Common Resources 2

This chapter describes resources that are common to several widgets.

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

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`.

2

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 `XtSetValues()`.
G	(Get.) The application can get the resource using `XtGetValues()`.
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 Properties or `xrdb` 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 Intrinsics 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
XtNbeep	OlDefine	OL_BEEP_ALWAYS	SGIO
XtNbeepVolume	int	0	SGIO
XtNcolorTupleList	OlColorTuple	NULL	GIO
XtNcontrolName	String	"Ctrl"	IOD
XtNdragRightDistance	Dimension	100 (pixels)	SGIO

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.

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 ≤ `XtNbeepVolume` ≤ 100

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

XtNcolorTupleList

Class	Type	Default	Access
XtCColorTupleList	OlColorTuple	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.

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.

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 Intrinsics 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

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
XtCOlDefaultFont	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).

In the C locale, the default value of `XtNolDefaultFont` 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 Intrinsics `XtNxtDefaultFont` (or `XtNxtDefaultFontSet`) resource, it may override the `XtNolDefaultFont` resource. See "XtNfont" on page 26 for a detailed description of the exact algorithm used.

XtNscale

Class	Type	Default	Access
XtCScale	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
XtCSelectDoesPreview	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
XtCShiftName	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.

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 ≤ `XtNborderWidth` ≤ min(`XtNwidth`, `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.

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 ≤ `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.

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 Intrinsics 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.

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.

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.

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:

- 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
XtCFontColor	Pixel	XtDefaultForeground	SGID

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
XtCForeground	Pixel	XtDefaultForeground	SGID

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
XtCInputFocusColor	Pixel	Red	SGID

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

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
XtCMnemonic	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
XtCReferenceName	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.

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
XtCReferenceWidget	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
XtCScale	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
XtCTextFormat	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.

`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
XtCTraversalOn	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
XtCUserData	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

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
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
XtNwidthInc	int	XtUnspecifiedShellInt	SGI

XtNallowShellResize

Class	Type	Default	Access
XtCAllowShellResize	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
XtCCreatePopupChildProc	XtCreatePopupChildProc	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.

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

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 ≤ `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 ≤ `XtNiconX`
–1 ≤ `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.

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 $1 \le$ `XtNmaxAspectX`
-1 = `XtNmaxAspectY` or $1 \le$ `XtNmaxAspectY`
-1 = `XtNminAspectX` or $1 \le$ `XtNminAspectX`
-1 = `XtNminAspectY` or $1 \le$ `XtNminAspectY`

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

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

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

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.

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
XtCMaxHeight	int	OL_IGNORE	SGI
XtCMaxWidth	int	OL_IGNORE	SGI
XtCMinHeight	int	OL_IGNORE	SGI
XtCMinWidth	int	OL_IGNORE	SGI

Synopsis: The range allowed for the size of the shell widget's window.
Values: `XtNminHeight` ≤ `XtNmaxHeight`
`XtNminWidth` ≤ `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
XtCTitle	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.

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 Intrinsics Programming Manual*.

XtNwidthInc

Class	Type	Default	Access
XtCWidthInc	int	XtUnspecifiedShellInt	SGI

Synopsis: The resizing increment for Shell widgets.
Values: 0 ≤ `XtNwidthInc`

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
XtCWindowGroup	Window	XtUnspecifiedWindow	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
XtCWinGravity	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 Intrinsics Reference Manual* for further details on `XtNwinGravity`.

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	OlFont	XtDefaultFontSet	SGI
XtNimStatusStyle	OlImStatusStyle	OL_NO_STATUS	GI
XtNleftFooterString	OlStr	NULL	SGI
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

Table 2-8 VendorShell Resources Summary (Continued)

Name	Type	Default	Access
XtNshellTitle	OlStr	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.

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	OlImStatusStyle	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.

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.

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
XtCUserData	XtPointer	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
XtCWindowHeader	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
XtCWMProtocol	XtCallbackList	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

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	SGI

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

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.

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
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

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
XtCGravity	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

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 ≤ `XtNhPad`
0 ≤ `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 ≤ `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.

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.

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.

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 < `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.

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.