL\_SCROLLRIGHT Activation Type, 538  
   OL\_SCROLLRIGHTEDGE Activation Type, 537  
   OL\_SCROLLTOP Activation Type, 537  
   OL\_SCROLLUP Activation Type, 538  
   OL\_SELECT Activation Type, 538  
   OL\_VSBMENU Activation Type, 536  
   Primitive Resources, 525  
   Resources, 526  
   Scrollbar Menu, 523, 524  
   Scrolling a Pane, 522  
   Scrolling Limits, 522  
   Scrolling One Unit, 521  
   Scrolling Several Units, 521  
   Subwidget Resources, 527  
   Subwidgets, 519  
   Vertical Orientation, 520  
   XtNcurrentPage resource, 527  
   XtNdragCBType resource, 528  
   XtNgranularity resource, 528  
   XtNhereToLeftLabel resource, 528  
   XtNhereToLeftMnemonic resource, 529  
   XtNhereToTopLabel resource, 528  
   XtNhereToTopMnemonic resource, 529  
   XtNinitialDelay resource, 529  
   XtNleftToHereLabel resource, 528  
   XtNleftToHereMnemonic resource, 529  
   XtNmenuPane resource, 529  
   XtNmenuTitle resource, 530  
   XtNorientation resource, 530  
   XtNpointerWarping resource, 530  
   XtNpreviousLabel resource, 528  
   XtNpreviousMnemonic resource, 529  
   XtNproportionLength resource, 530  
   XtNrepeatRate resource, 531

---

XtNshowPage resource, 531  
 XtNsliderMax resource, 531  
 XtNsliderMin resource, 531  
 XtNsliderMoved resource, 532  
 XtNsliderValue resource, 534  
 XtNstopPosition resource, 534  
 XtNtopToHereLabel resource, 528  
 XtNtopToHereMnemonic resource, 529  
 XtNuseSetValCallback resource, 534  
 ScrolledWindow, 539  
   Activation Types, 553  
   Ancestry, 539  
   Application Controlled Scrolling, 542  
   Coloration, 543  
   Components, 539, 540  
   Composite Resources, 544  
   Content and View of Content, 541  
   Core Resources, 544  
   Keyboard Traversal, 543  
   Manager Resources, 544  
   OL\_HSBMENU Activation Type, 554  
   OL\_PAGEDOWN Activation Type, 555  
   OL\_PAGELEFT Activation Type, 555  
   OL\_PAGERIGHT Activation Type, 555  
   OL\_PAGEUP Activation Type, 555  
   OL\_SCROLLBOTTOM Activation Type, 555  
   OL\_SCROLLDOWN Activation Type, 555  
   OL\_SCROLLLEFT Activation Type, 555  
   OL\_SCROLLLEFTEDGE Activation Type, 556  
   OL\_SCROLLRIGHT Activation Type, 556  
   OL\_SCROLLRIGHTEDGE Activation Type, 556  
   OL\_SCROLLTOP Activation Type, 556  
   OL\_SCROLLUP Activation Type, 556  
   OL\_VSBMENU Activation Type, 556  
   Resources, 545  
   Scrollbars, 542  
   Subwidgets, 540, 541  
   Upper Left Corner Fixed on Resize, 541  
   View Border, 541  
   View Larger than Content, 542  
   view of the content, 541  
   XtNalignHorizontal resource, 546  
   XtNalignVertical resource, 546  
   XtNcomputeGeometries resource, 546  
   XtNcurrentPage resource, 548  
   XtNforceHorizontalSB resource, 548  
   XtNforceVerticalSB resource, 549  
   XtNforeground resource, 549  
   XtNhAutoScroll resource, 550  
   XtNhInitialDelay resource, 550  
   XtNhMenuPane resource, 550  
   XtNhRepeatRate resource, 551  
   XtNhScrollbar resource, 551  
   XtNhSliderMoved resource, 551  
   XtNhStepSize resource, 552  
   XtNinitialX resource, 552  
   XtNinitialY resource, 552  
   XtNrecomputeHeight resource, 553  
   XtNrecomputeWidth resource, 553  
   XtNshowPage resource, 553  
   XtNvAutoScroll resource, 550  
   XtNvInitialDelay resource, 550  
   XtNvMenuPane resource, 550  
   XtNvRepeatRate resource, 551  
   XtNvScrollbar resource, 551  
   XtNvSliderMoved resource, 551  
   XtNvStepSize resource, 552  
 ScrollingList, 557  
   Activation Types, 584  
   Ancestry, 557  
   Callback Information, 581  
   Coloration, 562  
   Components, 557, 558  
   Composite Resources, 565  
   Core Resources, 564  
   Deleting Selected Items, 560  
   Editable ScrollingList, 558  
   Editable Text Field, 559

---

Item Order, 560  
 Keyboard Mnemonic Display, 564  
 Keyboard Traversal, 563  
 known deficiencies, 595  
 Making an Item Current, 560  
 Manager Resources, 565  
 OL\_ADJUST Activation Type, 585  
 OL\_ADJUSTKEY Activation Type, 585  
 OL\_COPY Activation Type, 585  
 OL\_CUT Activation Type, 585  
 OL\_MENU Activation Type, 586  
 OL\_MENUKEY Activation Type, 586  
 OL\_MULTIDOWN Activation Type, 586  
 OL\_MULTIUP Activation Type, 586  
 OL\_PAGEDOWN Activation Type, 586  
 OL\_PAGEUP Activation Type, 586  
 OL\_PANEEND Activation Type, 586  
 OL\_PANESTART Activation Type, 586  
 OL\_PASTE Activation Type, 586  
 OL\_SCROLLBOTTOM Activation Type, 586  
 OL\_SCROLLDOWN Activation Type, 586  
 OL\_SCROLLTOP Activation Type, 587  
 OL\_SCROLLUP Activation Type, 587  
 OL\_SELECT Activation Type, 587  
 OL\_SELECTKEY Activation Type, 587  
 OIListCallbackStruct, 581  
 OIListItemAttrs, 582  
 Resources, 565  
 ScrollingList Modes, 562  
 Selectable ScrollingList, 560  
 Subwidgets, 559  
 Text Selections on Items, 561  
 XtNalign resource, 567  
 XtNapplAddItem resource, 567  
 XtNapplDeleteItem resource, 568  
 XtNapplEditClose resource, 569  
 XtNapplEditOpen resource, 569  
 XtNapplTouchItem resource, 571  
 XtNapplUpdateView resource, 571  
 XtNapplViewItem resource, 572  
 XtNcurrentItems resource, 573  
 XtNfirstViewableItem resource, 573  
 XtNitemCurrentCallback resource, 573  
 XtNitemHeight resource, 573  
 XtNitemNotCurrentCallback resource, 574  
 XtNlastViewableItem resource, 574  
 XtNlistPane resource, 574  
 XtNmultiClickCallback resource, 574  
 XtNnumCurrentItems resource, 575  
 XtNnumItems resource, 575  
 XtNposition resource, 575  
 XtNprefMaxWidth resource, 575  
 XtNprefMinWidth resource, 575  
 XtNrecomputeWidth resource, 576  
 XtNscrollingListItems resource, 576  
 XtNscrollingListMode resource, 576  
 XtNselectable resource, 577  
 XtNspace resource, 577  
 XtNtextField resource, 577  
 XtNuserDeleteItems resource, 578  
 XtNuserMakeCurrent resource, 579  
 XtNviewableItems resource, 580  
 XtNviewHeight resource, 580  
 ScrollingList Functions, 588  
 OIListAddItem(), 588  
 OIListDeleteAllItems(), 588  
 OIListDeleteItem(), 589  
 OIListDeleteItems(), 589  
 OIListEditItem(), 589  
 OIListEndEdit(), 589  
 OIListFirstViewableItem(), 590  
 OIListGetItemAttrs(), 590  
 OIListGetItemImage(), 590  
 OIListGetItemLabel(), 590  
 OIListGetItemSensitivity(), 591  
 OIListGetItemType(), 591  
 OIListGetItemUserData(), 591  
 OIListGetMode(), 591  
 OIListGetNextItem(), 592  
 OIListGetPrevItem(), 592  
 OIListIsItemCurrent(), 592



---

OIListIsValidItem(), 593  
 OIListLastViewableItem(), 593  
 OIListMakeAllItemsNotCurrent(), 593  
 OIListMakeItemCurrent(), 593  
 OIListMakeItemNotCurrent(), 594  
 OIListSetItemAttrs(), 594  
 OIListTouchItem(), 594  
 OIListUpdateView(), 595  
 SELECT mouse button, 25  
     drop site, 119  
 selection atom, 123  
 shadow color, 19  
 shell  
     menu button in header, 55  
     title, 57  
 Shell Resources, 41  
     Base Windows and Popup Windows, 41  
     FileChooserShell widget, 324  
     FontChooserShell widget, 387  
     MenuShell widget, 432  
     NoticeShell widget, 447  
     PopupWindowShell widget, 491  
     XtNallowShellResize, 42  
     XtNcreatePopupChildProc, 42  
     XtNgeometry, 43  
     XtNoverrideRedirect, 43  
     XtNpopdownCallback, 43  
     XtNpopupCallback, 43  
     XtNsaveUnder, 44  
     XtNvisual, 44  
 shift key resource, 25  
 single-byte character text, 39  
 single-byte encoding, 84  
 size resources for flat widgets, 65  
 Slider, 596  
     Activation Types, 607  
     Ancestry, 596  
     Application Notification, 598  
     Clicking SELECT, 597  
     Coloration, 598  
     Components, 596  
     Core Resources, 600  
     Drag Box Motion, 597  
     Dragging SELECT, 597  
     Keyboard Accelerator Display, 600  
     Keyboard Mnemonic Display, 599  
     Keyboard Traversal, 599  
     Moving Drag Box to Limits, 598  
     OL\_SCROLLBOTTOM Activation Type, 608  
     OL\_SCROLLDOWN Activation Type, 608  
     OL\_SCROLLLEFT Activation Type, 608  
     OL\_SCROLLLEFTEDGE Activation Type, 608  
     OL\_SCROLLRIGHT Activation Type, 609  
     OL\_SCROLLRIGHTEDGE Activation Type, 609  
     OL\_SCROLLTOP Activation Type, 609  
     OL\_SCROLLUP Activation Type, 609  
     OL\_SELECT Activation Type, 609  
     Primitive Resources, 600  
     Resources, 601  
     XtNdragCBType resource, 602  
     XtNendBoxes resource, 602  
     XtNgranularity resource, 602  
     XtNinitialDelay resource, 602  
     XtNmaxLabel resource, 603  
     XtNminLabel resource, 603  
     XtNorientation resource, 604  
     XtNpointerWarping resource, 604  
     XtNrecomputeSize resource, 604  
     XtNrepeatRate resource, 604  
     XtNsliderMax resource, 605  
     XtNsliderMin resource, 605  
     XtNsliderMoved resource, 605  
     XtNsliderValue resource, 606  
     XtNspan resource, 606  
     XtNstopPosition resource, 606  
     XtNticks resource, 607  
     XtNtickUnit resource, 607  
     XtNuseSetValCallback resource, 607  
 Spacing Between Controls, 265  
 spacing resource for flat widgets, 64



---

standards for internationalization, 99

StartCurrentTextBufferWord(), 185

StaticText, 610

- Activation Types, 617
- Ancestry, 610
- Coloration, 611
- Core Resources, 612
- Keyboard Mnemonic and Accelerator Display, 612
- Keyboard Traversal, 611
- OL\_ADJUST Activation Type, 618
- OL\_COPY Activation Type, 618
- OL\_SELECT Activation Type, 618
- Primitive Resources, 612
- Resources, 613
- Selecting and Operating on Text, 611
- Space Stripping, 611
- Text Clipping, 610
- Word Wrap, 610
- XtNalignment resource, 613
- XtNgravity resource, 614
- XtNheight resource, 614
- XtNhSpace resource, 615
- XtNlineSpace resource, 615
- XtNrecomputeSize resource, 615
- XtNselectable resource, 615
- XtNstring resource, 616
- XtNstrip resource, 616
- XtNvSpace resource, 615
- XtNwidth resource, 616
- XtNwrap resource, 616

StaticText Subwidget Resources

- NoticeShell widget, 449

status feedback font set, 54

status region, 91

strclose(), 108

strexp(), 171

strxp(), 171

strgetc(), 108

stropen(), 108

strrexp(), 172

Stub, 619

- Activation Types, 631
- Ancestry, 619
- Coloration, 620, 621
- Core Resources, 621
- Graphics Applications, 620
- Inheriting Procedures from Existing Widgets, 620
- Keyboard Accelerator Display, 621
- Keyboard Mnemonic Display, 621
- Keyboard Traversal, 621
- Local Widgets, 619
- Primitive Resources, 622
- Resources, 622
- Wrapping Widgets Around Existing Windows, 620
- XtNacceptFocusFunc resource, 623
- XtNactivateFunc resource, 623
- XtNdestroy resource, 624
- XtNexpose resource, 624
- XtNgetValuesHook resource, 624
- XtNheight resource, 625
- XtNhighlightHandlerProc resource, 625
- XtNinitialize resource, 626
- XtNinitializeHook resource, 626
- XtNqueryGeometry resource, 627
- XtNrealize resource, 627
- XtNreferenceStub resource, 627
- XtNregisterFocusFunc resource, 628
- XtNresize resource, 628
- XtNsetValues resource, 628
- XtNsetValuesAlmost resource, 629
- XtNsetValuesHook resource, 629
- XtNtraversalHandlerFunc resource, 629
- XtNwidth resource, 630
- XtNwindow resource, 630

submenus

- previewing, 25

Subwidget Resources

- AbbrevMenuButton widget, 232
- ControlArea widget, 262
- MenuShell widget, 418
- Scrollbar widget, 527

SuperCaret, 23

- resource, 21

---

## T

### text

- copying, 215
  - cutting, 215
  - data format, 39
  - font color resource, 37
  - multibyte character, 39
  - pasting, 215
  - single-byte, 39
- Text Buffer Functions, 173
- AllocateTextBuffer(), 173
  - BackwardScanTextBuffer(), 174
  - CopyTextBufferBlock(), 174
  - EndCurrentTextBufferWord(), 175
  - ForwardScanTextBuffer(), 175
  - FreeTextBuffer(), 175
  - GetTextBufferBlock(), 176
  - GetTextBufferBuffer(), 176
  - GetTextBufferChar(), 176
  - GetTextBufferLine(), 177
  - GetTextBufferLocation(), 177
  - IncrementTextBufferLocation(), 177
  - LastTextBufferLocation(), 178
  - LastTextBufferPosition(), 178
  - LineOfPosition(), 178
  - LocationOfPosition(), 178
  - NextLocation(), 179
  - NextTextBufferWord(), 179
  - OAllocateTextBuffer(), 186
  - OBackwardScanTextBuffer(), 187
  - OCopyTextBufferBlock(), 188
  - OEndCurrentTextBufferWord(), 188
  - OForwardScanTextBuffer(), 189
  - OFreeTextBuffer(), 190
  - OGetTextBufferBlock(), 190
  - OGetTextBufferBuffer(), 191
  - OGetTextBufferCharAtLoc(), 192
  - OGetTextBufferFileName(), 192
  - OGetTextBufferLine(), 193
  - OGetTextUndoDeleteItem(), 193
  - OGetTextUndoInsertItem(), 194
  - OIncrementTextBufferLocation(), 194
  - OIsTextBufferEmpty(), 195
  - OIsTextBufferModified(), 195
  - OLastCharInTextBufferLine(), 195
  - OLastTextBufferLine(), 196
  - OLastTextBufferLocation(), 196
  - OLastTextBufferPosition(), 197
  - OLineOfPosition(), 197
  - OLinesInTextBuffer(), 197
  - OLocationOfPosition(), 198
  - OLNextLocation(), 199
  - OLNextTextBufferWord(), 199
  - OLNumBytesInTextBufferLine(), 200
  - OLNumCharsInTextBufferLine(), 200
  - OLNumUnitsInTextBufferLine(), 201
  - OPositionOfLine(), 201
  - OPositionOfLocation(), 201
  - OPreviousLocation(), 202
  - OPreviousTextBufferWord(), 202
  - OReadFileIntoTextBuffer(), 203
  - OReadStringIntoTextBuffer(), 204
  - ORegisterAllTextBufferScanFunctions(), 204
  - ORegisterAllTextBufferWordDefinition(), 205
  - ORegisterPerTextBufferScanFunctions(), 205
  - ORegisterPerTextBufferWordDefinition(), 206
  - ORegisterTextBufferUpdate(), 207
  - OReplaceBlockInTextBuffer(), 207
  - OReplaceCharInTextBuffer(), 209
  - OSaveTextBuffer(), 210
  - OSetTextUndoDeleteItem(), 210
  - OSetTextUndoInsertItem(), 211
  - OStartCurrentTextBufferWord(), 211
  - OTextEditOITextBuffer(), 212
  - OUnitOffsetOfLocation(), 212
  - OUnregisterTextBufferUpdate(), 213
  - PositionOfLine(), 179
  - PositionOfLocation(), 179
  - PreviousLocation(), 180
  - PreviousTextBufferWord(), 180
  - ReadFileIntoTextBuffer(), 180
  - ReadStringIntoTextBuffer(), 181
  - RegisterTextBufferScanFunctions(), 181
  - RegisterTextBufferUpdate(), 182

---

RegisterTextBufferWordDefinition(), 181  
 ReplaceBlockInTextBuffer(), 182  
 ReplaceCharInTextBuffer(), 184  
 SaveTextBuffer(), 184  
 StartCurrentTextBufferWord(), 185  
 TextBuffer Macros(), 185  
 TextLocation Structure(), 173  
 UnregisterTextBufferUpdate(), 185  
 Text Buffer Functions for Internationalization, 186  
 TEXT database, 220  
 text format  
     default, 86  
 text formats, 84  
 Text Selection Operations, 214  
     Adjusted Selection, 215  
     Deletion of the New Selection, 214  
     Multiclick Selection, 215  
     Setting Insert Point, 214  
     Wipethrough Selection, 214  
 TextBuffer Macros(), 185  
 TextEdit, 633  
     Activation Types, 652  
     Ancestry, 633  
     Coloration, 637  
     Core Resources, 639  
     Editing Capabilities, 635  
     Keyboard Accelerator Display, 638  
     Keyboard Mnemonic Display, 638  
     Keyboard Traversal, 637  
     OL\_ADJUST Activation Type, 654  
     OL\_CHARBAK Activation Type, 654  
     OL\_CHARFWD Activation Type, 654  
     OL\_COPY Activation Type, 655  
     OL\_CUT Activation Type, 655  
     OL\_DELCHARBAK Activation Type, 655  
     OL\_DELCHARFWD Activation Type, 656  
     OL\_DELLINE Activation Type, 656  
     OL\_DELLINEBAK Activation Type, 657  
     OL\_DELLINEFWD Activation Type, 657  
     OL\_DELWORDBAK Activation Type, 658  
     OL\_DELWORDFWD Activation Type, 658  
     OL\_DOCEND Activation Type, 659  
     OL\_DOCSTART Activation Type, 659  
     OL\_LINEEND Activation Type, 659  
     OL\_LINESTART Activation Type, 660  
     OL\_MENU Activation Type, 660  
     OL\_MENUKEY Activation Type, 660  
     OL\_PAGEDOWN Activation Type, 660  
     OL\_PAGELEFT Activation Type, 660  
     OL\_PAGERIGHT Activation Type, 661  
     OL\_PAGEUP Activation Type, 661  
     OL\_PANEEND Activation Type, 661  
     OL\_PANESTART Activation Type, 661  
     OL\_PASTE Activation Type, 662  
     OL\_ROWDOWN Activation Type, 662  
     OL\_ROWUP Activation Type, 662  
     OL\_SCROLLDOWN Activation Type, 663  
     OL\_SCROLLLEFT Activation Type, 663  
     OL\_SCROLLLEFTEDGE Activation Type, 663  
     OL\_SCROLLRIGHT Activation Type, 663  
     OL\_SCROLLRIGHTEDGE Activation Type, 663  
     OL\_SCROLLUP Activation Type, 664  
     OL\_SELCHARBAK Activation Type, 664  
     OL\_SELCHARFWD Activation Type, 664  
     OL\_SELECT Activation Type, 665  
     OL\_SELFLIPENDS Activation Type, 665  
     OL\_SELLINE Activation Type, 665  
     OL\_SELLINEBAK Activation Type, 666  
     OL\_SELLINEFWD Activation Type, 666

---

OL\_SELWORDBAK Activation Type, 666  
 OL\_SELWORDFWD Activation Type, 667  
 OL\_UNDO Activation Type, 667  
 OL\_WORDBAK Activation Type, 668  
 OL\_WORDFWD Activation Type, 668  
 Primitive Resources, 639  
 Resources, 640  
 Sizing the Display, 636  
 Text Hierarchy, 635  
 The Text Buffer, 635  
 Wrapping, 636  
 XtNblinkRate resource, 641  
 XtNbottomMargin resource, 641  
 XtNbuttons resource, 641  
 XtNcharsVisible resource, 642  
 XtNcopyLabel resource, 643  
 XtNcopyMnemonic resource, 643  
 XtNcursorPosition resource, 644  
 XtNcutLabel resource, 643  
 XtNcutMnemonic resource, 643  
 XtNdeleteLabel resource, 643  
 XtNdeleteMnemonic resource, 643  
 XtNdisplayPosition resource, 644  
 XtNeditType resource, 644  
 XtNgrowMode resource, 644  
 XtNimPreeditStyle resource, 645  
 XtNinsertTab resource, 645  
 XtNkeys resource, 646  
 XtNleftMargin resource, 646  
 XtNlinesVisible resource, 646  
 XtNmargin resource, 646  
 XtNmenuTitle resource, 647  
 XtNmodifyVerification resource, 647  
 XtNmotionVerification resource, 648  
 XtNpasteLabel resource, 643  
 XtNpasteMnemonic resource, 643  
 XtNpostModifyNotification resource, 649  
 XtNrightMargin resource, 649  
 XtNselectEnd resource, 650  
 XtNselectStart resource, 650  
 XtNsource resource, 650  
 XtNsourceType resource, 651  
 XtNtabTable resource, 651  
 XtNtopMargin resource, 651  
 XtNundoLabel resource, 643  
 XtNundoMnemonic resource, 643  
 XtNwrapMode resource, 651  
 TextEdit Functions, 669  
 OITextEditClearBuffer(), 669  
 OITextEditCopyBuffer(), 669  
 OITextEditCopySelection(), 670  
 OITextEditGetCursorPosition(), 670  
 OITextEditGetLastPosition(), 671  
 OITextEditInsert(), 673  
 OITextEditMoveDisplayPosition(), 671  
 OITextEditOITextBuffer(), 672  
 OITextEditPaste(), 673  
 OITextEditReadSubString(), 669  
 OITextEditRedraw(), 670  
 OITextEditSetCursorPosition(), 671  
 OITextEditTextBuffer(), 672  
 OITextEditUpdate(), 673  
 TextField, 674  
 Activation Types, 684  
 Ancestry, 674  
 Caret Position, 678  
 Coloration, 675  
 Components, 674  
 Composite Resources, 679  
 Core Resources, 678  
 Editing, 676  
 Input Validation, 677  
 Keyboard Accelerator Display, 676  
 Keyboard Input, 675  
 Keyboard Mnemonic Display, 676  
 Keyboard Traversal, 675  
 Manager Resources, 679  
 OL\_ADJUST Activation Type, 685  
 OL\_CHARBAK Activation Type, 685  
 OL\_CHARFWD Activation Type, 686  
 OL\_COPY Activation Type, 686  
 OL\_CUT Activation Type, 686  
 OL\_DELCHARBAK Activation Type, 686  
 OL\_DELCHARFWD Activation Type, 687

---

OL\_DELLINE Activation Type, 687  
 OL\_DELLINEBAK Activation Type, 687  
 OL\_DELLINEFWD Activation Type, 688  
 OL\_DELWORDBAK Activation Type, 688  
 OL\_DELWORDFWD Activation Type, 689  
 OL\_LINEEND Activation Type, 689  
 OL\_LINESTART Activation Type, 689  
 OL\_MENU Activation Type, 689  
 OL\_MENUKEY Activation Type, 689  
 OL\_PASTE Activation Type, 690  
 OL\_RETURN Activation Type, 690  
 OL\_SCROLLLEFT Activation Type, 690  
 OL\_SCROLLLEFTEDGE Activation Type, 690  
 OL\_SCROLLRIGHT Activation Type, 690  
 OL\_SCROLLRIGHTEDGE Activation Type, 690  
 OL\_SELCHARBAK Activation Type, 690  
 OL\_SELCHARFWD Activation Type, 691  
 OL\_SELECT Activation Type, 691  
 OL\_SELFLIPENDS Activation Type, 691  
 OL\_SELLINE Activation Type, 691  
 OL\_SELLINEBAK Activation Type, 692  
 OL\_SELLINEFWD Activation Type, 692  
 OL\_SELWORDBAK Activation Type, 692  
 OL\_SELWORDFWD Activation Type, 693  
 OL\_UNDO Activation Type, 693  
 OL\_WORDBAK Activation Type, 693  
 OL\_WORDFWD Activation Type, 693  
 Resources, 679  
 Scrolling, 676  
 Scrolling Long Text Input, 677  
 Selecting and Operating on the Input Field, 678  
 Subwidget, 675  
 Text Selection, 676  
 XtNcharsVisible resource, 680  
 XtNeditType resource, 680  
 XtNfont resource, 680  
 XtNfontColor resource, 680  
 XtNimPreeditStyle resource, 680  
 XtNinitialDelay resource, 681  
 XtNinsertTab resource, 681  
 XtNmaximumSize resource, 682  
 XtNrepeatRate resource, 682  
 XtNscale resource, 682  
 XtNstring resource, 682  
 XtNtextEditWidget resource, 683  
 XtNtextFormat resource, 683  
 XtNverification resource, 683  
 TextField Functions, 695  
   OITextFieldCopyString(), 695  
   OITextFieldGetOlString(), 695  
   OITextFieldGetString(), 696  
   OITextFieldOlCopyString(), 695  
 TextLine, 697  
   Activation Types, 711  
   Ancestry, 697  
   Caret Position, 699  
   Coloration, 699  
   Components, 697, 698  
   Core Resources, 700  
   Display of Text, 699  
   Keyboard Accelerator Display, 699  
   Keyboard Mnemonic Display, 698  
   Keyboard Traversal, 698  
   OL\_CHARBAK Activation Type, 712  
   OL\_CHARFWD Activation Type, 712  
   OL\_COPY Activation Type, 712  
   OL\_CUT Activation Type, 712  
   OL\_DELCHARBAK Activation Type, 712  
   OL\_DELCHARFWD Activation Type, 713  
   OL\_DELLINE Activation Type, 713  
   OL\_DELLINEBAK Activation Type, 713

---

OL\_DELLINEFWD Activation Type, 714  
 OL\_DELWORDBAK Activation Type, 714  
 OL\_DELWORDFWD Activation Type, 714  
 OL\_LINEEND Activation Type, 715  
 OL\_LINESTART Activation Type, 715  
 OL\_NEXTFIELD Activation Type, 715  
 OL\_PASTE Activation Type, 715  
 OL\_PREVFIELD Activation Type, 716  
 OL\_UNDO Activation Type, 716  
 OL\_WORDBAK Activation Type, 716  
 OL\_WORDFWD Activation Type, 716  
 Primitive Resources, 701  
 Resources, 701  
 Selection of Text, 699  
 XtNblinkRate resource, 702  
 XtNcaptionAlignment resource, 702  
 XtNcaptionFont resource, 702  
 XtNcaptionLabel resource, 703  
 XtNcaptionPosition resource, 703  
 XtNcaptionSpace resource, 703  
 XtNcaptionWidth resource, 703  
 XtNcharsVisible resource, 704  
 XtNcommitCallback resource, 704  
 XtNcursorPosition resource, 705  
 XtNeditType resource, 705  
 XtNimPreeditStyle resource, 705  
 XtNinitialDelay resource, 706  
 XtNinsertTab resource, 706  
 XtNmaximumChars resource, 706  
 XtNmenu resource, 707  
 XtNmotionCallback resource, 707  
 XtNpostModifyCallback resource, 709  
 XtNpreModifyCallback resource, 708  
 XtNrepeatRate resource, 710  
 XtNstring resource, 710  
 XtNunderline resource, 710  
 XtNupdateDisplay resource, 710  
 TextLine Functions, 717  
     OITLGetPosition(), 717  
     OITLGetSelection(), 717  
     OITLGetSubString(), 718  
     OITLOperateOnSelection(), 718  
     OITLSetSelection(), 719  
     OITLSetSubString(), 719  
 TextLine Resources  
     NumericField widget, 458  
 TextLocation Structure(), 173  
 TextUpdateFunction, 182, 208  
 three-dimensional resource, 26  
 tiling border resource, 29  
 title  
     shell widget, 57  
 title resource, 49  
 toolkit grabs (avoiding), 430  
 Toolkit Resource Functions, 216  
     OIGetApplicationValues(), 216  
     OISetApplicationValues(), 216  
 toolkit resources, *See* OLIT Toolkit Resources, 17  
 TopLevelShell Resources, 60  
     XtNiconic, 60  
     XtNiconName, 61  
     XtNiconNameEncoding, 61  
 transient window, 50  
 TransientShell Resources, 60  
     FileChooserShell widget, 327  
     FontChooserShell widget, 389  
     MenuShell widget, 434  
     NoticeShell widget, 449  
     PopupWindowShell widget, 493  
     XtNtransientFor, 60  
 translation manager  
     key event syntax, 74  
 translations resource, 32  
 traversal list, 163  
     inserting by widget ID, 39  
     inserting by widget name, 38  
 traversal order  
     default, 163  
 traversal, *See* keyboard traversal, 40  
 two-dimensional resource, 26

## U

underlined keyboard mnemonics, 26

Undo

    implementing, 183

UnregisterTextBufferUpdate(), 185

updating the display, 118

## V

validation of numeric fields, 464

VendorShell Resources, 52

    FileChooserShell widget, 326

    FontChooserShell widget, 388

    MenuShell widget, 434

    NoticeShell widget, 448

    PopupWindowShell widget, 492

    XtNbusy, 53

    XtNconsumeEvent, 53

    XtNdefaultImName, 53

    XtNfocusWidget, 54

    XtNfooterPresent, 54

    XtNimFontSet, 54

    XtNimStatusStyle, 54

    XtNleftFooterString, 55

    XtNleftFooterVisible, 55

    XtNmenuButton, 55

    XtNmenuType, 56

    XtNpushpin, 56

    XtNresizeCorners, 57

    XtNrightFooterString, 57

    XtNrightFooterVisible, 57

    XtNshellTitle, 57

    XtNuserData, 58

    XtNwindowHeader, 58

    XtNwmProtocol, 58

    XtNwmProtocolInterested, 59

Vertical and Horizontal Sliders, 596

Virtual Event Functions, 217

    LookupOIInputEvent(), 217

    OIClassSearchIEDB(), 219

    OIClassSearchTextDB(), 220

    OICreateInputEventDB(), 221

    OIDetermineMouseAction(), 217

    OILookupInputEvent(), 222

    OIReplayBtnEvent(), 219

    OIWidgetSearchIEDB(), 225

    OIWidgetSearchTextDB(), 226

virtual events

    relationship to Activation Types, 71

    use with callbacks, 72

visual, 44

visual class, 164

visuals

    multiple, 164

volume (beep) resource, 19

volume resource, 18

## W

wide character encoding, 84

wide character text, 39

widgets

    activating, 103

    associating by Activation Types, 73

    creating tree in one call, 166

    emanate, 443

    follower, 103

    leader, 103

    mapping, 31

    programmatically activation, 72

    updating, 118

window

    base, 41

    header, 58

    menu on shell, 56

    pushpin, 56

    resize corners, 57

    transient, 50

    visual, 44

window height resource, 30

window manager

    managing shell window, 43

    messages, 58

window menu resource, 56

window size

    constraining, 48

    increment, 50

    ranges, 49



---

window width resource, 32  
 WM\_COMMAND, 62  
 WM\_PROTOCOL messages, 58  
 WMSHELL Resources, 45
 

- FileChooserShell widget, 325
- FontChooserShell widget, 387
- MenuShell widget, 433
- NoticeShell widget, 447
- PopupWindowShell widget, 491
- XtNbaseHeight, 46
- XtNbaseWidth, 46
- XtNheightInc, 46
- XtNiconMask, 46
- XtNiconPixmap, 46
- XtNiconWindow, 47
- XtNiconX, 47
- XtNiconY, 47
- XtNinitialState, 47
- XtNinput, 48
- XtNmaxAspectX, 48
- XtNmaxAspectY, 48
- XtNmaxHeight, 49
- XtNmaxWidth, 49
- XtNminAspectX, 48
- XtNminAspectY, 48
- XtNminHeight, 49
- XtNminWidth, 49
- XtNtitle, 49
- XtNtitleEncoding, 50
- XtNtransient, 50
- XtNwaitForWm, 50
- XtNwidthInc, 50
- XtNwindowGroup, 51
- XtNwinGravity, 51
- XtNwmTimeout, 52

**X**

X Logical Font Description (XLFD), 368
 

- x-, y-coordinates
  - of base window, 47
  - XtNx/XtNy resources, 32
- XCopyArea, 273
- XCopyPlane, 273
- XEvent
  - consumption resource, 36
  - drag and drop types, 124
  - mapping to Activation Types, 73
  - translating to virtual event, 222
- Xevent
  - mapping, 32
- XEvent consumption resource, 53
- XFILESEARCHPATH, 89
- XFontSet, 87
- XGetSelection(), 123
- XGrabKeyboard(), 135
- XGrabPointer(), 135
- XLFD, 24, 87
  - fonts, 367
  - X Logical Font Description, 368
- XLookupString(), 217, 223
- XMatchVisualInfo(), 164
- XmbDrawString(), 40
- XnlLanguage, 83
- XParseGeometry(), 43
- XRaiseWindow(), 488
- XSetInputFocus(), 161, 162, 623
- XtAcceptFocusFunc, 623
- XtAddCallback(), 36, 59
- XtAddEventHandler(), 475
- XtAppCreateShell(), 102
- XtAppGetErrorDatabaseText(), 152
- XtAppInitialize(), 62, 102
- XtCallAcceptFocus(), 160
- XtCallbackProc procedure, 335
- XtConfigureWidget(), 631
- XtCreateApplicationContext(), 102
- XtCreateApplicationShell(), 41
- XtCreatePopupShell(), 41
- XtDisownSelection(), 134
- XtDisplayToApplicationContext(), 153
- XtFree(), 142
- XtGetSelectionValue(), 123
- XtInitializeDisplay(), 102
- XtIsSensitive(), 28
- XtMoveWidget(), 631



---

XtNaccelerator  
     CheckBox widget, 253  
     Primitive Resources, 35  
 XtNaccelerators  
     Core Resources, 27  
 XtNacceleratorText  
     CheckBox widget, 253  
     Primitive Resources, 35  
 XtNacceptFocusFunc  
     Stub widget, 623  
 XtNactivateFunc  
     Stub widget, 623  
 XtNalign  
     ScrollingList widget, 567  
 XtNalignCaptions  
     ControlArea widget, 262  
 XtNalignHorizontal  
     ScrolledWindow widget, 546  
 XtNalignment  
     Caption widget, 243  
     StaticText widget, 613  
 XtNalignVertical  
     ScrolledWindow widget, 546  
 XtNallowChangeBars  
     ControlArea widget, 262  
 XtNallowShellResize  
     Shell Resources, 42  
 XtNancestorSensitive  
     Core Resources, 28  
 XtNapplAddItem  
     ScrollingList widget, 567  
 XtNapplDeleteItem  
     ScrollingList widget, 568  
 XtNapplEditClose  
     ScrollingList widget, 569  
 XtNapplEditOpen  
     ScrollingList widget, 569  
 XtNapplicationFolders  
     FileChooser widget, 309  
 XtNapplicationFoldersMaxCount  
     FileChooser widget, 309  
 XtNapplTouchEvent  
     ScrollingList widget, 571  
 XtNapplUpdateView  
     ScrollingList widget, 571  
 XtNapplViewItem  
     ScrollingList widget, 572  
 XtNapply  
     PopupWindowShell widget, 494  
 XtNapplyCallback  
     FileChooser widget, 373  
 XtNapplyLabel  
     FileChooser widget, 373  
     PopupWindowShell widget, 495  
 XtNapplyMnemonic  
     PopupWindowShell widget, 495  
 XtNargc  
     ApplicationShell Resources, 61  
 XtNargv  
     ApplicationShell Resources, 61  
 XtNattributeListHeight  
     FileChooser widget, 374  
 XtNbackground  
     Core Resources, 28  
     overriding, 28  
 XtNbackgroundPixmap  
     Core Resources, 28  
 XtNbaseHeight  
     WMSHELL Resources, 46  
 XtNbaseWidth  
     WMSHELL Resources, 46  
 XtNbeep  
     OLIT Toolkit Resources, 18  
 XtNbeepVolume  
     OLIT Toolkit Resources, 19  
 XtNblinkRate  
     TextEdit widget, 641  
     TextLine widget, 702  
 XtNborderColor  
     Core Resources, 29  
     overriding, 29  
 XtNborderPixmap  
     Core Resources, 29  
 XtNborderWidth  
     Core Resources, 29  
 XtNbottomMargin

---

TextEdit widget, 641  
 XtNbusy  
     OblongButton widget, 480  
     VendorShell Resources, 53  
 XtNbusyPixmap  
     DropTarget widget, 281  
 XtNbuttons  
     TextEdit widget, 641  
 XtNcancelAccelerator  
     FileChooser widget, 313  
 XtNcancelButtonWidget  
     FileChooser widget, 315  
 XtNcancelCallback  
     FileChooser widget, 316  
     FontChooser widget, 374  
 XtNcancelLabel  
     FileChooser widget, 319  
     FontChooser widget, 375  
 XtNcancelMnemonic  
     FileChooser widget, 313  
 XtNcaptionAlignment  
     TextLine widget, 702  
 XtNcaptionFont  
     TextLine widget, 702  
 XtNcaptionLabel  
     TextLine widget, 703  
 XtNcaptionPosition  
     TextLine widget, 703  
 XtNcaptionSpace  
     TextLine widget, 703  
 XtNcaptionWidth  
     TextLine widget, 703  
 XtNcenter  
     ControlArea widget, 263  
 XtNchangeBar  
     ControlArea widget, 263  
 XtNchangedCallback  
     FileChooser widget, 375  
 XtNcharsetInfo  
     FontChooser widget, 376  
 XtNcharsVisible  
     TextEdit widget, 642  
     TextField widget, 680  
     TextLine widget, 704  
 XtNchildren  
     Composite Resources, 33  
 XtNclientData  
     FlatExclusives widget, 352  
 XtNcolormap  
     Core Resources, 29  
 XtNcolorTupleList  
     OLIT Toolkit Resources, 19  
 XtNcommandButtonWidget  
     FileChooser widget, 315  
 XtNcommitCallback  
     TextLine widget, 704  
 XtNcomparisonFunc  
     FileChooser widget, 311  
 XtNcomputeGeometries  
     ScrolledWindow widget, 546  
 XtNconsumeEvent  
     Primitive Resources, 36  
     use with Activation Types, 72  
     VendorShell Resources, 53  
 XtNcontrolArea  
     NoticeShell widget, 450  
 XtNcontrolName  
     OLIT Toolkit Resources, 20  
 XtNconvertProc  
     NumericField widget, 459  
 XtNcopyLabel  
     TextEdit widget, 643  
 XtNcopyMnemonic  
     TextEdit widget, 643  
 XtNcreatePopupChildProc  
     Shell Resources, 42  
 XtNcurrentFolder  
     FileChooser widget, 300  
 XtNcurrentFolderLabelString  
     FileChooser widget, 319  
 XtNcurrentFolderLabelWidget  
     FileChooser widget, 315  
 XtNcurrentFolderWidget  
     FileChooser widget, 315  
 XtNcurrentItems  
     ScrollingList widget, 573

---

XtNcurrentPage  
     Scrollbar widget, 527  
     ScrolledWindow widget, 548  
 XtNcursorPosition  
     TextEdit widget, 644  
     TextLine widget, 705  
 XtNcutLabel  
     TextEdit widget, 643  
 XtNcutMnemonic  
     TextEdit widget, 643  
 XtNdefault  
     FlatExclusives widget, 352  
     MenuButton widget, 419  
     OblongButton widget, 481  
     RectButton widget, 504  
 XtNdefaultDocumentName  
     FileChooser widget, 319  
 XtNdefaultDocumentSuffix  
     FileChooser widget, 320  
 XtNdefaultImName, 97  
     VendorShell Resources, 53  
 XtNdeleteLabel  
     TextEdit widget, 643  
 XtNdeleteMnemonic  
     TextEdit widget, 643  
 XtNdelta  
     NumericField widget, 459  
 XtNdeltaCallback  
     NumericField widget, 460  
 XtNdeltaState  
     NumericField widget, 460  
 XtNdepth  
     Core Resources, 30  
 XtNdestroy  
     Stub widget, 624  
 XtNdestroyCallback  
     Core Resources, 30  
 XtNdim  
     CheckBox widget, 254  
     FlatExclusives widget, 353  
     RectButton widget, 505  
 XtNdisplayPosition  
     TextEdit widget, 644  
 XtNdndAcceptCursor  
     DropTarget widget, 281  
 XtNdndAnimateCallback  
     DropTarget widget, 282  
 XtNdndCopyCursor  
     DropTarget widget, 282  
 XtNdndMoveCursor  
     DropTarget widget, 282  
 XtNdndPreviewCallback  
     DropTarget widget, 283  
 XtNdndPreviewHints  
     DropTarget widget, 282  
 XtNdndRejectCursor  
     DropTarget widget, 283  
 XtNdndTriggerCallback  
     DropTarget widget, 284  
 XtNdocumentListWidget  
     FileChooser widget, 315  
 XtNdocumentNameLabelWidget  
     FileChooser widget, 315  
 XtNdocumentNameTypeInWidget  
     FileChooser widget, 315  
 XtNdragCBType  
     Scrollbar widget, 528  
     Slider widget, 602  
 XtNdragRightDistance  
     OLIT Toolkit Resources, 20  
 XtNeditType  
     TextEdit widget, 644  
     TextField widget, 680  
     TextLine widget, 705  
 XtNemanateWidget  
     NoticeShell widget, 450  
 XtNendBoxes  
     Slider widget, 602  
 XtNerrorCallback  
     FontChooser widget, 376  
 XtNexpandTilde  
     FileChooser widget, 312  
 XtNexpose  
     Stub widget, 624  
 XtNexposeCallback  
     DrawArea widget, 272

---

XtNextensionArea  
     FontChooser widget, 377  
 XtNextensionClass  
     FileChooser widget, 314  
 XtNextensionName  
     FileChooser widget, 314  
 XtNextensionWidget  
     FileChooser widget, 314  
 XtNfileChooserWidget  
     FileChooserShell widget, 327  
 XtNfilterProc  
     FileChooser widget, 307  
 XtNfilterString  
     FileChooser widget, 308  
 XtNfirstViewableItem  
     ScrollingList widget, 573  
 XtNfocusWidget  
     VendorShell Resources, 54  
 XtNfolderOpenedCallback  
     FileChooser widget, 316  
 XtNfolderPromptString  
     FileChooser widget, 318  
 XtNfollowSymlinks  
     FileChooser widget, 300  
 XtNfont  
     Caption widget, 243  
     CheckBox widget, 254  
     FileChooser widget, 300  
     Primitive Resources, 36  
     TextField widget, 680  
 XtNfontChooserWidget  
     FontChooserShell widget, 389  
 XtNfontColor  
     Caption widget, 243  
     CheckBox widget, 254  
     FileChooser widget, 300  
     Primitive Resources, 37  
     TextField widget, 680  
 XtNfontSearchSpec  
     FontChooser widget, 377  
 XtNfooterPanel  
     PopupWindowShell widget, 495  
 XtNfooterPresent  
     VendorShell Resources, 54  
 XtNforceHorizontalSB  
     ScrolledWindow widget, 548  
 XtNforceVerticalSB  
     ScrolledWindow widget, 549  
 XtNforeground  
     CheckBox widget, 254  
     DrawArea widget, 273  
     FileChooser widget, 301  
     Primitive Resources, 37  
     ScrolledWindow widget, 549  
 XtNfull  
     DropTarget widget, 284  
 XtNgeometry  
     Shell Resources, 43  
 XtNgetValuesHook  
     Stub widget, 624  
 XtNgotoButtonWidget  
     FileChooser widget, 315  
 XtNgotoHomeAccelerator  
     FileChooser widget, 313  
 XtNgotoHomeButtonWidget  
     FileChooser widget, 315  
 XtNgotoHomeLabel  
     FileChooser widget, 319  
 XtNgotoHomeMnemonic  
     FileChooser widget, 313  
 XtNgotoLabel  
     FileChooser widget, 319  
 XtNgotoMenuWidget  
     FileChooser widget, 315  
 XtNgotoPromptString  
     FileChooser widget, 318  
 XtNgotoPromptWidget  
     FileChooser widget, 315  
 XtNgotoTypeInWidget  
     FileChooser widget, 315  
 XtNgoUpOneFolderLabel  
     FileChooser widget, 319  
 XtNgrabPointer  
     OLIT Toolkit Resources, 20  
 XtNgranularity  
     Scrollbar widget, 528

---

Slider widget, 602  
 XtNgraphicsExposeCallback  
     DrawArea widget, 273  
 XtNgravity  
     Flat Resources, 63  
     StaticText widget, 614  
 XtNgravity Values, 614  
 XtNgrowMode  
     TextEdit widget, 644  
 XtNhAutoScroll  
     ScrolledWindow widget, 550  
 XtNheight  
     Core Resources, 30  
     StaticText widget, 614  
     Stub widget, 625  
 XtNheightInc  
     WMSHELL Resources, 46  
 XtNhelpModel  
     OLIT Toolkit Resources, 21  
 XtNhereToLeftLabel  
     Scrollbar widget, 528  
 XtNhereToLeftMnemonic  
     Scrollbar widget, 529  
 XtNhereToTopLabel  
     Scrollbar widget, 528  
 XtNhereToTopMnemonic  
     Scrollbar widget, 529  
 XtNhideDotFiles  
     FileChooser widget, 308  
 XtNhighlightHandlerProc  
     Stub widget, 625  
 XtNhInitialDelay  
     ScrolledWindow widget, 550  
 XtNhistoryFoldersMaxCount  
     FileChooser widget, 309  
 XtNhistoryFoldersMinCount  
     FileChooser widget, 309  
 XtNhMenuPane  
     ScrolledWindow widget, 550  
 XtNhomeFolder  
     FileChooser widget, 310  
 XtNhPad  
     ControlArea widget, 264  
     Flat Resources, 64  
 XtNhRepeatRate  
     ScrolledWindow widget, 551  
 XtNhScrollbar  
     ScrolledWindow widget, 551  
 XtNhSliderMoved  
     ScrolledWindow widget, 551  
 XtNhSpace  
     ControlArea widget, 264  
     Flat Resources, 64  
     FlatExclusives widget, 353  
     StaticText widget, 615  
 XtNhStepSize  
     ScrolledWindow widget, 552  
 XtNiconic  
     TopLevelShell Resources, 60  
 XtNiconMask  
     WMSHELL Resources, 46  
 XtNiconName  
     TopLevelShell Resources, 61  
 XtNiconNameEncoding  
     TopLevelShell Resources, 61  
 XtNiconPixmap  
     overriding, 47  
     WMSHELL Resources, 46  
 XtNiconWindow  
     WMSHELL Resources, 47  
 XtNiconX  
     WMSHELL Resources, 47  
 XtNiconY  
     WMSHELL Resources, 47  
 XtNimFontSet, 97  
     VendorShell Resources, 54  
 XtNimPreeditStyle, 93  
     TextEdit widget, 645  
     TextField widget, 680  
     TextLine widget, 705  
 XtNimStatusStyle, 95  
     VendorShell Resources, 54  
 XtNincludeAccelerator  
     FileChooser widget, 313  
 XtNincludeLabel  
     FileChooser widget, 319

---

XtNincludeMnemonic  
    FileChooser widget, 313

XtNinitialDelay  
    Scrollbar widget, 529  
    Slider widget, 602  
    TextField widget, 681  
    TextLine widget, 706

XtNinitialFontName  
    FontChooser widget, 378

XtNinitialize  
    Stub widget, 626

XtNinitializeHook  
    Stub widget, 626

XtNinitialState  
    WMShell Resources, 47

XtNinitialX  
    ScrolledWindow widget, 552

XtNinitialY  
    ScrolledWindow widget, 552

XtNinput  
    WMShell Resources, 48

XtNinputDocumentCallback  
    FileChooser widget, 303

XtNinputFocusColor  
    Primitive Resources, 37

XtNinputFocusFeedback  
    OLIT Toolkit Resources, 21

XtNinsertPosition  
    Composite Resources, 33

XtNinsertTab  
    TextEdit widget, 645  
    TextField widget, 681  
    TextLine widget, 706

XtNitemCurrentCallback  
    ScrollingList widget, 573

XtNitemFields  
    Flat Resources, 64

XtNitemGravity  
    Flat Resources, 65

XtNitemHeight  
    ScrollingList widget, 573

XtNitemMaxHeight  
    Flat Resources, 65

XtNitemMaxWidth  
    Flat Resources, 65

XtNitemMinHeight  
    Flat Resources, 65

XtNitemMinWidth  
    Flat Resources, 65

XtNitemNotCurrentCallback  
    ScrollingList widget, 574

XtNitems  
    Flat Resources, 66

XtNitemsTouched  
    Flat Resources, 66

XtNkeys  
    TextEdit widget, 646

XtNlabel  
    Caption widget, 244  
    CheckBox widget, 254  
    Flat Resources, 67  
    MenuButton widget, 419  
    OblongButton widget, 481  
    RectButton widget, 505

XtNlabelImage  
    CheckBox widget, 255  
    Flat Resources, 67  
    MenuButton widget, 419  
    OblongButton widget, 481  
    RectButton widget, 505

XtNlabelJustify  
    CheckBox widget, 255  
    Flat Resources, 67  
    MenuButton widget, 420  
    OblongButton widget, 482  
    RectButton widget, 506

XtNlabelTile  
    CheckBox widget, 255  
    Flat Resources, 67  
    OblongButton widget, 482  
    RectButton widget, 506

XtNlabelType  
    CheckBox widget, 256  
    MenuButton widget, 420  
    OblongButton widget, 482  
    RectButton widget, 506

XtNlastDocumentName

---

FileChooser widget, 301  
 XtNlastViewableItem  
     ScrollingList widget, 574  
 XtNlayout  
     BulletinBoard widget, 237  
 XtNlayoutHeight  
     Flat Resources, 68  
 XtNlayoutType  
     ControlArea widget, 265  
     Exclusives widget, 291  
     Flat Resources, 68  
     Nonexclusives widget, 441  
 XtNlayoutWidth  
     Flat Resources, 68  
 XtNleftFooterString  
     VendorShell Resources, 55  
 XtNleftFooterVisible  
     overriding, 55  
     VendorShell Resources, 55  
 XtNleftMargin  
     Gauge widget, 408  
     TextEdit widget, 646  
 XtNleftToHereLabel  
     Scrollbar widget, 528  
 XtNleftToHereMnemonic  
     Scrollbar widget, 529  
 XtNlineSpace  
     StaticText widget, 615  
 XtNlinesVisible  
     TextEdit widget, 646  
 XtNlistChoiceCallback  
     FileChooser widget, 317  
 XtNlistPane  
     ScrollingList widget, 574  
 XtNlistPromptWidget  
     FileChooser widget, 315  
 XtNlistVisibleItemCount  
     FileChooser widget, 301  
 XtNlistVisibleItemMinCount  
     FileChooser widget, 301  
 XtNlockName  
     OLIT Toolkit Resources, 21  
 XtNlowerControlArea  
     PopupWindowShell widget, 496  
 XtNmappedWhenManaged  
     Core Resources, 31  
 XtNmargin  
     TextEdit widget, 646  
 XtNmaxAspectX  
     WMSHELL Resources, 48  
 XtNmaxAspectY  
     WMSHELL Resources, 48  
 XtNmaxHeight  
     WMSHELL Resources, 49  
 XtNmaximumChars  
     TextLine widget, 706  
 XtNmaximumPointSize  
     FontChooser widget, 378  
 XtNmaximumSize  
     TextField widget, 682  
 XtNmaxLabel  
     Gauge widget, 408  
     Slider widget, 603  
 XtNmaxValue  
     NumericField widget, 461  
 XtNmaxWidth  
     WMSHELL Resources, 49  
 XtNmeasure  
     ControlArea widget, 266  
     Exclusives widget, 291  
     Flat Resources, 69  
     Nonexclusives widget, 441  
 XtNmenu  
     TextLine widget, 707  
 XtNmenuAugment  
     MenuShell widget, 435  
 XtNmenuButton  
     VendorShell Resources, 55  
 XtNmenuMark  
     MenuButton widget, 420  
 XtNmenuMarkRegion  
     OLIT Toolkit Resources, 22  
 XtNmenuPane  
     AbbrevMenuButton widget, 232  
     MenuButton widget, 421  
     MenuShell widget, 436

---

Scrollbar widget, 529  
 XtNmenuTitle  
     PopupWindowShell widget, 496  
     Scrollbar widget, 530  
     TextEdit widget, 647  
 XtNmenuType  
     VendorShell Resources, 56  
 XtNmetaKey  
     OLIT Toolkit Resources, 22  
 XtNminAspectX  
     WMSHELL Resources, 48  
 XtNminAspectY  
     WMSHELL Resources, 48  
 XtNminHeight  
     WMSHELL Resources, 49  
 XtNminLabel  
     Gauge widget, 409  
     Slider widget, 603  
 XtNminValue  
     NumericField widget, 461  
 XtNminWidth  
     WMSHELL Resources, 49  
 XtNmnemonic  
     Caption widget, 244  
     CheckBox widget, 256  
     Primitive Resources, 38  
 XtNmnemonicPrefix  
     OLIT Toolkit Resources, 22  
 XtNmod1Name  
     OLIT Toolkit Resources, 23  
 XtNmod2Name  
     OLIT Toolkit Resources, 23  
 XtNmod3Name  
     OLIT Toolkit Resources, 23  
 XtNmod4Name  
     OLIT Toolkit Resources, 23  
 XtNmod5Name  
     OLIT Toolkit Resources, 23  
 XtNmodifyVerification  
     TextEdit widget, 647  
 XtNmotionCallback  
     TextLine widget, 707  
 XtNmotionVerification  
     TextEdit widget, 648  
 XtNmouseDampingFactor  
     OLIT Toolkit Resources, 23  
 XtNmouseless  
     OLIT Toolkit Resources, 23  
 XtNmultiClickCallback  
     ScrollingList widget, 574  
 XtNmultiClickTimeout  
     OLIT Toolkit Resources, 24  
 XtNmultiObjectCount  
     OLIT Toolkit Resources, 24  
 XtNnoneSet  
     Exclusives widget, 291  
     FlatExclusives widget, 354  
 XtNnoPreviewText  
     FontChooser widget, 378  
 XtNnumChildren  
     Composite Resources, 34  
 XtNnumCurrentItems  
     ScrollingList widget, 575  
 XtNnumItemFields  
     Flat Resources, 69  
 XtNnumItems  
     Flat Resources, 69  
     ScrollingList widget, 575  
 XtNolDefaultFont, 88  
     OLIT Toolkit Resources, 24  
 XtNopenAccelerator  
     FileChooser widget, 313  
 XtNopenButtonWidget  
     FileChooser widget, 315  
 XtNopenFolderAccelerator  
     FileChooser widget, 313  
 XtNopenFolderCallback  
     FileChooser widget, 305  
 XtNopenFolderLabel  
     FileChooser widget, 319  
 XtNopenFolderMnemonic  
     FileChooser widget, 313  
 XtNopenLabel  
     FileChooser widget, 319  
 XtNopenMnemonic  
     FileChooser widget, 313



---

XtNopenPromptString  
    FileChooser widget, 318

XtNoperation  
    FileChooser widget, 301  
    FileChooserShell widget, 327

XtNorientation  
    Gauge widget, 409  
    RubberTile widget, 515  
    Scrollbar widget, 530  
    Slider widget, 604

XtNoutputDocumentCallback  
    FileChooser widget, 306

XtNoverrideRedirect  
    Shell Resources, 43

XtNownSelectionCallback  
    DropTarget widget, 284

XtNpasteLabel  
    TextEdit widget, 643

XtNpasteMnemonic  
    TextEdit widget, 643

XtNpointerWarping  
    FileChooserShell widget, 328  
    NoticeShell widget, 451  
    PopupWindowShell widget, 496  
    Scrollbar widget, 530  
    Slider widget, 604

XtNpopdownCallback  
    Shell Resources, 43

XtNpopupCallback  
    Shell Resources, 43

XtNposition  
    Caption widget, 244  
    CheckBox widget, 256  
    FlatCheckBox widget, 345  
    ScrollingList widget, 575

XtNpostModifyCallback  
    TextLine widget, 709

XtNpostModifyNotification  
    TextEdit widget, 649

XtNpreferredPointSizes  
    FontChooser widget, 379

XtNprefMaxWidth  
    ScrollingList widget, 575

XtNprefMinWidth  
    ScrollingList widget, 575

XtNpreModifyCallback  
    TextLine widget, 708

XtNpreviewBackground  
    FontChooser widget, 379

XtNpreviewBorderWidth  
    FontChooser widget, 379

XtNpreviewFontColor  
    FontChooser widget, 379

XtNpreviewForeground  
    FontChooser widget, 379

XtNpreviewHeight  
    FontChooser widget, 380

XtNpreviewPresent  
    FontChooser widget, 380

XtNpreviewSwitchLabel  
    FontChooser widget, 380

XtNpreviewSwitchOffLabel  
    FontChooser widget, 380

XtNpreviewSwitchOnLabel  
    FontChooser widget, 381

XtNpreviewText  
    FontChooser widget, 381

XtNpreviewWidget  
    AbbrevMenuButton widget, 232

XtNpreviousLabel  
    Scrollbar widget, 528

XtNpreviousMnemonic  
    Scrollbar widget, 529

XtNproportionLength  
    Scrollbar widget, 530

XtNpushpin  
    MenuShell widget, 436  
    VendorShell Resources, 56

XtNpushpinDefault  
    MenuShell widget, 436

XtNqueryGeometry  
    Stub widget, 627

XtNrealize  
    Stub widget, 627

XtNrecomputeHeight  
    ScrolledWindow widget, 553

---

XtNrecomputeSize  
     Caption widget, 244  
     CheckBox widget, 256  
     DropTarget widget, 285  
     Exclusives widget, 292  
     Gauge widget, 410  
     MenuButton widget, 421  
     Nonexclusives widget, 442  
     OblongButton widget, 482  
     RectButton widget, 506  
     Slider widget, 604  
     StaticText widget, 615  
 XtNrecomputeWidth  
     ScrolledWindow widget, 553  
     ScrollingList widget, 576  
 XtNreferenceName  
     Primitive Resources, 38  
 XtNreferenceStub  
     Stub widget, 627  
 XtNreferenceWidget  
     Primitive Resources, 39  
 XtNrefName  
     RubberTile widget, 516  
 XtNrefWidget  
     RubberTile widget, 516  
 XtNregisterFocusFunc  
     Stub widget, 628  
 XtNrepeatRate  
     Scrollbar widget, 531  
     Slider widget, 604  
     TextField widget, 682  
     TextLine widget, 710  
 XtNreset  
     PopupWindowShell widget, 494  
 XtNresetFactory  
     PopupWindowShell widget, 494  
 XtNresetFactoryLabel  
     PopupWindowShell widget, 495  
 XtNresetFactoryMnemonic  
     PopupWindowShell widget, 495  
 XtNresetLabel  
     PopupWindowShell widget, 495  
 XtNresetMnemonic  
     PopupWindowShell widget, 495  
 XtNresize  
     Stub widget, 628  
 XtNresizeCallback  
     DrawArea widget, 274  
 XtNresizeCorners  
     VendorShell Resources, 57  
 XtNrevertCallback  
     FontChooser widget, 381  
 XtNrevertLabel  
     FontChooser widget, 382  
 XtNrightFooterString  
     VendorShell Resources, 57  
 XtNrightFooterVisible  
     overriding, 57  
     VendorShell Resources, 57  
 XtNrightMargin  
     Gauge widget, 410  
     TextEdit widget, 649  
 XtNsameHeight  
     Flat Resources, 69  
 XtNsameSize  
     ControlArea widget, 266  
 XtNsameWidth  
     Flat Resources, 70  
 XtNsaveAccelerator  
     FileChooser widget, 313  
 XtNsaveAsAccelerator  
     FileChooser widget, 313  
 XtNsaveAsLabel  
     FileChooser widget, 319  
 XtNsaveAsMnemonic  
     FileChooser widget, 313  
 XtNsaveLabel  
     FileChooser widget, 319  
 XtNsaveMnemonic  
     FileChooser widget, 313  
 XtNsaveUnder  
     Shell Resources, 44  
 XtNscale  
     CheckBox widget, 257  
     FileChooser widget, 302  
     OLIT Toolkit Resources, 25  
     Primitive Resources, 39

---

TextField widget, 682  
 XtNscreen  
     Core Resources, 31  
 XtNscrollingListItems  
     ScrollingList widget, 576  
 XtNscrollingListMode  
     ScrollingList widget, 576  
 XtNselect  
     CheckBox widget, 257  
     OblongButton widget, 483  
     RectButton widget, 507  
 XtNselectable  
     ScrollingList widget, 577  
     StaticText widget, 615  
 XtNselectDoesPreview  
     OLIT Toolkit Resources, 25  
 XtNselectEnd  
     TextEdit widget, 650  
 XtNselectionAtom  
     DropTarget widget, 285  
 XtNselectProc  
     FlatExclusives widget, 354  
 XtNselectStart  
     TextEdit widget, 650  
 XtNsensitive  
     Core Resources, 31  
 XtNset  
     CheckBox widget, 257  
     FlatExclusives widget, 354  
     RectButton widget, 507  
 XtNsetDefaults  
     PopupWindowShell widget, 494  
 XtNsetDefaultsLabel  
     PopupWindowShell widget, 495  
 XtNsetDefaultsMnemonic  
     PopupWindowShell widget, 495  
 XtNsetValues  
     Stub widget, 628  
 XtNsetValuesAlmost  
     Stub widget, 629  
 XtNsetValuesHook  
     Stub widget, 629  
 XtNshellTitle, 97  
 VendorShell Resources, 57  
 XtNshiftName  
     OLIT Toolkit Resources, 25  
 XtNshowAccelerators  
     OLIT Toolkit Resources, 26  
 XtNshowGlyphs  
     FileChooser widget, 302  
 XtNshowInactive  
     FileChooser widget, 308  
 XtNshowMnemonics  
     OLIT Toolkit Resources, 26  
 XtNshowPage  
     Scrollbar widget, 531  
     ScrolledWindow widget, 553  
 XtNsizeLabel  
     FontChooser widget, 382  
 XtNsizeOf  
     NumericField widget, 462  
 XtNsliderMax  
     Gauge widget, 410  
     Scrollbar widget, 531  
     Slider widget, 605  
 XtNsliderMin  
     Gauge widget, 410  
     Scrollbar widget, 531  
     Slider widget, 605  
 XtNsliderMoved  
     Scrollbar widget, 532  
     Slider widget, 605  
 XtNsliderValue  
     Gauge widget, 410  
     Scrollbar widget, 534  
     Slider widget, 606  
 XtNsource  
     TextEdit widget, 650  
 XtNsourceType  
     TextEdit widget, 651  
 XtNspace  
     Caption widget, 245  
     conflict, 516  
     RubberTile widget, 516  
     ScrollingList widget, 577  
 XtNspan

---

- Gauge widget, 411
- Slider widget, 606
- XtNstopPosition
  - Scrollbar widget, 534
  - Slider widget, 606
- XtNstring
  - StaticText widget, 616
  - TextField widget, 682
  - TextLine widget, 710
- XtNstrip
  - StaticText widget, 616
- XtNstyleLabel
  - FontChooser widget, 383
- XtNsubstituteShellVariables
  - FileChooser widget, 312
- XtNtabTable
  - TextEdit widget, 651
- XtNtextArea
  - NoticeShell widget, 451
- XtNtextEditWidget
  - TextField widget, 683
- XtNtextField
  - ScrollingList widget, 577
- XtNtextFormat, 85, 87
  - Caption widget, 245
  - CheckBox widget, 257
  - FileChooser widget, 302
  - FileChooserShell widget, 328
  - FontChooser widget, 383
  - FontChooserShell widget, 390
  - NoticeShell widget, 451
  - Primitive Resources, 39
  - TextField widget, 683
- XtNthreeD
  - OLIT Toolkit Resources, 26
- XtNticks
  - Gauge widget, 411
  - Slider widget, 607
- XtNtickUnit
  - Gauge widget, 411
  - Slider widget, 607
- XtNtitle
  - WMSHELL Resources, 49
- XtNtitleEncoding
  - WMSHELL Resources, 50
- XtNtopMargin
  - TextEdit widget, 651
- XtNtopToHereLabel
  - Scrollbar widget, 528
- XtNtopToHereMnemonic
  - Scrollbar widget, 529
- XtNtransient
  - WMSHELL Resources, 50
- XtNtranslations
  - Core Resources, 32
- XtNtraversalHandlerFunc
  - Stub widget, 629
- XtNtraversalOn
  - Primitive Resources, 40
- XtNtype
  - NumericField widget, 462
- XtNtypefaceLabel
  - FontChooser widget, 383
- XtNunderline
  - TextLine widget, 710
- XtNundoLabel
  - TextEdit widget, 643
- XtNundoMnemonic
  - TextEdit widget, 643
- XtNunselect
  - CheckBox widget, 257
  - RectButton widget, 507
- XtNunselectProc
  - FlatExclusives widget, 354
- XtNupdateDisplay
  - TextLine widget, 710
- XtNupperControlArea
  - PopupWindowShell widget, 496
- XtNuserData
  - Primitive Resources, 40
  - VendorShell Resources, 58
- XtNuserDeleteItems
  - ScrollingList widget, 578
- XtNuserFolders
  - FileChooser widget, 310
- XtNuserFoldersMaxCount

---

FileChooser widget, 311  
 XtNuserMakeCurrent  
     ScrollingList widget, 579  
 XtNuseSetValCallback  
     Scrollbar widget, 534  
     Slider widget, 607  
 XtNvalidateCallback  
     NumericField widget, 462  
 XtNvalue  
     NumericField widget, 463  
 XtNvAutoScroll  
     ScrolledWindow widget, 550  
 XtNverification  
     TextField widget, 683  
 XtNverify  
     PopupWindowShell widget, 496  
 XtNverifyCallback  
     FileChooserShell widget, 328  
 XtNviewableItems  
     ScrollingList widget, 580  
 XtNviewHeight  
     ScrollingList widget, 580  
 XtNvInitialDelay  
     ScrolledWindow widget, 550  
 XtNvisual  
     DrawArea widget, 274  
     Shell Resources, 44  
 XtNvMenuPane  
     ScrolledWindow widget, 550  
 XtNvPad  
     ControlArea widget, 264  
     Flat Resources, 64  
 XtNvScrollbar  
     ScrolledWindow widget, 551  
 XtNvSliderMoved  
     ScrolledWindow widget, 551  
 XtNvSpace  
     ControlArea widget, 264  
     Flat Resources, 64  
     FlatExclusives widget, 353  
     StaticText widget, 615  
 XtNvStepSize  
     ScrolledWindow widget, 552  
 XtNwaitForWm  
     WMSHELL Resources, 50  
 XtNweight  
     RubberTile widget, 517  
 XtNwidth  
     Core Resources, 32  
     StaticText widget, 616  
     Stub widget, 630  
 XtNwidthInc  
     WMSHELL Resources, 50  
 XtNwindow  
     Stub widget, 630  
 XtNwindowGroup  
     WMSHELL Resources, 51  
 XtNwindowHeader  
     VendorShell Resources, 58  
 XtNwinGravity  
     WMSHELL Resources, 51  
 XtNwmProtocol  
     VendorShell Resources, 58  
 XtNwmProtocolInterested  
     VendorShell Resources, 59  
 XtNwmTimeout  
     WMSHELL Resources, 52  
 XtNwrap  
     StaticText widget, 616  
 XtNwrapMode  
     TextEdit widget, 651  
 XtNx  
     Core Resources, 32  
 XtNxAddWidth  
     Form widget, 401  
 XtNxAttachOffset  
     Form widget, 401  
 XtNxAttachRight  
     Form widget, 401  
 XtNxOffset  
     Form widget, 402  
 XtNxRefName  
     Form widget, 402  
 XtNxRefWidget  
     Form widget, 403  
 XtNxResizable

---

Form widget, 403  
XtNxtDefaultFont, 25  
XtNxtDefaultFontSet, 25  
XtNxVaryOffset  
Form widget, 404  
XtNy  
Core Resources, 32  
XtNyAddHeight  
Form widget, 401  
XtNyAttachBottom  
Form widget, 401  
XtNyAttachOffset  
Form widget, 401  
XtNyOffset  
Form widget, 402  
XtNyRefName  
Form widget, 402  
XtNyRefWidget  
Form widget, 403  
XtNyResizable  
Form widget, 403  
XtNyVaryOffset  
Form widget, 404  
XtOpenDisplay(), 102  
XtOwnSelection(), 136  
XtPopDown(), 445  
XtPopup(), 488  
XtResizeWidget(), 631  
XtSetLanguageProc(), 90  
XtSetMappedWhenManaged(), 31  
XtSetSensitive(), 28  
XtSetValues(), 631  
XtToolkitInitialize(), 102  
XtVaCreateManagedWidget(), 40  
XtWindow(), 269, 630  
XwcDrawString(), 40  
XYBitmap, 481, 505  
XYPixmap, 419, 481, 505

## Z

ZPixmap, 419, 481, 505