Widget Reference (D–F) 7

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 Intrinsics Programming Manual* for details on this function.

7

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.

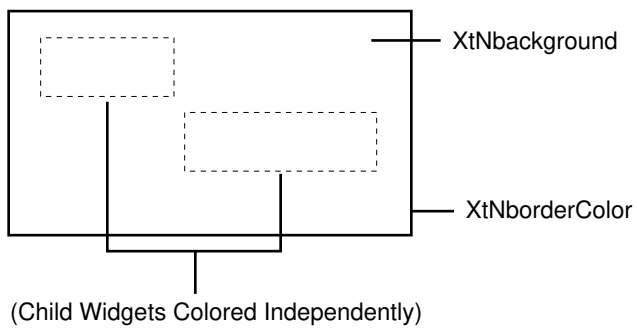

Figure 7-1 DrawArea Coloration

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)

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[1]

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.

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 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 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

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 DrawArea widget has no activation types besides the ones in “Common Activation Types” on page 68.

See Also

“BulletinBoard Widget” on page 225.

7

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

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

7

DropTarget Widget

Table 7-8 DropTarget 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 7-9 DropTarget Pixmap Resources

Name	Type	Default	Access
XtNpixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNrecomputeSize	Boolean	FALSE	SGI

Table 7-10 DropTarget Resources

Name	Type	Default	Access
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

OlDropTargetCallbackStruct

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;
    OlDnDDropSiteID         dropsiteid;
    OlDnDTriggerOperation   operation;
    Boolean                 send_done;
    Boolean                 forwarded;
    int                     eventcode;
    Boolean                 sensitivity;
} OlDropTargetCallbackStruct;
```

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

root_y The root-relative y-coordinate at which the drop occurred

selection The Selection Atom for the selection transfer of the drop information

time The timestamp of the pertinent event; for example, the trigger or preview

dropsiteid The ID of the drop site on which the drop occurred

operation One of the following values:

`OlDnDTriggerCopyOp`	The object is copied
`OlDnDTriggerMoveOp`	The object is moved

send_done 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 `OlDnDDragNDropDone()` when the selection transaction is completed successfully.

forwarded The drop has been forwarded to this target by another drop site.

eventcode One of the following values:

`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

sensitivity Indicates whether the drop site the cursor has entered is interested in receiving drops.

Since `OlDropTargetCallbackStruct` is shared among all the callbacks of the DropTarget widget, not all the fields of `OlDropTargetCallbackStruct` 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 `OlDropTargetCallbackStruct` for all the callbacks of the DropTarget widget. The rows of the table contain fields of `OlDropTargetCallbackStruct`. 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 OlDropTargetCallbackStruct Field Validity

Field	Preview	Trigger	Ownselection	Animate
reason	+	+	+	+
widget	+	+	+	+

Table 7-11 OlDropTargetCallbackStruct Field Validity *(Continued)*

Field	Preview	Trigger	Ownselection	Animate
window	+	+	–	–
root_x	+	+	–	–
root_y	+	+	–	–
selection	–	+	–	–
time	+	+	+	+
dropsiteid	+	+	–	–
operation	–	+	–	–
send_done	–	+	–	–
forwarded	+	+	–	–
eventcode	+	–	–	+
sensitivity	–	–	–	+

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`).

XtNdndAnimateCallback

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 `XtNdndAcceptCursor` and `XtNdndRejectCursor`.)

XtNdndCopyCursor

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).

XtNdndMoveCursor

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).

XtNdndPreviewHints

Class	Type	Default	Access
XtCDndPreviewHints	OlDnDSitePreviewHints	OlDnDSitePreviewNone	SGI

Synopsis: When a DropTarget preview callback is activated.
Values: One of the following values:

```
OlDnDSitePreviewNone
OlDnDSitePreviewEnterLeave
OlDnDSitePreviewMotion
OlDnDSitePreviewBoth
OlDnDSitePreviewDefaultSite
OlDnDSitePreviewForwarded
OlDnDSitePreviewInsensitive
```

This resource is used by *destination* DropTarget widgets only.

See "Drag and Drop Functions" for more details, in particular the section "OlDnDSitePreviewHints" on page 117.

XtNdndPreviewCallback

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`).

XtNdndRejectCursor

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 `XtNdndAnimateCallback`).

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).

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 `OlDnDOwnSelection()` (see page 126).

XtNrecomputeSize

Class	Type	Default	Access
XtCRecomputeSize	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
XtCSelectionAtom	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.

Activation Types

The following table lists the activation types used by the DropTarget.

Table 7-12 DropTarget 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 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.

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

Table 7-13 Exclusives Core Resources *(Continued)*

Name	Type	Default	Access
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

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

Table 7-15 Exclusives 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 7-16 Exclusives Resources

Name	Type	Default	Access
XtNlayoutType	OlDefine	OL_FIXEDROWS	SGI
XtNmeasure	int	1	SGI
XtNnoneSet	Boolean	FALSE	SGI
XtNrecomputeSize	Boolean	TRUE	SGI

XtNlayoutType

Class	Type	Default	Access
XtCLayoutType	OlDefine	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

Class	Type	Default	Access
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

Class	Type	Default	Access
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).

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.

7

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 then 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

Table 7-18 FileChooser Core Resources *(Continued)*

Name	Type	Default	Access
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

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

Table 7-20 FileChooser Manager Resources

Name	Type	Default	Access
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

Name	Type	Default	Access
XtNorientation	OlDefine	OL_VERTICAL	SGI

Table 7-22 FileChooser Resources

Name	Type	Default	Access
XtNapplicationFolders	OlFolderList	NULL	SGI
XtNapplicationFoldersMaxCount	Cardinal	5	GI
XtNcancelAccelerator	String	NULL	SGI
XtNcancelButtonWidget	Widget	(calculated)	G
XtNcancelCallback	XtCallbackList	NULL	SGIO
XtNcancelLabel	OlStr	"Cancel"	SGI
XtNcancelMnemonic	OlMnemonic	'\0'	SGI
XtNcommandButtonWidget	Widget	(calculated)	G
XtNcomparisonFunc	OlComparisonFunc	(string case-insensitive ascending)	SGI
XtNcurrentFolder	String	(see [1])	SGI
XtNcurrentFolderLabelString	OlStr	"Current Folder:"	SGI
XtNcurrentFolderLabelWidget	Widget	(calculated)	G
XtNcurrentFolderWidget	Widget	(calculated)	G
XtNdefaultDocumentName	OlStr	"Untitled1"	SGI
XtNdefaultDocumentSuffix	OlStr	".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	OlStr	"Select a folder and click %."	SGI
XtNfollowSymlinks	Boolean	TRUE	SGI
XtNfont	OlFont	XtDefaultFont	SGI
XtNfontColor	Pixel	XtDefaultForeground	SGID
XtNforeground	Pixel	XtDefaultForeground	SGID
XtNgotoButtonWidget	Widget	(calculated)	G
XtNgotoHomeAccelerator	String	NULL	SGI
XtNgotoHomeButtonWidget	Widget	(calculated)	G

Table 7-22 FileChooser Resources *(Continued)*

Name	Type	Default	Access
XtNgotoHomeLabel	OlStr	"Home"	SGI
XtNgotoHomeMnemonic	OlMnemonic	'\0'	SGI
XtNgotoLabel	OlStr	"Go To"	SGI
XtNgotoMenuWidget	Widget	(calculated)	G
XtNgotoPromptString	OlStr	"Type in the path to the folder and press Return."	SGI
XtNgotoPromptWidget	Widget	(calculated)	G
XtNgotoTypeInWidget	Widget	(calculated)	G
XtNgoUpOneFolderLabel	OlStr	"...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	OlStr	"Include"	SGI
XtNincludeMnemonic	OlMnemonic	'\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	OlStr	"Open Folder"	SGI
XtNopenFolderMnemonic	OlMnemonic	'\0'	SGI
XtNopenLabel	OlStr	"Open"	SGI
XtNopenMnemonic	OlMnemonic	'\0'	SGI
XtNopenPromptString	OlStr	"Select a document or folder and click %."	SGI

Table 7-22 FileChooser Resources *(Continued)*

Name	Type	Default	Access
XtNoperation	OlDefine	OL_OPEN	GI
XtNoutputDocumentCallback	XtCallbackList	NULL	SGIO
XtNsaveAccelerator	String	NULL	SGI
XtNsaveAsAccelerator	String	NULL	SGI
XtNsaveAsLabel	OlStr	"Save As"	SGI
XtNsaveAsMnemonic	OlMnemonic	'\0'	SGI
XtNsaveLabel	OlStr	"Save"	SGI
XtNsaveMnemonic	OlMnemonic	'\0'	SGI
XtNscale	int	(see [2])	SGI
XtNshowGlyphs	Boolean	TRUE	SGI
XtNshowInactive	Boolean	TRUE	SGI
XtNsubstituteShellVariables	Boolean	TRUE	SGI
XtNtextFormat	OlStrRep	OlDefaultTextFormat	GI
XtNuserFolders	OlFolderList	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

7

FileChooser Widget

Base Resources

XtNcurrentFolder

Class	Type	Default	Access
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

Class	Type	Default	Access
XtCFollowSymlinks	Boolean	TRUE	SGI

Synopsis: The traversal of symbolic links, rather 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

Class	Type	Default	Access
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

Class	Type	Default	Access
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*.

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:

reason	Can be any one of: OL_REASON_OPEN_FOLDER OL_REASON_OPEN_DOCUMENT OL_REASON_OUTPUT_DOCUMENT OL_REASON_FILTER OL_REASON_LIST_CHOICE OL_REASON_FOLDER_OPENED OL_REASON_CANCEL
version	Version of the FileChooser
extension	Reserved for future use
operation	Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE
current_folder	The folder that the FileChooser is in currently (absolute path name)

XtNinputDocumentCallback

Class	Type	Default	Access
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).

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:

reason	OL_REASON_INPUT_DOCUMENT or OL_REASON_OUTPUT_DOCUMENT
version	Version of the FileChooser
extension	Reserved for future use
operation	OL_OPEN or OL_INCLUDE
current_folder	The folder that the FileChooser is in currently (absolute pathname)
request_document_folder	The folder in which the FileChooser has been requested for a document (absolute pathname)
request_document	The base document name being requested
request_document_node	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:

name	The name of the file (a folder or document)
is_folder	TRUE if the file is a folder
operational	Not used
filtered	Not used
active	TRUE is the file is selected
glyph	The glyph used in the content list
sbufp	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;
```

Fields in the *call_data* structure are:

reason	OL_REASON_OPEN_FOLDER
version	Version of the FileChooser
extension	Reserved for future use
operation	Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE
current_folder	The folder that the FileChooser is in currently (absolute pathname)
request_folder	The folder that the FileChooser has been requested to move to. This field is NULL if the request is for the *current_folder*.
request_folder_node	A struct of type `OlFNavNode`; 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:

reason	OL_REASON_OUTPUT_DOCUMENT
version	Version of the FileChooser
extension	Reserved for future use
operation	OL_SAVE or OL_SAVE_AS
current_folder	The folder that the FileChooser is in currently (absolute pathname)
request_document_folder	The folder in which the FileChooser has been requested for a document (absolute pathname).
request_document	The base document name being requested
request_document_node	A struct of type `OlFNavNode`; see "XtNinputDocumentCallback" on page 293

File Filtering

XtNfilterProc

Class	Type	Default	Access
XtCCallback	XtCallbackList	NULL	SGI

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:

reason	OL_REASON_FILTER
version	Version of the FileChooser

extension	Reserved for future use
operation	Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE
current_folder	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 Intrinsics 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

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 Intrinsics 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);
```

The following sorting styles are provided by the toolkit and may be specified with these symbolic constants:

Table 7-23 FileChooser Sorting Styles

Sort Style	OlComparisonFunc()
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

Class	Type	Default	Access
XtCBooleanDefault	Boolean	TRUE	SGI

Synopsis: The treatment of the tilde (~) character.
Values: `TRUE/"true"` – The tilde is expanded according to the `csh`(1) or `ksh`(1) rules.
`FALSE/"false"` – The tilde is treated as a literal ~ character.

XtNsubstituteShellVariables

Class	Type	Default	Access
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

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
XtCComponentWidget	Widget	(calculated)	G

Synopsis: The IDs for various widgets, as follows:

XtNcancelButtonWidget	The Cancel button widget
XtNcommandButtonWidget	The Open, Save, Save As, or Include button widget
XtNcurrentFolderLabelWidget	The Current Folder: label widget
XtNcurrentFolderWidget	The widget displaying the pathname of the current folder
XtNdocumentListWidget	The folder content scrolling list widget
XtNdocumentNameLabelWidget	The label widget of the document name type-in field (in the Save and Save As operations)
XtNdocumentNameTypeInWidget	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
XtNopenButtonWidget	The Open button widget
XtNgotoButtonWidget	The Go To: button widget
XtNgotoHomeButtonWidget	The Home button widget in the Go To: menu
XtNgotoMenuWidget	The Go To: menu 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:

reason	OL_REASON_CANCEL
version	Version of the FileChooser
extension	Reserved for future use
operation	Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE
current_folder	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:

reason	OL_REASON_FOLDER_OPENED

version	Version of the FileChooser
extension	Reserved for future use
operation	Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE
current_folder	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:

reason	OL_REASON_LIST_CHOICE
version	Version of the FileChooser
extension	Reserved for future use
operation	Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE
current_folder	The folder that the FileChooser is in currently (absolute pathname)
chosen_item	Label of the item selected in the list
chosen_item_node	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.

XtNdefaultDocumentSuffix

Class	Type	Default	Access
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

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 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"
```

7

FileChooserShell Widget

Coloration

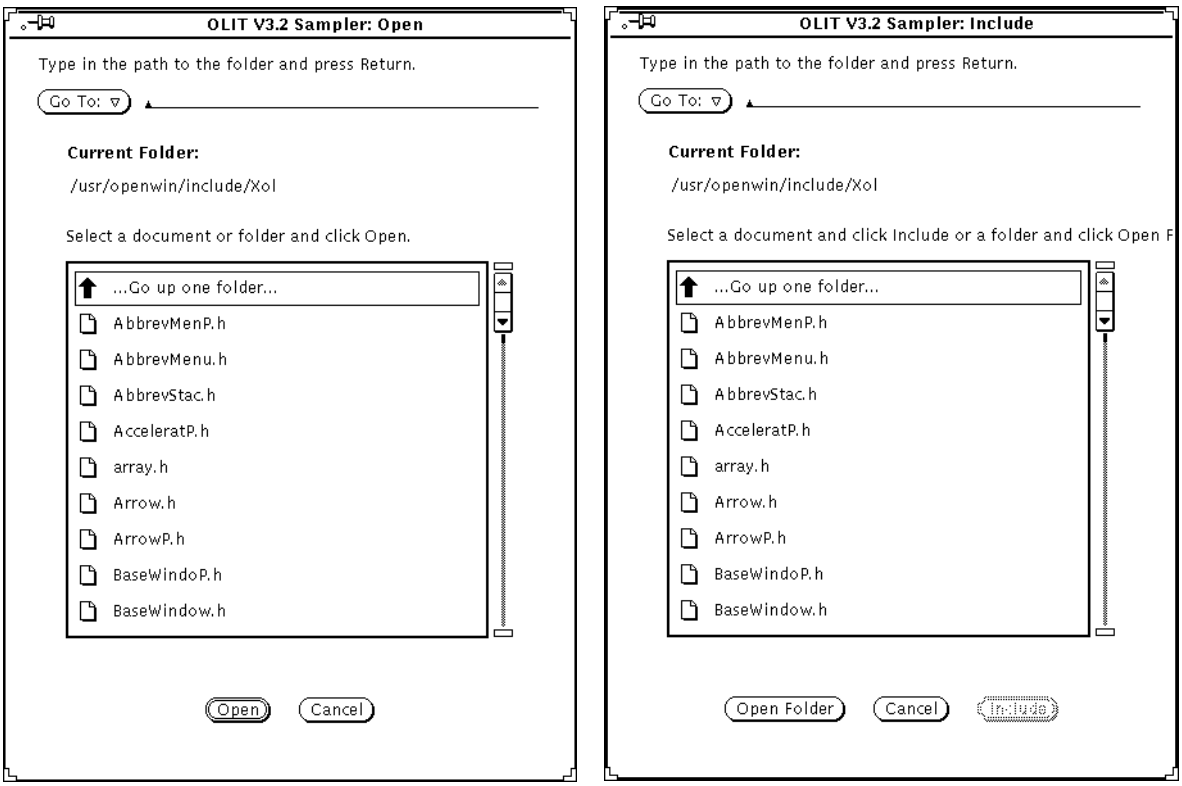

Figure 7-4 FileChooserShell Appearance (Open, Include Operations)

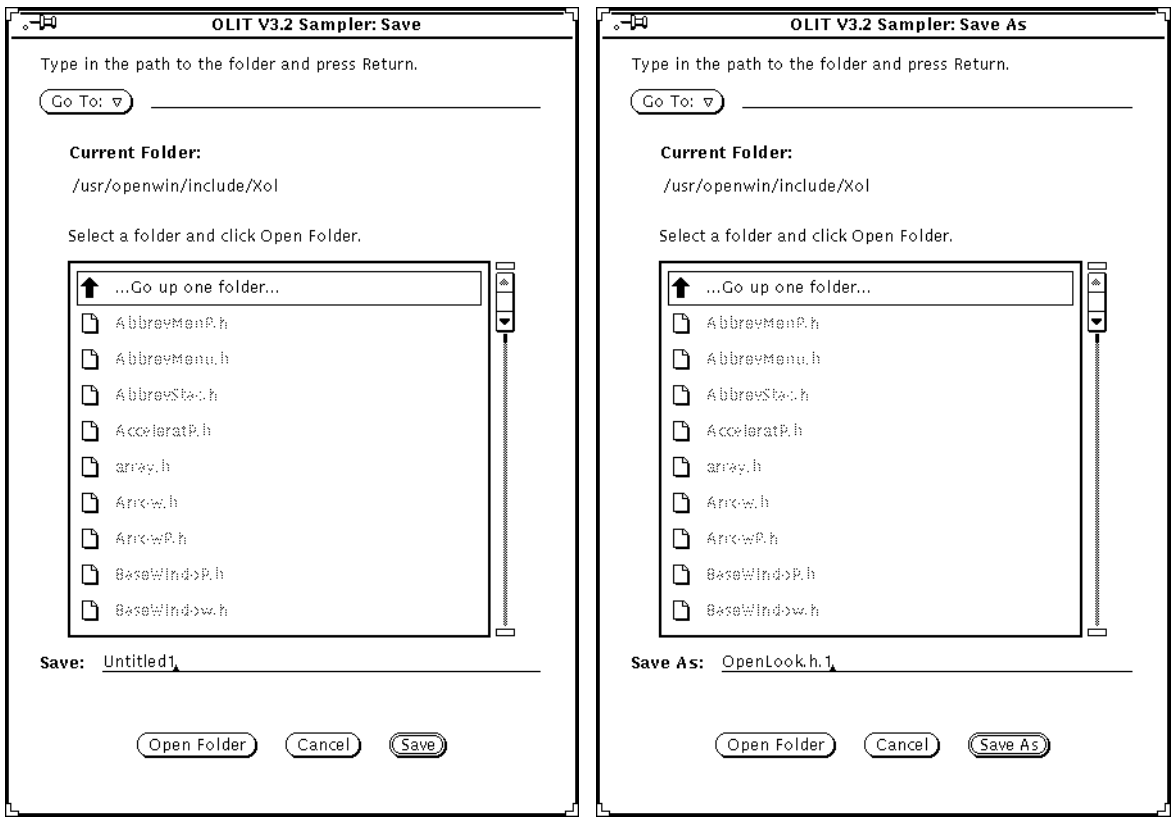

Figure 7-5 FileChooserShell Appearance (Save, Save As Operations)

7

FileChooserShell Widget

Resources

Table 7-25 FileChooserShell 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	(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

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

Table 7-27 FileChooserShell Shell Resources

Name	Type	Default	Access
XtNallowShellResize	Boolean	TRUE	SGI

Table 7-27 FileChooserShell Shell Resources *(Continued)*

Name	Type	Default	Access
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

Name	Type	Default	Access
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[1])	SGI
XtNtitleEncoding	Atom	(see[2])	SGI
XtNtransient	Boolean	(see[3])	SGI
XtNwaitForWm	Boolean	TRUE	GI
XtNwidthInc	int	XtUnspecifiedShellInt	GI

Table 7-28 FileChooserShell WMShell Resources *(Continued)*

Name	Type	Default	Access
XtNwindowGroup	Window	XtUnspecifiedWindow	GI
XtNwinGravity	int	XtUnspecifiedShellInt	GI
XtNwmTimeout	int	(see[4])	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

Name	Type	Default	Access
XtNbusy	Boolean	FALSE	SGI
XtNdefaultImName	String	NULL	SGI
XtNfooterPresent	Boolean	TRUE	SGI
XtNfocusWidget	Widget	(see [1])	SGI
XtNimFontSet	OlFont	XtDefaultFontSet	SGI
XtNimStatusStyle	OlImStatusStyle	OL_NO_STATUS	GI
XtNleftFooterString	OlStr	NULL	SGI
XtNleftFooterVisible	Boolean	TRUE	SGI
XtNmenuButton	Boolean	FALSE	GI
XtNmenuType	OlDefine	OL_MENU_LIMITED	SGI
XtNpushpin	OlDefine	OL_OUT	SGI
XtNresizeCorners	Boolean	TRUE	SGI
XtNrightFooterString	OlStr	NULL	SGI
XtNrightFooterVisible	Boolean	TRUE	SGI
XtNshellTitle	OlStr	(see Description)	SGI
XtNtextFormat	OlStrRep	(default text format)	SGI
XtNuserData	XtPointer	NULL	SGI
XtNwindowHeader	Boolean	TRUE	GI
XtNwinType	OlDefine	OL_WT_CMD	GI
XtNwmProtocolInterested	int	OL_WM_DELETE_WINDOW | OL_WM_TAKE_FOCUS	I

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*.

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:

reason	OL_REASON_VERIFY
extension	Reserved for future use
operation	Can be any one of OL_OPEN, OL_SAVE, OL_SAVE_AS, or OL_INCLUDE
current_folder	The folder that the FileChooser is in (absolute pathname)
accept_verify	Set to TRUE by default in the *call_data*. 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 suppling 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 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:

item_index	Index of the item responsible for the callback.
items	The head of item list that contains the item initiating the callback
num_items	The total number of items in the item list
item_fields	The list of resource name used to identify the records in the item list.
num_item_fields	The number of resource names contained in *item_fields*.

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, fcp->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.

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 Intrinsics 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.

Components

The following diagram illustrates the components of the FlatCheckBox:

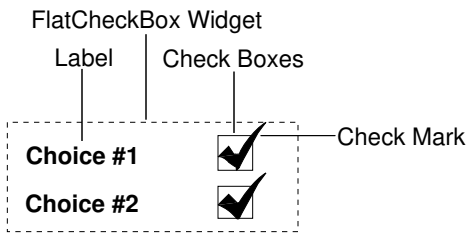

Figure 7-6 FlatCheckBox Components

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.

7

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[1]	Boolean	(calculated)	G
XtNbackground[1]	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap[1]	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth[1]	Dimension	0	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	Cardinal	(parent's)	GI
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged[1]	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive[1]	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[1]	String	NULL	SGI
XtNacceleratorText[1]	String	NULL	SGI
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNfont[1]	OlFont	XtDefaultFont	SGID
XtNfontColor[1]	Pixel	XtDefaultForeground	SGID
XtNforeground[1]	Pixel	XtDefaultForeground	SGID
XtNinputFocusColor[1]	Pixel	XtDefaultForeground	SGID
XtNmnemonic[1]	unsigned char	'\0'	SGI
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNscale	int	12	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI
XtNtraversalOn[1]	Boolean	TRUE	SGI
XtNuserData[1]	XtPointer	NULL	SGI

1. This resource is available on the container and as an item resource.

Table 7-35 FlatCheckBox Flat Resources[1]

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[2]	OlStr	NULL	SGI

Table 7-35 FlatCheckBox Flat Resources[1] *(Continued)*

Name	Type	Default	Access
XtNlabelImage[2]	XImage *	NULL	SGI
XtNlabelJustify[2]	OlDefine	OL_LEFT	SGI
XtNlabelTile[2]	Boolean	FALSE	SGI
XtNlayoutHeight	OlDefine	OL_MINIMIZE	SGI
XtNlayoutType	OlDefine	OL_FIXEDROWS	SGI
XtNlayoutWidth	OlDefine	OL_MINIMIZE	SGI
XtNmanaged[2]	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[1]

Name	Type	Default	Access
XtNclientData[2]	XtPointer	NULL	SGI
XtNdefault[2]	Boolean	FALSE	SGI
XtNdim	Boolean	FALSE	SGI
XtNhSpace	Dimension	OL_IGNORE	SGI
XtNnoneSet	Boolean	FALSE	SGI
XtNselectProc[2]	XtCallbackProc	NULL	SGI
XtNset[2]	Boolean	FALSE	SGI
XtNunselectProc[2]	XtCallbackProc	NULL	SGI
XtNvSpace	Dimension	OL_IGNORE[3]	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[1]	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.

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 be 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

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[1]	Boolean	(calculated)	G
XtNbackground[1]	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap[1]	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth[1]	Dimension	0	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	Cardinal	(parent's)	GI

Table 7-39 FlatExclusives Core Resources *(Continued)*

Name	Type	Default	Access
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged[1]	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive[1]	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

Name	Type	Default	Access
XtNaccelerator[1]	String	NULL	SGI
XtNacceleratorText[1]	String	NULL	SGI
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNfont[1]	OlFont	XtDefaultFont	SGID
XtNfontColor[1]	Pixel	XtDefaultForeground	SGID
XtNforeground[1]	Pixel	XtDefaultForeground	SGID
XtNinputFocusColor[1]	Pixel	XtDefaultForeground	SGID
XtNmnemonic[1]	unsigned char	'\0'	SGI
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNscale	int	12	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI
XtNtraversalOn[1]	Boolean	TRUE	SGI
XtNuserData[1]	XtPointer	NULL	SGI

1. This resource is available on the container and as an item resource.

Table 7-41 FlatExclusives Flat Resources[1]

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[2]	OlStr	NULL	SGI
XtNlabelImage[2]	XImage *	NULL	SGI
XtNlabelJustify[2]	OlDefine	OL_LEFT	SGI
XtNlabelTile[2]	Boolean	FALSE	SGI
XtNlayoutHeight	OlDefine	OL_MINIMIZE	SGI
XtNlayoutType	OlDefine	OL_FIXEDROWS	SGI
XtNlayoutWidth	OlDefine	OL_MINIMIZE	SGI
XtNmanaged[2]	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 7-42 FlatExclusives Resources

Name	Type	Default	Access
XtNclientData[1]	XtPointer	NULL	SGI
XtNdefault[1]	Boolean	FALSE	SGI
XtNdim	Boolean	FALSE	SGI
XtNhSpace	Dimension	OL_IGNORE	G
XtNnoneSet	Boolean	FALSE	SGI
XtNselectProc[1]	XtCallbackProc	NULL	SGI
XtNset[1]	Boolean	FALSE	SGI
XtNunselectProc[1]	XtCallbackProc	NULL	SGI
XtNvSpace	Dimension	OL_IGNORE[2]	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:

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.

XtNnoneSet

Class	Type	Default	Access
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

Class	Type	Default	Access
XtCCallbackProc	XtCallbackProc	NULL	SGI

Synopsis: The callback procedure invoked when an unset item is set by user input.

XtNset

Class	Type	Default	Access
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

Class	Type	Default	Access
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

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.

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[1]	Boolean	(calculated)	G
XtNbackground[1]	Pixel	XtDefaultBackground	SGID
XtNbackgroundPixmap[1]	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGID
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderWidth[1]	Dimension	0	SGI
XtNcolormap	Colormap	(parent's)	SGI
XtNdepth	Cardinal	(parent's)	GI
XtNdestroyCallback	XtCallbackList	NULL	SGIO
XtNheight	Dimension	0	SGI
XtNmappedWhenManaged[1]	Boolean	TRUE	SGI
XtNscreen	Screen *	(parent's)	G
XtNsensitive[1]	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.

7

FlatNonexclusives Widget

Table 7-45 FlatNonexclusives Primitive Resources

Name	Type	Default	Access
XtNaccelerator[1]	String	NULL	SGI
XtNacceleratorText[1]	String	NULL	SGI
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNfont[1]	OlFont	XtDefaultFont	SGID
XtNfontColor[1]	Pixel	XtDefaultForeground	SGID
XtNforeground[1]	Pixel	XtDefaultForeground	SGID
XtNinputFocusColor[1]	Pixel	XtDefaultForeground	SGID
XtNmnemonic[1]	unsigned char	'\0'	SGI
XtNreferenceName	String	NULL	GI
XtNreferenceWidget	Widget	NULL	GI
XtNscale	int	12	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI
XtNtraversalOn[1]	Boolean	TRUE	SGI
XtNuserData[1]	XtPointer	NULL	SGI

1. This resource is available on the container and as an item resource.

Table 7-46 FlatNonexclusives Flat Resources[1]

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[2]	OlStr	NULL	SGI

Table 7-46 FlatNonexclusives Flat Resources[1] *(Continued)*

Name	Type	Default	Access
XtNlabelImage[2]	XImage *	NULL	SGI
XtNlabelJustify[2]	OlDefine	OL_LEFT	SGI
XtNlabelTile[2]	Boolean	FALSE	SGI
XtNlayoutHeight	OlDefine	OL_MINIMIZE	SGI
XtNlayoutType	OlDefine	OL_FIXEDROWS	SGI
XtNlayoutWidth	OlDefine	OL_MINIMIZE	SGI
XtNmanaged[2]	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[1]

Name	Type	Default	Access
XtNclientData[2]	XtPointer	NULL	SGI
XtNdefault[2]	Boolean	FALSE	SGI
XtNdim	Boolean	FALSE	SGI
XtNhSpace	Dimension	OL_IGNORE	SGI
XtNnoneSet	Boolean	FALSE	SGI
XtNselectProc[2]	XtCallbackProc	NULL	SGI
XtNset[2]	Boolean	FALSE	SGI
XtNunselectProc[2]	XtCallbackProc	NULL	SGI
XtNvSpace	Dimension	OL_IGNORE[3]	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.

7

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

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 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.

7

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

- 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

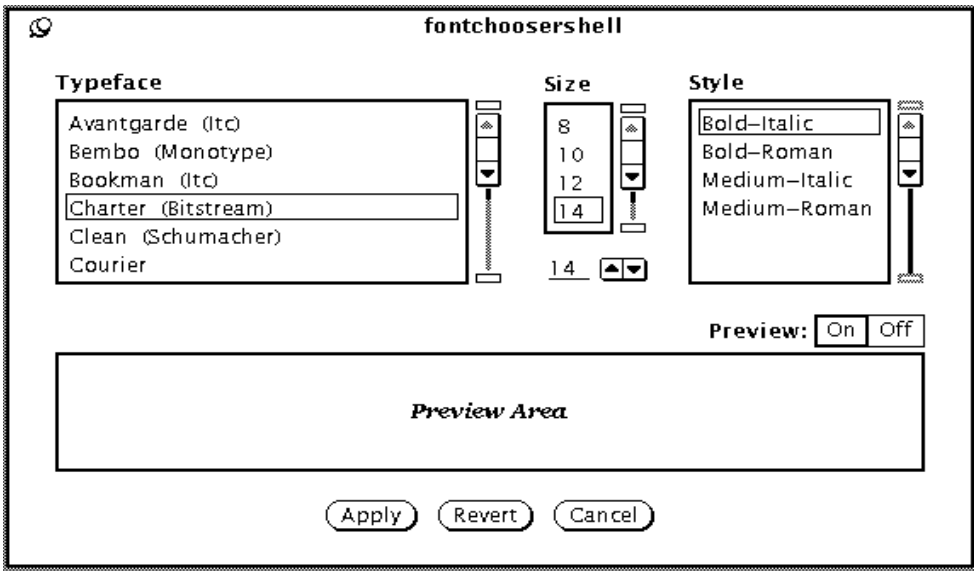

Figure 7-7 FontChooser Appearance

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-myoungjo-medium-r-normal--16-140-75-75-p-140-korean-0"
```

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-myoungjo-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-myoungjo-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-myoungjo-medium-r-normal--16-140-75-75-c-140-korean-0”:

```
hngmnj14: alias, -sun-myoungjo-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:

```
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
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

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 7-52 FontChooser RubberTile Resources

Name	Type	Default	Access
XtNorientation	OlDefine	OL_VERTICAL	G

Table 7-53 FontChooser Resources

Name	Type	Default	Access
XtNapplyCallback	XtCallbackList	NULL	SGIO
XtNapplyLabel	OlStr	"Apply"	GI
XtNattributeListHeight	Dimension	(calculated)	SGI
XtNcancelCallback	XtCallbackList	NULL	SGIO
XtNcancelLabel	OlStr	"Cancel"	GI
XtNchangedCallback	XtCallbackList	NULL	SGIO
XtNcharsetInfo	String	"iso8859-1"	G
XtNerrorCallback	XtCallbackList	NULL	SGIO
XtNextensionArea	Widget	NULL	G
XtNfontSearchSpec	String	"*-*-*-*-*-*-*-*-*-*-*-*-*-*"	GI
XtNinitialFontName	String	NULL	GI
XtNmaximumPointSize	Cardinal	99	GI
XtNnoPreviewText	OlStr	"The preview is turned off"	SGI
XtNpreferredPointSizes	String	"8 10 12 14 18 24"	GI
XtNpreviewHeight	Dimension	(calculated)	SGI
XtNpreviewBorderWidth	Dimension	(calculated)	GI
XtNpreviewBackground	Pixel	XtDefaultBackground	GI
XtNpreviewFontColor	Pixel	XtDefaultForeground	GI
XtNpreviewForeground	Pixel	XtDefaultForeground	GI
XtNpreviewPresent	Boolean	TRUE	SGI
XtNpreviewText	OlStr	"%T %S %s"	SGI
XtNrevertCallback	XtCallbackList	NULL	SGIO
XtNrevertLabel	OlStr	"Revert"	GI
XtNsizeLabel	OlStr	"Size"	GI
XtNstyleLabel	OlStr	"Style"	GI
XtNpreviewSwitchLabel	OlStr	"Preview:"	GI
XtNpreviewSwitchOffLabel	OlStr	"Off"	GI

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:

reason	OL_REASON_APPLY_FONT
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.

XtNapplyLabel

Class	Type	Default	Access
XtCApplyLabel	OlStr	"Apply"	GI

Synopsis: The label displayed on the APPLY 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.

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:

reason	OL_REASON_CHANGED_FONT
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.
previous_font_name	An XLFD font specification corresponding to the choices of attributes before the change.
previous_font	An `XFontStruct *` corresponding to *previous_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 *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 (XLFD) 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

error_num A number from the following table that identifies the specific error condition.

font_name An XLFD font specification corresponding to the current choice of attributes.

Table 7-54 XtNerrorCallback error_num Descriptions

error_num	Description
OL_FC_ERR_NO_FONTS_FOUND	No fonts found
OL_FC_ERR_NO_INITIAL_FONT	Initial font is not matched
OL_FC_ERR_BAD_FONT_SEARCH_SPEC	Font Search specification is not XLFD
OL_FC_ERR_MISSING_CHARSETS	Incomplete font description (missing charsets)

XtNextensionArea

Class	Type	Default	Access
XtCExtensionArea	Widget	NULL	G

Synopsis: The container for the application to extend the GUI of the FontChooser. The extension area is a widget of class RubberTile. Graphically, the extension area appears above the preview area, just below the row of scrolling lists.

Values: A RubberTile widget (see "RubberTile Widget" on page 502) instantiated the first time an `XtGetValues()` is done on the resource.

XtNfontSearchSpec

Class	Type	Default	Access
XtCFontSearchSpec	String	"*-*-*-*-*-*-*-*-*-*-*-*-*-*"	GI

Synopsis: The simple regular expression that constrains the set of fonts used by the FontChooser. See "Regular Expression Functions" on page 161.

Values: Any string that is a valid XLFD specification.

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.

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

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.

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

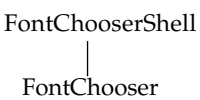

Figure 7-8 FontChooserShell Components

Coloration

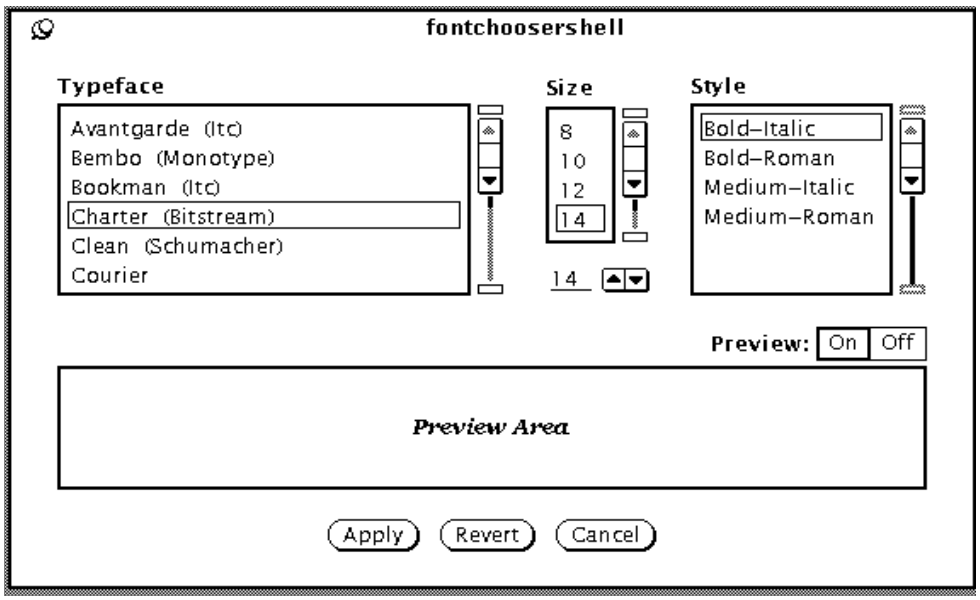

Figure 7-9 FontChooserShell Appearance

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

Table 7-56 FontChooserShell Core Resources *(Continued)*

Name	Type	Default	Access
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

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

Table 7-58 FontChooserShell 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 7-59 FontChooserShell WMShell Resources

Name	Type	Default	Access
XtNbaseHeight	int	XtUnspecifiedShellInt	SGI
XtNbaseWidth	int	XtUnspecifiedShellInt	SGI
XtNheightInc	int	XtUnspecifiedShellInt	SGI

Table 7-59 FontChooserShell WMShell Resources *(Continued)*

Name	Type	Default	Access
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

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

Table 7-60 FontChooserShell VendorShell Resources *(Continued)*

Name	Type	Default	Access
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
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

Table 7-61 FontChooserShell TransientShell Resources

Name	Type	Default	Access
XtNtransientFor	Widget	NULL	SGI

Table 7-62 FontChooserShell Resources

Name	Type	Default	Access
XtNfontChooserWidget	Widget	(calculated)	G
XtNtextFormat	OlStrRep	OlDefaultTextFormat	GI

XtNfontChooserWidget

Class	Type	Default	Access
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.

XtNtextFormat

Class	Type	Default	Access
XtCTextFormat	OlStrRep	OlDefaultTextFormat	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

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 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.

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

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

Table 7-64 FooterPanel Core Resources *(Continued)*

Name	Type	Default	Access
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

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

Table 7-66 FooterPanel 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

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.

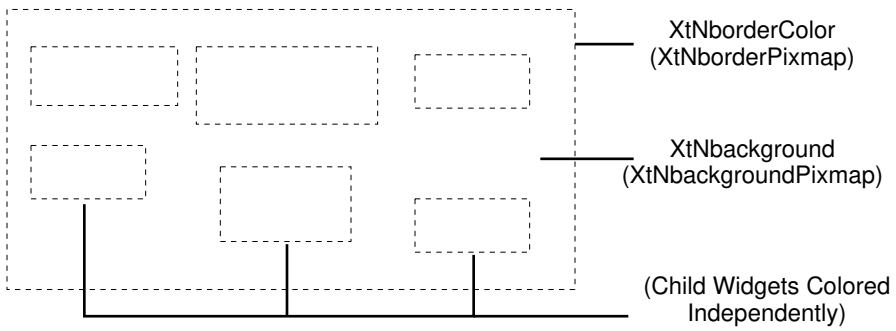

Figure 7-10 Form Coloration

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.

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., Consume_Width = Form_Width – 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

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 7-69 Form Composite Resources

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

Table 7-70 Form Manager Resources

Name	Type	Default	Access
XtNconsumeEvent	XtCallbackList	NULL	SGIO
XtNinputFocusColor	Pixel	Red	SGID

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

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
XtCXOffset	int	0	SGI
XtCYOffset	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
XtCXRefName	String	NULL	SGI
XtCYRefName	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

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
XtCXRefWidget	Widget	(form)	SGI
XtCYRefWidget	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
XtCXResizable	Boolean	FALSE	SGI
XtCYResizable	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.

XtNxVaryOffset/
XtNyVaryOffset

Class	Type	Default	Access
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

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 Form widget has no activation types besides the ones in "Common Activation Types" on page 68.

See Also

"RubberTile Widget" on page 502.