Section 2 - Motif and Xt Widget Classes

This page describes the format and contents of each reference page in Section 2, which covers each of the Motif and Xt Intrinsics widget types.

Name

Widget – a brief description of the widget.

Synopsis

Public Headers:
The files to include when you use this widget.

Class Name:
The name of the widget class; used as the resource class for each instance of the widget.

Class Hierarchy:
The superclasses of this widget, listed in superclass-to-subclass order. The arrow symbol (→) indicates a subclass.

Class Pointer:
The global variable that points to the widget class structure. This is the value used when creating a widget.

Instantiation:
C code that instantiates the widget, for widgets that can be instantiated. For the widgets and gadgets in the Motif toolkit, we have shown how to instantiate the widget using `XtCreateWidget`(). Each widget and gadget has a convenience creation routine of the general form:

```
Widget XmCreateobject ( Widget    parent
                        String    name
                        ArgList   arglist,
                        Cardinal  argcount)
```

where *object* is the shorthand for the class.

Functions/Macros:
Functions and/or macros specific to this widget class.

Availability

This section describes the availability of the widget class across various versions of Motif. The section is omitted if the widget class has always been present in the toolkit.

Description

This section gives an overview of the widget class and the functionality it provides.

Traits

This section appears for any traits that are set by the widget class. The Trait mechanisms are available in Motif 2.0 and later.

New Resources

This section presents a table of the resources that are newly defined by each widget class (not inherited from a superclass). In addition to the resource's name, class, data type, and default value, a fifth column lists a code consisting of one or more of the letters C, S, and G. This code indicates whether the resource can be set when the widget is created (C), whether it can be set with XtSetValues() (S), and whether it can be read with `XtGetValues()` (G). A brief description of each new resource follows the table. For resources whose values are defined constants, these constants are listed. Unless otherwise noted, they are defined in `<Xm/Xm.h>`.

Other New Resources

If present, these sections describe resources associated with specific uses of the widget; for example, RowColumn widget resources for use with simple creation routines, or Text widget resources for use in text input.

Callback Resources

This section presents a table of the callback resources that are newly defined by this class. The table lists the name of each resource along with its reason constant.

Callback Structure

This section lists the structure(s) associated with the object's callback functions.

New Constraint Resources

This section defines any constraint resources that are newly defined by each widget class (not inherited from a superclass). In addition to the resource's name, class, data type, and default value, a fifth column lists a code consisting of one or more of the letters C, S, and G. This code indicates whether the constraint resource can be set when a child widget is created (C), whether it can be set with XtSetValues() (S), and whether it can be read with `XtGetValues()` (G). A brief description of each new constraint resource follows the table. For resources whose values are defined constants, these constants are listed. Unless otherwise noted, they are defined in `<Xm/Xm.h>`.

Procedures

This section lists any procedure or function prototypes associated with the widget.

Default Resource Values

This section presents a table of the default resource values that are set when a compound object is created.

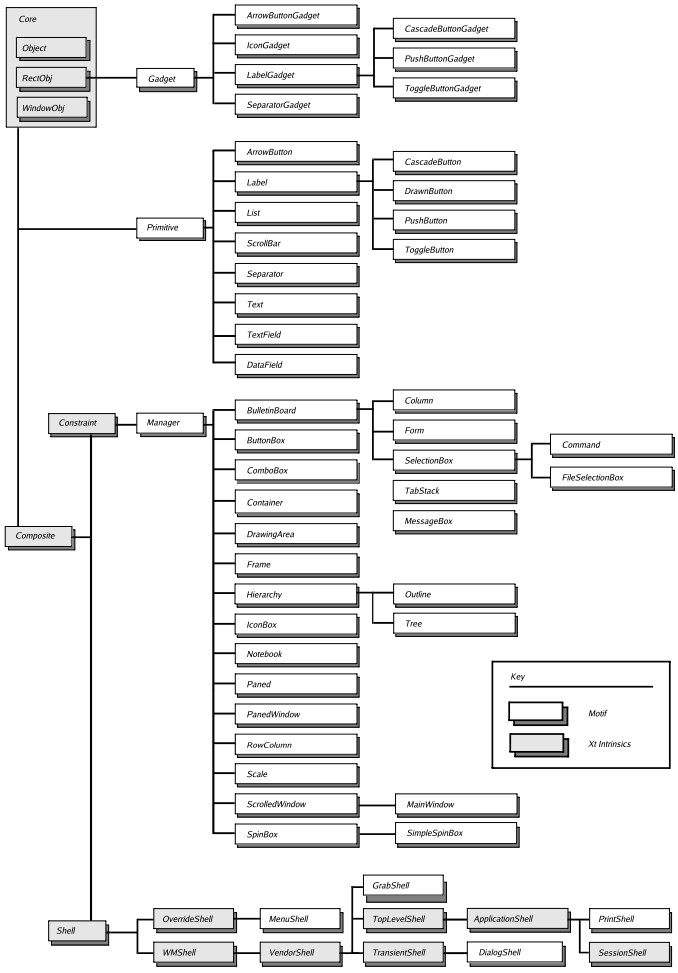
Inherited Resources

This section presents an alphabetically arranged table of inherited resources, along with the superclass that defines them.

Widget Hierarchy

This section presents the widget instance hierarchy that results from creating a compound object.

The full widget hierarchy is shown in Figure 1.


Core
Object
RectObj
WindowObj
Gadget
ArrowButtonGadget
IconGadget
LabelGadget
SeparatorGadget
CascadeButtonGadget
PushButtonGadget
ToggleButtonGadget
Primitive
ArrowButton
Label
List
ScrollBar
Separator
Text
TextField
DataField
CascadeButton
DrawnButton
PushButton
ToggleButton
Constraint
Manager
Composite
BulletinBoard
ButtonBox
ComboBox
Container
DrawingArea
Frame
Hierarchy
IconBox
Notebook
Paned
PanedWindow
RowColumn
Scale
ScrolledWindow
SpinBox
Column
Form
SelectionBox
TabStack
MessageBox
Command
FileSelectionBox
Outline
Tree
MainWindow
SimpleSpinBox
Key
Motif
Xt Intrinsics
Shell
OverrideShell
WMShell
MenuShell
VendorShell
GrabShell
TopLevelShell
TransientShell
ApplicationShell
DialogShell
PrintShell
SessionShell

Translations

This section presents the translations associated with each widget or gadget. Because the button events and key events used in Motif do not necessarily correspond to the events in the X Window System, the Motif toolkit has created a mechanism called *virtual bindings*. Virtual bindings link the translations used in Motif to their X event counterparts. The "Translations" sections list their events in terms of these virtual bindings. In order to understand the syntax used in the "Translations" sections of these reference pages, you must understand the correspondence between virtual bindings and actual keysyms or buttons. The following tables describe the virtual bindings of events.

Virtual Modifier	Actual Modifier
MAlt	<Mod1>
MCtrl	<Ctrl>
MShift	<Shift>
MLink	<Ctrl><Shift>
MMove	<Shift>
MCopy	<Ctrl>

Virtual Button	Actual Button Events
BCustom	<Btn3>
BTransfer	<Btn2>
BExtend	<Shift><Btn1>
BMenu	<Btn3>
BSelect	<Btn1>
BToggle	<Ctrl><Btn1>

Virtual Key	Actual Key Events
KActivate	<Key>Return <Ctrl><Key>Return <Key>osfActivate
KAddMode	<Key>osfAddMode
KBackSpace	<Key>osfBackSpace
KBackTab	<Shift><Key>Tab
KBeginData	<Ctrl><Key>osfBeginLine
KBeginLine	<Key>osfBeginLine

Virtual Key	Actual Key Events
KCancel	<Key>osfCancel
KClear	<Key>osfClear
KCopy	<Key>osfCopy <Ctrl><Key>osfInsert
KCut	<Key>osfCut <Shift><Key>osfDelete
KDelete	<Key>osfDelete
KDeselectAll	<Ctrl><Key>backslash
KDown	<Key>osfDown
KEndData	<Ctrl><Key>osfEndLine
KEndLine	<Key>osfEndLine
KEnter	<Key>Return
KEscape	<Key>Escape
KExtend	<Ctrl><Shift><Key>space <Shift><Key>osfSelect
KHelp	<Key>osfHelp
KInsert	<Key>osfInsert
KLeft	<Key>osfLeft
KMenu	<Key>osfMenu
KMenuBar	<Key>osfMenuBar
KNextField	<Key>Tab <Ctrl><Key>Tab
KNextMenu	<Ctrl><Key>osfDown <Ctrl><Key>osfRight
KPageDown	<Key>osfPageDown
KPageLeft	<Ctrl<Key>osfPageUp <Key>osfPageUp
KPageRight	<Ctrl><Key>osfPageDown
KPageUp	<Key>osfPageUp
KPaste	<Key>osfPaste <Shift><Key>osfInsert
KPrevField	<Shift><Key>Tab <Ctrl><Key><Shift><Tab>

Virtual Key	Actual Key Events
KPrevMenu	<Ctrl><Key>osfUp <Ctrl><Key>osfLeft
KPrimaryCopy	<Ctrl><Key>osfPrimaryPaste <Mod1><Key>osfCopy <Mod1><Ctrl><Key>osfInsert
KPrimaryCut	<Mod1><Key>osfPrimaryPaste <Mod1><Key>osfCut <Mod1><Shift><Key>osfDelete
KPrimaryPaste	<Key>osfPrimaryPaste
KQuickCopy	<Ctrl><Key>osfQuickPaste
KQuickCut	<Mod1><Key>osfQuickPaste
KQuickExtend	<Shift><Key>osfQuickPaste
KQuickPaste	<key>osfQuickPaste
KReselect	<Ctrl><Shift><Key>osfSelect
KRestore	<Ctrl><Shift><Key>osfInsert
KRight	<Key>osfRight
KSelect	<Key>space <Ctrl><Key>space <Key>osfSelect
KSelectAll	<Ctrl><Key>slash
KSpace	<Key>space
KTab	<Key>Tab
KUndo	<Key>osfUndo <Mod1><Key>osfBackSpace
KUp	<Key>osfUp
KAny	<Key>

Keysyms that begin with the letters osf are not defined by the X server. These keysyms are generated at run time by a client, interpreted by `XmTranslateKey()`, and used by the translation manager when the server sends an actual key event. An application maintains a mapping between osf keysyms and actual keysyms that is based on information that is retrieved at application startup. This information comes from one of the following sources, listed in order of precedence:

• The `XmNdefaultVirtualBindings` resource in a resource database. A sample specification is shown below:

```
*defaultVirtualBindings:\
osfBackSpace:       <Key>BackSpace \n\
osfInsert:          <Key>InsertChar \n\
osfDelete:          <Key>DeleteChar
```

• A property on the root window. *mwm* sets this property on startup. It can also be set by the *xmbind* client in Motif 1.2 or later, or the prior startup of another Motif application.

• A file named *.motifbind*, in the user's home directory. In this file, the previous specification would be typed as follows:

```
osfBackSpace:       <Key>BackSpace
osfInsert:          <Key>InsertChar
osfDelete:          <Key>DeleteChar
```

• A vendor-specific set of bindings located using the file *xmbind.alias*. If this file exists in the user's home directory, it is searched for a pathname associated with the vendor string or the vendor string and vendor release. If the search is unsuccessful, Motif continues looking for *xmbind.alias* in the directory specified by XMBINDDIR or in */usr/lib/Xm/bindings* if the variable is not set. If this file exists, it is searched for a pathname as before. If either search locates a pathname and the file exists, the bindings in that file are used. An *xmbind.alias* file contains lines of the following form:

```
"vendor_string[vendor_release]"bindings_file
```

• Via fixed fallback defaults. *osf* keysym strings have the fixed fallback default bindings listed below:

```
osfActivate         <unbound>
osfAddMode          <Shift> F8
osfBackSpace        Backspace
osfBeginLine        Home
osfClear            Clear
osfCopy             <unbound>
osfCut              <unbound>
osfDelete           Delete
osfDown             Down
osfEndLine          End
osfCancel           <Escape>
osfHelp             F1
osfInsert           Insert
osfLeft             Left
osfMenu             F4
osfMenuBar          F10
```

osfPageDown	Next
osfPageLeft	<unbound>
osfPageRight	<unbound>
osfPageUp	Prior
osfPaste	<unbound>
osfPrimaryPaste	<unbound>
osfQuickPaste	<unbound>
osfRight	Right
osfSelect	Select
osfUndo	Undo
osfUp	Up

Action Routines

This section describes the action routines that are listed in the "Translations" section.

Behavior

This section describes the keyboard and mouse events that affect gadgets, which do not have translations or actions.

Additional Behavior

This section describes any additional widget behavior that is not provided by translations and actions.

See Also

This section refers you to related functions and widget classes. The numbers in parentheses following each reference refer to the sections of this book in which they are found.

Name

ApplicationShell widget class – the main shell for an application.

Synopsis

Public Headers:
<Xm/Xm.h>
<X11/Shell.h>

Class Name:
ApplicationShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → TopLevelShell → ApplicationShell

Class Pointer:
`applicationShellWidgetClass`

Instantiation:
```
widget = XtAppInitialize (...)
```
or
```
widget = XtAppCreateShell (app_name, app_class,
                           applicationShellWidget-
                           Class,...)
```

Functions/Macros:
`XtAppCreateShell()`, `XtVaAppCreateShell()`, `XtIsApplica-tionShell()`

Availability

From X11R6, the ApplicationShell is considered deprecated: you should give preference to the SessionShell widget class.

Description

An ApplicationShell is the normal top-level window for an application. It does not have a parent and it is at the root of the widget tree. An application should have only one ApplicationShell, unless the application is implemented as multiple logical applications. Normally, an application will use TopLevelShell widgets for other top-level windows.

An ApplicationShell is returned by the call to `XtVaAppInitialize()`. It can also be created explicitly with a call to `XtVaAppCreateShell()`.

New Resources

ApplicationShell defines the following resources:

Name	Class	Type	Default	Access
XmNargc	XmCArgc	int	0	CSG
XmNargv	XmCArgv	String *	NULL	CSG

`XmNargc`
Number of arguments in `XmNargv`.

`XmNargv`
List of command-line arguments used to start the application. This is the standard C *argv*, passed in the call to `XtAppInitialize()`. It is used to set the WM_COMMAND property for this window, which is the argument list required by a session manager to restart the application if necessary. The resource value can be changed at appropriate points if some specific internal state has been reached from which the application can be directly restarted.

Inherited Resources

ApplicationShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of `XmNborderWidth` is reset to 0 by VendorShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNmappedWhenManaged	Core
XmNallowShellResize	Shell	XmNmaxAspectX	WMShell
XmNancestorSensitive	Core	XmNmaxAspectY	WMShell
XmNaudibleWarning	VendorShell	XmNmaxHeight	WMShell
XmNbackground	Core	XmNmaxWidth	WMShell
XmNbackgroundPixmap	Core	XmNminAspectX	WMShell
XmNbaseHeight	WMShell	XmNminAspectY	WMShell
XmNbaseWidth	WMShell	XmNminHeight	WMShell
XmNborderColor	Core	XmNminWidth	WMShell
XmNborderPixmap	Core	XmNmwmDecorations	VendorShell
XmNborderWidth	Core	XmNmwmFunctions	VendorShell
XmNbuttonFontList	VendorShell	XmNmwmInputMode	VendorShell
XmNbuttonRenderTable	VendorShell	XmNmwmMenu	VendorShell
XmNchildren	Composite	XmNnumChildren	Composite
XmNcolormap	Core	XmNoverrideRedirect	Shell

Resource	Inherited From	Resource	Inherited From
XmNcreatePopupChildProc	Shell	XmNpopdownCallback	Shell
XmNdefaultFontList	VendorShell	XmNsaveUnder	Shell
XmNdeleteResponse	VendorShell	XmNscreen	Core
XmNdepth	Core	XmNsensitive	Core
XmNdestroyCallback	Core	XmNshellUnitType	VendorShell
XmNgeometry	Shell	XmNtextFontList	VendorShell
XmNheight	Core	XmNtextRenderTable	VendorShell
XmNheightInc	WMShell	XmNtitle	WMShell
XmNiconic	TopLevelShell	XmNtitleEncoding	WMShell
XmNiconMask	WMShell	XmNtoolTipEnable	VendorShell
XmNiconName	TopLevelShell	XmNtoolTipPostDuration	VendorShell
XmNiconNameEncoding	TopLevelShell	XmNtoolTipString	VendorShell
XmNiconPixmap	WMShell	XmNtransient	WMShell
XmNiconWindow	WMShell	XmNtranslations	Core
XmNinitialResourcesPersistent	Core	XmNuseAsyncGeometry	VendorShell
XmNinitialState	WMShell	XmNunitType	VendorShell
XmNinput	WMShell	XmNvisual	Shell
XmNinputMethod	VendorShell	XmNwaitForWm	WMShell
XmNinputPolicy	VendorShell	XmNwidth	Core
XmNinsertPosition	Composite	XmNwidthInc	WMShell
XmNkeyboardFocusPolicy	VendorShell	XmNwindowGroup	WMShell
XmNlabelFontList	VendorShell	XmNwinGravity	WMShell
XmNlabelRenderTable	VendorShell	XmNwmTimeout	WMShell
XmNlayoutDirection	VendorShell	XmNx	Core
XmNpopupCallback	Shell	XmNy	Core
XmNpreeditType	VendorShell		

The VendorShell superclass installs a handler which intercepts the window manager WM_DELETE_WINDOW message. The handler is inherited by sub-classes of VendorShell, and has the behavior that if XmNdeleteResponse is XmDESTROY, and the widget is an instance of an ApplicationShell, then the application context associated with the widget is destroyed, followed by a call to exit().

See Also

`Composite`(2), `Core`(2), `SessionShell`(2), `Shell`(2),

TopLevelShell(2), VendorShell(2), WMShell(2).

Name

Composite widget class – the fundamental widget that can have children.

Synopsis

Public Headers:
<Xm/Xm.h>
<X11/Composite.h>

Class Name:
Composite

Class Hierarchy:
Core → Composite

Class Pointer:
compositeWidgetClass

Instantiation:
Composite is an Intrinsics meta-class and is not normally instantiated.

Functions/Macros:
`XtIsComposite()`

Description

Composite widgets contain other widgets. A Composite widget supports an arbitrary number of children, although derived classes may impose a limit for whatever reason. Composite handles the geometry management of its children. It also manages the destruction of descendants when it is destroyed. Children of a Composite widget are ordered, and Composite provides the means to sort or place the list of children in some logical order.

New Resources

Composite defines the following resources:

Name	Class	Type	Default	Access
XmNchildren	XmCReadOnly	WidgetList	NULL	G
XmNinsertPosition	XmCInsertPosition	XtOrderProc	NULL	CSG
XmNnumChildren	XmCReadOnly	Cardinal	0	G

XmNchildren
: List of widget's children.

XmNinsertPosition
: Points to an `XtOrderProc()` function that is called to determine the position at which each child is inserted into the XmNchildren array. Composite supplies a default function that appends children in the order of creation.

XmNnumChildren
Length of the list in XmNchildren.

Procedures

XtOrderProc
An `XtOrderProc` is a pointer to a function, specified as follows:

```
typedef Cardinal (*XtOrderProc) (Widget);
```

An `XtOrderProc` function is called by the insert_child method of a Composite or derived class, when a new child is created within the widget. The function has a single parameter, which is the widget ID of the new child. The function returns the number of children that go before the new child in the XmNchildren array. Composite supplies a default function that simply appends new children in the order of creation. Sub-classes may supply alternative default behavior. Programmers may supply their own XtOrderProc to sort children in some specified manner.

Inherited Resources

Composite inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them.

Name	Inherited From	Name	Inherited From
XmNaccelerators	Core	XmNheight	Core
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNscreen	Core
XmNborderColor	Core	XmNsensitive	Core
XmNborderPixmap	Core	XmNtranslations	Core
XmNborderWidth	Core	XmNwidth	Core
XmNcolormap	Core	XmNx	Core
XmNdepth	Core	XmNy	Core
XmNdestroyCallback	Core		

See Also

`Core`(2).

Name

Constraint widget class – a widget that provides constraint resources for its children.

Synopsis

Public Headers:
<Xm/Xm.h>
<X11/Constraint.h>

Class Name:
Constraint

Class Hierarchy:
Core → Composite → Constraint

Class Pointer:
constraintWidgetClass

Instantiation:
Constraint is an Intrinsics meta-class and is not normally instantiated.

Functions/Macros:
`XtIsConstraint()`

Description

Constraint widgets are so named because they may manage the geometry of their children based on constraints associated with each child. These constraints can be as simple as the maximum width and height the parent allows the child to occupy, or as complicated as how other children change if a child is moved or resized. Constraint widgets let a parent define resources that are supplied for their children. For example, if a Constraint parent defines the maximum width and height for its children, these resources are retrieved for each child as if they are resources that are defined by the child widget itself.

New Resources

Constraint does not define any new resources.

Inherited Resources

Constraint inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them.

Name	Inherited From	Name	Inherited From
XmNaccelerators	Core	XmNheight	Core
XmNancestorSensitive	Core	XmNinsertPosition	Composite
XmNbackground	Core	XmNinitialResourcesPersistent	Core
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core

Name	Inherited From	Name	Inherited From
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNscreen	Core
XmNborderWidth	Core	XmNsensitive	Core
XmNchildren	Composite	XmNtranslations	Core
XmNcolormap	Core	XmNwidth	Core
XmNdepth	Core	XmNx	Core
XmNdestroyCallback	Core	XmNy	Core

See Also

`Composite`(2), `Core`*(2)*.

Name

Core widget class – the fundamental class for windowed widgets.

Synopsis

Public Header:
<Xm/Xm.h>
<X11/Core.h>

Class Name:
Core

Class Hierarchy:
Object → RectObj → *unnamed* → Core

Class Pointer:
widgetClass or coreWidgetClass

Instantiation:
Core is an Intrinsics meta-class and is not normally instantiated.

Functions/Macros:
`XtIsWidget`()

Description

Core is the fundamental class for windowed widgets. All widgets with windows are subclasses of Core. The Object and RectObj classes support gadgets (windowless widgets). Core is sometimes instantiated for use as a basic drawing area.

New Resources

Core defines the following resources (some of which are actually defined by the Object and RectObj classes)

Name	Class	Type	Default	Access
XmNaccelerators	XmCAccelerators	XtAccelerators	dynamic	CSG
XmNancestorSensitive	XmCSensitive	Boolean	dynamic	G
XmNbackground	XmCBackground	Pixel	dynamic	CSG
XmNbackgroundPixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNborderColor	XmCBorderColor	Pixel	XtDefaultForeground	CSG
XmNborderPixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNborderWidth	XmCBorderWidth	Dimension	1	CSG
XmNcolormap	XmCColormap	Colormap	dynamic	CSG
XmNdepth	XmCDepth	int	dynamic	CSG
XmNdestroyCallback	XmCCallback	XtCallbackList	NULL	C
XmNheight	XmCHeight	Dimension	dynamic	CSG

Name	Class	Type	Default	Access
XmNinitialResourcesPersistent	XmCInitialResourcesPersistent	Boolean	True	C
XmNmappedWhenManaged	XmCMappedWhenManaged	Boolean	True	CSG
XmNscreen	XmCScreen	Screen *	dynamic	CG
XmNsensitive	XmCSensitive	Boolean	True	CSG
XmNtranslations	XmCTranslations	XtTranslations	dynamic	CSG
XmNwidth	XmCWidth	Dimension	dynamic	CSG
XmNx	XmCPosition	Position	0	CSG
XmNy	XmCPosition	Position	0	CSG

XmNaccelerators
: A translation table bound with its actions for a widget. A destination widget can be set up to use this accelerator table.

XmNancestorSensitive
: Tells whether a widget's immediate parent should receive input. Default value is True if the widget is a top-level shell, copied from the XmNancestorSensitive resource of its parent if the widget is a popup shell, or the bitwise AND of the XmNsensitive and XmNancestorSensitive resources of the parent for other widgets.

XmNbackground
: Widget's background color.

XmNbackgroundPixmap
: Pixmap with which to tile the background, beginning at the upper-left corner.

XmNborderColor
: Pixel value that defines the color of the border.

XmNborderPixmap
: Pixmap with which to tile the border, beginning at the upper-left corner of the border.

XmNborderWidth
: Width (in pixels) of the window's border.

XmNcolormap
: Colormap used in converting to pixel values. Previously created pixel values are unaffected. The default value is the screen's default colormap for top-level shells or is copied from the parent for other widgets.

XmNdepth
: Number of bits allowed for each pixel. The Xt Intrinsics set this resource when the widget is created. As with the XmNcolormap resource, the default value comes from the screen's default or is copied from the parent.

XmNdestroyCallback
: List of callbacks invoked when the widget is destroyed.

XmNheight
: Window height (in pixels), excluding the border.

XmNinitialResourcesPersistent
: Tells whether resources should be reference counted. If True (default), it is assumed that the widget won't be destroyed while the application is running, and thus the widget's resources are not reference counted. Set this resource to False if your application might destroy the widget and will need to deallocate the resources.

XmNmappedWhenManaged
: If True (default), the widget becomes visible (is mapped) as soon as it is both realized and managed. If False, the application performs the mapping and unmapping of the widget. If changed to False after the widget is realized and managed, the widget is unmapped.

XmNscreen
: Screen location of the widget. The default value comes either from the screen's default or is copied from the parent.

XmNsensitive
: Tells whether a widget is sensitive to input. The `XtSetSensitive`() routine can be used to change a widget's sensitivity and to guarantee that if a parent has its XmNsensitive resource set to False, then its children will have their ancestor-sensitive flag set correctly.

XmNtranslations
: Points to a translation table; must be compiled with `XtParseTranslationTable`().

XmNwidth
: Window width (in pixels), excluding the border.

XmNx
: The x-coordinate of the widget's upper-left outer corner, relative to the upper-left inner corner of its parent.

XmNy
: The y-coordinate of the widget's upper-left outer corner, relative to the upper-left inner corner of its parent.

See Also

`Object`(2), `RectObj`(2).

Name

Object widget class – fundamental object class.

Synopsis

Public Headers:
<Xm/Xm.h>
<X11/Object.h>

Class Name:
Object

Class Hierarchy:
Object

Class Pointer:
objectClass

Instantiation:
Object is an Intrinsics meta-class and is not normally instantiated.

Functions/Macros:
`XtIsObject`()

Description

Object is the root of the class hierarchy; it does not have a superclass. All widgets and gadgets are subclasses of Object. Object encapsulates the mechanisms for resource management and is never instantiated.

New Resources

Object defines the following resources:

Name	Class	Type	Default	Access
XmNdestroyCallback	XmCCallback	XtCallbackList	NULL	C

XmNdestroyCallback
List of callbacks invoked when the Object is destroyed.

See Also

`Core`(2).

Name

OverrideShell widget class –a popup shell that bypasses window management.

Synopsis

Public Header:
<X11/Shell.h>

Class Name:
OverrideShell

Class Hierarchy:
Core → Composite → Shell → OverrideShell

Class Pointer:
overrideShellWidgetClass

Instantiation:
widget = `XtCreatePopupShell` (name, overrideShellWidgetClass,...)

Functions/Macros:
`XtIsOverrideShell`()

Description

OverrideShell is a direct subclass of Shell that performs no interaction with window managers. It is used for widgets, such as popup menus, that should bypass the window manager.

New Resources

OverrideShell does not define any new resources.

Inherited Resources

OverrideShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. OverrideShell sets the default values of both XmNoverrideRedirect and XmNsaveUnder to True.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinitialResourcesPersistent	Core
XmNallowShellResize	Shell	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNmappedWhenManaged	Core
XmNaudibleWarning	VendorShell	XmNnumChildren	Composite
XmNbackground	Core	XmNoverrideRedirect	Shell
XmNbackgroundPixmap	Core	XmNpopdownCalback	Shell
XmNborderColor	Core	XmNpopupCallback	Shell
XmNborderPixmap	Core	XmNsaveUnder	Shell
XmNborderWidth	Core	XmNscreen	Core

Resource	Inherited From	Resource	Inherited From
XmNchildren	Composite	XmNsensitive	Core
XmNcolormap	Core	XmNtranslations	Core
XmNcreatePopupChildProc	Shell	XmNvisual	Shell
XmNdepth	Core	XmNwidth	Core
XmNdestroyCallback	Core	XmNx	Core
XmNgeometry	Shell	XmNy	Core
XmNheight	Core		

See Also

`Composite`(2), `Core`(2), `Shell`(2).

Name

RectObj widget class – fundamental object class with geometry.

Synopsis

Public Header:
<Xm/Xm.h>
<X11/RectObj.h>

Class Name:
RectObj

Class Hierarchy:
Object → RectObj

Class Pointer:
rectObjClass

Instantiation:
RectObj is an Intrinsics meta-class and is not normally instantiated.

Functions/Macros:
`XtIsRectObj`()

Description

RectObj is a supporting superclass for widgets and gadgets, defined by the X toolkit intrinsics. All of the Motif widgets are ultimately derived from RectObj. It does not have a window, but it does have a height, width, and location, and it encapsulates the mechanisms for geometry management.

New Resources

RectObj defines the following resources:

Name	Class	Type	Default	Access
XmNancestorSensitive	XmCSensitive	Boolean	dynamic	G
XmNborderWidth	XmCBorderWidth	Dimension	1	CSG
XmNheight	XmCHeight	Dimension	dynamic	CSG
XmNsensitive	XmCSensitive	Boolean	True	CSG
XmNwidth	XmCWidth	Dimension	dynamic	CSG
XmNx	XmCPosition	Position	0	CSG
XmNy	XmCPosition	Position	0	CSG

XmNancestorSensitive
Tells whether a gadget's immediate parent should receive input. Default value is the bitwise AND of the XmNsensitive and XmNancestorSensitive resources of the parent.

XmNborderWidth
: Width (in pixels) of the window's border.

XmNheight
: Window height (in pixels), excluding the border.

XmNsensitive
: Tells whether a widget receives input (is sensitive). The XtSetSensitive() routine can be used to change a widget's sensitivity and to guarantee that if a parent has its XmNsensitive resource set to False, then its children will have their ancestor-sensitive flag set correctly.

XmNwidth
: Window width (in pixels), excluding the border.

XmNx
: The x-coordinate of the widget's upper-left outer corner, relative to the upper-left inner corner of its parent.

XmNy
: The y-coordinate of the widget's upper-left outer corner (that is, outside of any border or shadow rectangle), relative to its parents upper-left inner corner (inside all border or shadow rectangles).

Inherited Resources

RectObj inherits the following resource:

Resource	Inherited From
XmNdestroyCallback	Core

See Also

`Object`(2).

Name

SessionShell widget class – the main shell for an application.

Synopsis

Public Headers:
<Xm/Xm.h>
<X11/Shell.h>

Class Name:
SessionShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → TopLevelShell → ApplicationShell → SessionShell

Class Pointer:
`sessionShellWidgetClass`

Instantiation:
```
widget = XtAppInitialize (...)
```
or
```
widget = XtAppCreateShell (app_name, app_class,
                           sessionShellWidget-
                           Class,...)
```

Functions/Macros:
`XtAppCreateShell()`, `XtVaAppCreateShell()`, `XtIsSession-Shell()`

Availability

The Session shell is only available from X11R6.

Description

A SessionShell is the normal top-level window for an application. It does not have a parent and it is at the root of the widget tree. An application should have only one SessionShell, unless the application is implemented as multiple logical applications. Normally, an application will use TopLevelShell widgets for other top-level windows.

The SessionShell differs from the ApplicationShell in that it interfaces to the X11R6 Session Management facilities, which enable applications to save or restart themselves in a known state in response to commands from the desktop

A SessionShell is returned by the call to `XtVaAppInitialize()`. It can also be created explicitly with a call to `XtVaAppCreateShell()`.

Interaction with the user during session management is implemented through a token-passing mechanism. A *Checkpoint* token is passed between the Session

Manager and the Session Shell callbacks; the application may interact with the user directly (usually to ask the user if unsaved changes to the application should be saved) only if the SessionShell of the application holds the token. The application may only interact with the user after issuing a request to do so. Issuing a request takes the form of registering a procedure on the XtNinteractCallback list. If and when the Session Manager decides that it is time to allow user interaction, the interact procedures are invoked, with the *checkpoint* token passed to the application through *call_data* for the procedures so registered. After the interaction with the user is complete, the application returns the *checkpoint* by calling `XtSessionReturnToken`(). Unusually, callbacks on the XtNinteractCallback list are invoked one at a time; after the first procedure is called, it is removed from the list.

New Resources

SessionShell defines the following resources:

Name	Class	Type	Default	Access
XtNcancelCallback	XtCCallback	XtCallbackList	NULL	C
XtNcloneCommand	XtCCloneCommand	String *	dynamic	CSG
XtNconnection	XtCConnection	SmcConn	NULL	CSG
XtNcurrentDirectory	XtCCurrentDirectory	String	NULL	CSG
XtNdieCallback	XtCCallback	XtCallbackList	NULL	C
XtNdiscardCommand	XtCDiscardCommand	String *	NULL	CSG
XtNenvironment	XtCEnvironment	String *	NULL	CSG
XtNerrorCallback	XtCCallback	XtCallbackList	NULL	C
XtNinteractCallback	XtCCallback	XtCallbackList	NULL	C
XtNjoinSession	XtCJoinSession	Boolean	True	CSG
XtNprogramPath	XtCProgramPath	String	dynamic	CSG
XtNresignCommand	XtCResignCommand	String *	NULL	CSG
XtNrestartCommand	XtCRestartCommand	String *	dynamic	CSG
XtNrestartStyle	XtCRestartStyle	unsigned char	SmRestartIfRunning	CSG
XtNsaveCallback	XtCCallback	XtCallbackList	NULL	C
XtNsaveCompleteCallback	XtCCallback	XtCallbackList	NULL	C
XtNsessionID	XtCSessionID	String	NULL	CSG
XtNshutdownCommand	XtCShutdownCommand	String *	NULL	CSG

`XtNcancelCallback`
List of callbacks to be invoked when the SessionShell receives a ShutdownCancelled message from the Session Manager.

XtNcloneCommand
: Specifies a command which the Session Manager uses to create a new instance of the application. If the value is NULL, the Session Manager will use the value of the XtNrestartCommand resource.

XtNconnection
: Specifies the connection between the SessionShell and the Session Manager. Normally the SessionShell instantiates this value when the shell is created, although the programmer can specify a value if the application has already established a private connection.

XtNcurrentDirectory
: Specifies a location in the file system where the Session Manager should arrange to restart the application when required to do so.

XtNdieCallback
: List of callbacks invoked when the application receives a Die message from the Session Manager. The application should take whatever steps are required to cleanly terminate.

XtNdiscardCommand
: Specifies a command which, if invoked, will cause the host to discard all information pertaining to the current application state. If NULL, the Session Manager assumes that the application state is fully recoverable from the XtNrestartCommand specification.

XtNenvironment
: Specifies the environment variables (and values) which the Session Manager should set up prior to restarting the application. The resource is assumed to consist of a list of “name=value” strings.

XtNerrorCallback
: List of callbacks to be invoked if the connection between the Session Manager and the SessionShell becomes irrevocably lost. The XtNconnection is reset to NULL by the SessionShell in these circumstances.

XtNinteractCallback
: List of callbacks invoked when a client wants to interact with the user before a session shutdown. This callback list is implemented in a special manner: each time the Session Manager is issued a request to interact with the user, the Session Manager calls *and then removes* the top callback from the list. Furthermore, the request to interact with the user during Session Management operations is performed *simply by registering a callback on this list*. If there is more than one callback on the list, subsequent
callbacks below the top are not called until the application calls `XtSessionReturnToken`(), returning the checkpoint token to the Session Manager.

XtNjoinSession
: Specifies whether the SessionShell should automatically initialize a connection to the Session Manager. Setting the resource True at any time will initialize the connection; subsequently setting it False will resign from the session.

XtNprogramPath
: The full path of the program which is running. If NULL, the session management uses the first element of the XtNrestartCommand array as the program path.

XtNresignCommand
: Specifies a command which logically undoes the client: saved state should be removed.

XtNrestartCommand
: Specifies a command which should cause an instance of the application to be invoked, such that it restarts in its current state. If NULL, the XmNargv resource is used as fallback.

XtNrestartStyle
: Specifies a hint to the session manager, indicating how the application would like to be restarted. Possible values are:

  SmRestartIfRunning
  SmRestartAnyway
  SmRestartImmediately
  SmRestartNever

  SmRestartIfRunning is the default, and specifies that the client should restart if it was running at the end of the current session.

  SmRestartAnyway specifies that the client should be restarted even if it terminated before the end of the current session.

  SmRestartImmediately specifies that the client is meant to run continuously. If it exits at any time, the session manager should restart it in the current session.

  SmRestartNever specifies that the client should not be restarted under session management control.

XtNsaveCallback
: Specifies a list of callbacks to be invoked when the client receives a SaveYourself message from the session manager. The procedures are responsible for saving the application state.

XtNsaveCompleteCallback
: Specifies a list of callbacks to be invoked when the session manager sends a SaveComplete message to the client. Clients can continue their normal operations thereafter.

XtNsessionID
This resource identifies the client to the session manager. This is either assigned by the session manager when connection is established, or deduced from any -xtsessionID command line argument to the application.

XtNshutdownCommand
Specifies a command which the session manager will invoke at shutdown; the command should clean up after the client, but not remove any saved state.

Callback Structure

Callbacks on the XtNsaveCallback and XtNinteractCallback lists are each passed the following structure as call_data when invoked:

```
typedef struct _XtCheckpointTokenRec {
    int             save_type;
    int             interact_style;
    Boolean         shutdown;
    Boolean         fast;
    Boolean         cancel_shutdown;
    int             phase;
    int             interact_dialog_type;
    Boolean         request_cancel;
    Boolean         request_next_phase;
    Boolean         save_success;
    int             type;
    Widget          widget;
} XtCheckpointTokenRec, *XtCheckpointToken;
```

The *save_type* element indicates the type of information which the application should attempt to save. Possible values are: SmSaveLocal, SmSaveGlobal, SmSaveBoth.

The *interact_style* element indicates the kind of user interaction which is currently permitted. Possible values are: SmInteractStyleNone, SmInteractStyleErrors, SmInteractStyleAny.

The *shutdown* element indicates whether the save interaction is being performed prior to a session shutdown.

If *fast* is True, the client should endeavour to save the minimum recovery state possible.

If *cancel_shutdown* is True, the Session Manager has sent a ShutdownCancelled message to the client.

The *phase* element is for specialized manager clients use only (the window manager), and indicates the state of the interaction between the Session Manager and the client. The value will be either 1 or 2.

The remaining fields are where the client communicates back to the Session Manager.

The *interact_dialog_type* element specifies the kind of interaction required by the client. The initial value is SmDialogNormal, which is for a normal interactive dialog. The value of SmDialogError requests an error dialog interaction.

The *request_cancel* element is only used by XtNinteractCallbacks, and is a hint to the session manager that the client requests that the current shutdown operation should be cancelled.

The *request_next_phase* element is used by the specialized manager clients: the default value is False, but can be set True by these clients.

The *save_success* element is where the client indicates to the Session Manager the status of the application save-state operations. The value False indicates that the client could not save its state successfully.

The *type* and *widget* fields are internal to the implementation, and are not to be used by application programmers.

Inherited Resources

SessionShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of `XmNborderWidth` is reset to 0 by VendorShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNlabelRenderTable	VendorShell
XmNallowShellResize	Shell	XmNlayoutDirection	VendorShell
XmNancestorSensitive	Core	XmNmappedWhenManaged	Core
XmNargc	ApplicationShell	XmNmaxAspectX	WMShell
XmNargv	ApplicationShell	XmNmaxAspectY	WMShell
XmNaudibleWarning	VendorShell	XmNmaxHeight	WMShell
XmNbackground	Core	XmNmaxWIdth	WMShell
XmNbackgroundPixmap	Core	XmNminAspectX	WMShell
XmNbaseHeight	WMShell	XmNminAspectY	WMShell
XmNbaseWidth	WMShell	XmNminHeight	WMShell
XmNborderColor	Core	XmNminWidth	WMShell
XmNborderPixmap	Core	XmNmwmDecorations	VendorShell

Resource	Inherited From	Resource	Inherited From
XmNborderWidth	Core	XmNmwmMenu	VendorShell
XmNbuttonFontList	VendorShell	XmNnumChildren	Composite
XmNbuttonRenderTable	VendorShell	XmNoverrideRedirect	Shell
XmNchildren	Composite	XmNpopdownCalback	Shell
XmNcolormap	Core	XmNpopupCallback	Shell
XmNcreatePopupChildProc	Shell	XmNpreeditType	VendorShell
XmNdefaultFontList	VendorShell	XmNsaveUnder	Shell
XmNdeleteResponse	VendorShell	XmNscreen	Core
XmNdepth	Core	XmNsensitive	Core
XmNdestroyCallback	Core	XmNshellUnitType	VendorShell
XmNgeometry	Shell	XmNtextFontList	VendorShell
XmNheight	Core	XmNtextRenderTable	VendorShell
XmNheightInc	WMShell	XmNtitle	WMShell
XmNiconic	TopLevelShell	XmNtitleEncoding	WMShell
XmNiconMask	WMShell	XmNtoolTipEnable	VendorShell
XmNiconName	TopLevelShell	XmNtoolTipPostDuration	VendorShell
XmNiconNameEncoding	TopLevelShell	XmNtoolTipString	VendorShell
XmNiconPixmap	WMShell	XmNtransient	WMShell
XmNiconWindow	WMShell	XmNtranslations	Core
XmNinitialResourcesPersistent	Core	XmNvisual	Shell
XmNinitialState	WMShell	XmNwaitForWm	WMShell
XmNinput	WMShell	XmNwidth	Core
XmNinputMethod	VendorShell	XmNwidthInc	WMShell
XmNinputPolicy	VendorShell	XmNwindowGroup	WMShell
XmNinsertPosition	Composite	XmNwinGravity	WMShell
XmNkeyboardFocusPolicy	VendorShell	XmNwmTimeout	WMShell
XmNlabelFontList	VendorShell	XmNx	Core
XmNmwmFunctions	VendorShell	XmNy	Core
XmNmwmInputMode	VendorShell		

The VendorShell superclass installs a handler which intercepts the window manager WM_DELETE_WINDOW message. The handler is inherited by sub-classes of VendorShell, and has the behavior that if XmNdeleteResponse is XmDESTROY, and the widget is an instance of an ApplicationShell, then the application context associated with the widget is destroyed, followed by a call to exit().

See Also

ApplicationShell(2), Composite(2), Core(2), Shell(2), TopLevelShell(2), VendorShell(2), WMShell(2).

Name

Shell widget class – fundamental widget class that controls interaction between top-level windows and the window manager.

Synopsis

Public Header:
<Xm/Xm.h>
<X11/Shell.h>

Class Name:
Shell

Class Hierarchy:
Core → Composite → Shell

Class Pointer:
shellWidgetClass

Instantiation:
Shell is an Intrinsics meta-class and is not normally instantiated.

Functions/Macros:
`XtIsShell`()

Description

Shell is a subclass of Composite that handles interaction between the window manager and its single child.

New Resources

Shell defines the following resources:

Name	Class	Type	Default	Access
XmNallowShellResize	XmCAllowShellResize	Boolean	False	CSG
XmNcreatePopupChildProc	XmCCreatePopupChildProc	XtCreatePopupChildProc	NULL	CSG
XmNgeometry	XmCGeometry	String	NULL	CSG
XmNoverrideRedirect	XmCOverrideRedirect	Boolean	False	CSG
XmNpopdownCallback	XmCCallback	XtCallbackList	NULL	C
XmNpopupCallback	XmCCallback	XtCallbackList	NULL	C
XmNsaveUnder	XmCSaveUnder	Boolean	False	CSG
XmNvisual	XmCVisual	Visual *	CopyFromParent	CSG

XmNallowShellResize
If False (default), the Shell widget refuses geometry requests from its children (by returning XtGeometryNo).

XmNcreatePopupChildProc
A pointer to an XtCreatePopupChildProc procedure that creates a child widget--but only when the shell is popped up, not when the application is started. This is useful in menus, for example, since you don't need to create the menu until it is popped up. This procedure is called after any callbacks specified in the XmNpopupCallback resource.

XmNgeometry
This resource specifies the values for the resources XmNx, XmNy, XmNwidth, and
XmNheight in situations where an unrealized widget has added or removed some of its managed children.

XmNoverrideRedirect
If True, the widget is considered a temporary window that redirects the keyboard focus away from the main application windows. Usually this resource shouldn't be changed.

XmNpopdownCallback
List of callbacks that are called when the widget is popped down using `XtPopdown`().

XmNpopupCallback
List of callbacks that are called when the widget is popped up using `XtPopup`().

XmNsaveUnder
If True, screen contents that are obscured by a widget are saved, thereby avoiding the overhead of sending expose events after the widget is unmapped.

XmNvisual
The visual server resource that is used when creating the widget.

Procedures

XtCreatePopupChildProc
An XtCreatePopupChildProc is a pointer to a procedure, specified as follows:

```
typedef void (*XtCreatePopupChildProc) (Widget);
```

An XtCreatePopupChildProc procedure is called when a Shell or derived class is popped up, typically through a call to `XtPopup`(). The function has a single parameter, which is the widget ID of the shell.

Inherited Resources

Shell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them.

Name	Inherited From	Name	Inherited From
XmNaccelerators	Core	XmNheight	Core
XmNancestorSensitive	Core	XmNinsertPosition	Composite
XmNbackground	Core	XmNinitialResourcesPersistent	Core
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNscreen	Core
XmNborderWidth	Core	XmNsensitive	Core
XmNchildren	Composite	XmNtranslations	Core
XmNcolormap	Core	XmNwidth	Core
XmNdepth	Core	XmNx	Core
XmNdestroyCallback	Core	XmNy	Core

See Also

`Composite`(2), `Core`(2).

Name

TopLevelShell widget class – additional top-level shells for an application.

Synopsis

Public Header:
<Xm/Xm.h>
<X11/Shell.h>

Class Name:
TopLevelShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → TopLevelShell

Class Pointer:
topLevelShellWidgetClass

Instantiation:
widget = `XtCreatePopupShell` (name, topLevelShellWidgetClass,...)

Functions/Macros:
`XtIsTopLevelShell`()

Description

TopLevelShell is a subclass of VendorShell that is used for additional shells in applications having more than one top-level window.

New Resources

TopLevelShell defines the following resources:

Name	Class	Type	Default	Access
XmNiconic	XmCIconic	Boolean	False	CSG
XmNiconName	XmCIconName	String	NULL	CSG
XmNiconNameEncoding	XmCIconNameEncoding	Atom	dynamic	CSG

XmNiconic
If True, the widget is realized as an icon, otherwise as a normal window. XmNiconic overrides the value of the inherited XmNinitialState resource.

XmNiconName
The abbreviated name that labels an iconified application.

XmNiconNameEncoding
The property type for encoding the XmNiconName resource.

Inherited Resources

TopLevelShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. TopLevelShell resets XmNinput to True.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNmaxAspectY	WMShell
XmNallowShellResize	Shell	XmNmaxHeight	WMShell
XmNancestorSensitive	Core	XmNmaxWIdth	WMShell
XmNaudibleWarning	VendorShell	XmNminAspectX	WMShell
XmNbackground	Core	XmNminAspectY	WMShell
XmNbackgroundPixmap	Core	XmNminHeight	WMShell
XmNbaseHeight	WMShell	XmNminWidth	WMShell
XmNbaseWidth	WMShell	XmNmwmDecorations	VendorShell
XmNborderColor	Core	XmNmwmFunctions	VendorShell
XmNborderPixmap	Core	XmNmwmInputMode	VendorShell
XmNborderWidth	Core	XmNmwmMenu	VendorShell
XmNbuttonFontList	VendorShell	XmNnumChildren	Composite
XmNbuttonRenderTable	VendorShell	XmNoverrideRedirect	Shell
XmNchildren	Composite	XmNpopdownCalback	Shell
XmNcolormap	Core	XmNpopupCallback	Shell
XmNcreatePopupChildProc	Shell	XmNpreeditType	VendorShell
XmNdefaultFontList	VendorShell	XmNsaveUnder	Shell
XmNdeleteResponse	VendorShell	XmNscreen	Core
XmNdepth	Core	XmNsensitive	Core
XmNdestroyCallback	Core	XmNshellUnitType	VendorShell
XmNgeometry	Shell	XmNtextFontList	VendorShell
XmNheight	Core	XmNtextRenderTable	VendorShell
XmNheightInc	WMShell	XmNtitle	WMShell
XmNiconMask	WMShell	XmNtitleEncoding	WMShell
XmNiconPixmap	WMShell	XmNtoolTipEnable	VendorShell
XmNiconWindow	WMShell	XmNtoolTipPostDuration	VendorShell
XmNinitialResourcesPersistent	Core	XmNtoolTipString	VendorShell
XmNinitialState	WMShell	XmNtransient	WMShell
XmNinput	WMShell	XmNtranslations	Core

Resource	Inherited From	Resource	Inherited From
XmNinputMethod	VendorShell	XmNvisual	Shell
XmNinputPolicy	VendorShell	XmNwaitForWm	WMShell
XmNinsertPosition	Composite	XmNwidth	Core
XmNkeyboardFocusPolicy	VendorShell	XmNwidthInc	WMShell
XmNlabelFontList	VendorShell	XmNwindowGroup	WMShell
XmNlabelRenderTable	VendorShell	XmNwinGravity	WMShell
XmNlayoutDirection	VendorShell	XmNwmTimeout	WMShell
XmNmappedWhenManaged	Core	XmNx	Core
XmNmaxAspectX	WMShell	XmNy	Core

See Also

`Composite`(2), `Core`(2). `Shell`(2), `VendorShell`(2), `WMShell`(2).

Name

TransientShell widget class – popup shell that interacts with the window manager.

Synopsis

Public Header:
<Xm/Xm.h>
<X11/Shell.h>

Class Name:
TransientShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → TransientShell

Class Pointer:
transientShellWidgetClass

Instantiation:

Functions/Macros:
`XtIsTransientShell`()

Description

TransientShell is a subclass of VendorShell that is used for popup shell widgets, such as dialog boxes, that interact with the window manager. Most window managers will not allow the user to iconify a TransientShell window on its own and may iconify it automatically if the window that it is transient for is iconified.

New Resources

TransientShell defines the following resources:

Name	Class	Type	Default	Access
XmNtransientFor	XmCTransientFor	Widget	NULL	CSG

XmNtransientFor
The widget from which the TransientShell will pop up. If the value of this resource is NULL or identifies an unrealized widget, then TransientShell uses the value of the WMShell resource XmNwindowGroup.

Inherited Resources

TransientShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. TransientShell resets the resources XmNinput, XmNtransient, and XmNsaveUnder to True.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNmaxAspectY	WMShell
XmNallowShellResize	Shell	XmNmaxHeight	WMShell
XmNancestorSensitive	Core	XmNmaxWIdth	WMShell
XmNaudibleWarning	VendorShell	XmNminAspectX	WMShell
XmNbackground	Core	XmNminAspectY	WMShell
XmNbackgroundPixmap	Core	XmNminHeight	WMShell
XmNbaseHeight	WMShell	XmNminWidth	WMShell
XmNbaseWidth	WMShell	XmNmwmDecorations	VendorShell
XmNborderColor	Core	XmNmwmFunctions	VendorShell
XmNborderPixmap	Core	XmNmwmInputMode	VendorShell
XmNborderWidth	Core	XmNmwmMenu	VendorShell
XmNbuttonFontList	VendorShell	XmNnumChildren	Composite
XmNbuttonRenderTable	VendorShell	XmNoverrideRedirect	Shell
XmNchildren	Composite	XmNpopdownCalback	Shell
XmNcolormap	Core	XmNpopupCallback	Shell
XmNcreatePopupChildProc	Shell	XmNpreeditType	VendorShell
XmNdefaultFontList	VendorShell	XmNsaveUnder	Shell
XmNdeleteResponse	VendorShell	XmNscreen	Core
XmNdepth	Core	XmNsensitive	Core
XmNdestroyCallback	Core	XmNshellUnitType	VendorShell
XmNgeometry	Shell	XmNtextFontList	VendorShell
XmNheight	Core	XmNtextRenderTable	VendorShell
XmNheightInc	WMShell	XmNtitle	WMShell
XmNiconMask	WMShell	XmNtitleEncoding	WMShell
XmNiconPixmap	WMShell	XmNtoolTipEnable	VendorShell
XmNiconWindow	WMShell	XmNtoolTipPostDuration	VendorShell
XmNinitialResourcesPersistent	Core	XmNtoolTipString	VendorShell
XmNinitialState	WMShell	XmNtransient	WMShell
XmNinput	WMShell	XmNtranslations	Core

Resource	Inherited From	Resource	Inherited From
XmNinputMethod	VendorShell	XmNvisual	Shell
XmNinputPolicy	VendorShell	XmNwaitForWm	WMShell
XmNinsertPosition	Composite	XmNwidth	Core
XmNkeyboardFocusPolicy	VendorShell	XmNwidthInc	WMShell
XmNlabelFontList	VendorShell	XmNwindowGroup	WMShell
XmNlabelRenderTable	VendorShell	XmNwinGravity	WMShell
XmNlayoutDirection	VendorShell	XmNwmTimeout	WMShell
XmNmappedWhenManaged	Core	XmNx	Core
XmNmaxAspectX	WMShell	XmNy	Core

See Also

`Composite`(2), `Core`(2), `Shell`(2), `VendorShell`(2), `WMShell`(2).

Name

VendorShell widget class – shell widget with Motif-specific hooks for window manager interaction.

Synopsis

Public Header:
<Xm/VendorS.h>
<X11/Shell.h>

Class Name:
VendorShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell

Class Pointer:
vendorShellWidgetClass

Instantiation:
VendorShell is a meta-class and is not normally instantiated.

Functions/Macros:
`XmIsVendorShell`()

Description

VendorShell is a vendor-specific supporting superclass for all shell classes that are visible to the window manager and that do not have override redirection. VendorShell defines resources that provide the Motif look-and-feel and manages the specific communication needed by the Motif Window Manager (*mwm*).

Traits

VendorShell holds the `XmQTspecifyRenderTable`, `XmQTspecifyLayoutDirection`, `XmQTaccessColors` and `XmQTspecifyUnitType` traits, which are inherited by any derived classes, and uses the `XmQTspecifyRenderTable` trait.

New Resources

VendorShell defines the following resources:

Name	Class	Type	Default	Access
XmNaudibleWarning	XmCAudibleWarning	unsigned char	XmBELL	CSG
XmNbuttonFontList	XmCButtonFontList	XmFontList	dynamic	CSG
XmNbuttonRenderTable	XmCButtonRenderTable	XmRenderTable	dynamic	CSG
XmNdefaultFontList	XmCDefaultFontList	XmFontList	dynamic	CG
XmNdeleteResponse	XmCDeleteResponse	unsigned char	XmDESTROY	CSG
XmNinputMethod	XmCInputMethod	String	NULL	CSG

Name	Class	Type	Default	Access
XmNinputPolicy	XmCInputPolicy	XmInputPolicy	XmPER_SHELL	CSG
XmNkeyboardFocusPolicy	XmCKeyboardFocusPolicy	unsigned char	XmEXPLICIT	CSG
XmNlabelFontList	XmCLabelFontList	XmFontList	dynamic	CG
XmNlabelRenderTable	XmCLabelRenderTable	XmRenderTable	dynamic	CSG
XmNlayoutDirection	XmCLayoutDirection	XmDirection	XmLEFT_TO_RIGHT	CG
XmNmwmDecorations	XmCMwMDecorations	int	-1	CSG
XmNmwmFunctions	XmCMwMFunctions	int	-1	CSG
XmNmwmInputMode	XmCMwMInputMode	int	-1	CSG
XmNmwmMenu	XmCMwmMenu	String	NULL	
XmNpreeditType	XmCPreeditType	String	dynamic	CSG
XmNshellUnitType	XmCShellUnitType	unsigned char	XmPIXELS	CSG
XmNtextFontList	XmCTextFontList	XmFontList	dynamic	CG
XmNtextRenderTable	XmCTextRenderTable	XmRenderTable	dynamic	CSG
XmNtoolTipEnable	XmCToolTipEnable	Boolean	False	CSG
XmNtoolTipPostDelay	XmCToolTipPostDelay	int	5000	CSG
XmNtoolTipPostDuration	XmCToolTipPostDuration	int	5000	CSG
XmNuseAsyncGeometry	XmCUseAsyncGeometry	Boolean	False	CSG
XmNunitType	XmCUnitType	unsigned char	XmPIXELS	CSG

XmNaudibleWarning
Specifies whether an action performs an associated audible cue. Possible values:

```
XmBELL      /* rings the bell  */
XmNONE      /* does nothing  */
```

XmNbuttonFontList
Specifies the font list used for the button descendants of the VendorShell widget. In Motif 2.0 and later, the XmFontList is considered obsolete, and is replaced by the XmRenderTable. The XmNbuttonRenderTable resource is the preferred method of specifying appearance.

XmNbuttonRenderTable
In Motif 2.0 and later, specifies the render table to be used by VendorShell's button descendants. If initially NULL, the value is taken from any specified XmNdefaultFontList value for backwards compatibility. If this is also NULL, the nearest ancestor which has the XmQTspecifyRenderTable trait is sought, taking the XmBUTTON_RENDER_TABLE value from any widget so found.

XmNdefaultFontList
The default font list for the children of the VendorShell widget. The resource is obsolete, replaced by the XmNbuttonFontList, XmNlabelFontList, and XmNtextFontList resources, which are in their turn also obsolete.

XmNdeleteResponse
The action to perform when the shell receives a WM_DELETE_WINDOW message. Possible values:

```
XmDESTROY       /* destroy window      */
XmUNMAP         /* unmap window        */
XmDO_NOTHING    /* leave window as is  */
```

XmNinputMethod
Specifies the string that sets the locale modifier for the input method.

XmNinputPolicy
In Motif 2.0 and later, specifies the policy to adopt when creating an Input Context. Possible values:

```
XmPER_SHELL     /* one input context per shell hierarchy */
XmPER_WIDGET    /* one input context per widget          */
```

XmNkeyboardFocusPolicy
The method of assigning keyboard focus. Possible values:

```
XmEXPLICIT      /* click-to-type policy    */
XmPOINTER       /* pointer-driven policy   */
```

XmNlabelFontList
Specifies the font list used for the label descendants of the VendorShell widget. In Motif 2.0 and later, the XmFontList is considered obsolete, and is replaced by the XmRenderTable. The XmNlabelRenderTable resource is the preferred method of specifying appearance.

XmNlabelRenderTable
In Motif 2.0 and later, specifies the render table to be used by VendorShell's label descendants. If initially NULL, the value is taken from any specified XmNdefaultFontList value for backwards compatibility. If this is also NULL, the nearest ancestor which has the XmQTspecifyRenderTable trait is sought, taking the XmLABEL_RENDER_TABLE value from any widget so found.

XmNlayoutDirection
In Motif 2.0 and later, specifies the default direction in which visual components are to be laid out. Descendants of VendorShell use this value in the absence of an explicit layout direction further down the widget hierarchy. Possible values:

```
XmLEFT_TO_RIGHT
XmRIGHT_TO_LEFT
```

XmBOTTOM_TO_TOP
XmTOP_TO_BOTTOM
XmBOTTOM_TO_TOP_LEFT_TO_RIGHT
XmBOTTOM_TO_TOP_RIGHT_TO_LEFT
XmTOP_TO_BOTTOM_LEFT_TO_RIGHT
XmTOP_TO_BOTTOM_RIGHT_TO_LEFT
XmLEFT_TO_RIGHT_BOTTOM_TO_TOP
XmRIGHT_TO_LEFT_BOTTOM_TO_TOP
XmLEFT_TO_RIGHT_TOP_TO_BOTTOM
XmRIGHT_TO_LEFT_TOP_TO_BOTTOM

XmNmwmDecorations

This resource corresponds to the values assigned by the decorations field of the _MOTIF_WM_HINTS property. This resource determines which frame buttons and handles to include with a window. The value for the resource is a bitwise inclusive OR of one or more of the following, which are defined in *<Xm/MwmUtil.h>*:

```
MWM_DECOR_ALL          /* remove decorations from full set */
MWM_DECOR_BORDER       /* window border                    */
MWM_DECOR_RESIZEH      /* resize handles                   */
MWM_DECOR_TITLE        /* title bar                        */
MWM_DECOR_MENU         /* window's menu button             */
MWM_DECOR_MINIMIZE     /* minimize button                  */
MWM_DECOR_MAXIMIZE     /* maximize button                  */
```

XmNmwmFunctions

This resource corresponds to the values assigned by the functions field of the _MOTIF_WM_HINTS property. This resource determines which functions to include in the system menu. The value for the resource is a bitwise inclusive OR of one or more of the following, which are defined in the header file *<Xm/MwmUtil.h>*.

```
MWM_FUNC_ALL           /* remove functions from full set */
MWM_FUNC_RESIZE        /* f.resize                       */
MWM_FUNC_MOVE          /* f.move                         */
MWM_FUNC_MINIMIZE      /* f.minimize                     */
MWM_FUNC_MAXIMIZE      /* f.maximize                     */
MWM_FUNC_CLOSE         /* f.kill                         */
```

XmNmwmInputMode

This resource corresponds to the values assigned by the input_mode field of the _MOTIF_WM_HINTS property. This resource determines the constraints on the window's keyboard focus. That is, it determines whether the application takes the

keyboard focus away from the primary window or not. The possible values are as follows, defined in <*Xm/MwmUtil.h*>:

MWM_INPUT_MODELESS[1]
MWM_INPUT_PRIMARY_APPLICATION_MODAL
MWM_INPUT_SYSTEM_MODAL
MWM_INPUT_FULL_APPLICATION_MODAL

If the value is MWM_INPUT_MODELESS, input can be directed to any window. If the value is MWM_INPUT_PRIMARY_APPLICATION_MODAL, input can not be directed at an ancestor of the window. The value MWM_INPUT_SYSTEM_MODAL indicates that input only goes to the current window. MWM_INPUT_FULL_APPLICATION_MODAL specifies that input may not be directed at any other window of the application.

XmNmwmMenu
The menu items to add at the bottom of the client's window menu. The string has this format:

label [mnemonic] [accelerator] mwm_f.function

XmNpreeditType
Specifies the input method style(s) that are available. The resource value is a comma separated list of the following values:

```
OffTheSpot      /* XIMPreeditArea      */
Root            /* XIMPreeditNothing   */
None            /* XIMPreeditNone      */
OverTheSpot     /* XIMPreeditPosition  */
OnTheSpot       /* XIMPreeditCallbacks */
```

XmNshellUnitType
The measurement units to use in resources that specify a size or position. In Motif 2.0 and later, the resource is obsolete, being replaced by the XmNunitType resource.

XmNtextFontList
Specifies the font list used for the text descendants of the VendorShell widget. In Motif 2.0 and later, the XmFontList is considered obsolete, and is replaced by the XmRenderTable. The XmNtextRenderTable resource is the preferred method of specifying appearance.

XmNtextRenderTable
In Motif 2.0 and later, specifies the render table to be used by VendorShell's text and list descendants. If initially NULL, the value is taken from any specified

1.Erroneously given as MWM_INPUT_MODELES in 2nd edition.

XmNdefaultFontList value for backwards compatibility. If this is also NULL, the nearest ancestor which has the XmQTspecifyRenderTable trait is sought, taking the XmTEXT_RENDER_TABLE value from any widget so found.

XmNtoolTipEnable
Specifies whether toolTips are enabled or not for this shell.

XmNtoolTipPostDelay
Specifies the time, in milliseconds, to wait after the pointer enters a widget before posting the toolTip associated with this widget.

XmNtoolTipPostDuration
Specifies the time, in milliseconds, that the toolTip is displayed. A value of 0 will display the tip indefinitely.

XmNuseAsyncGeometry
If True, the geometry manager doesn't wait to confirm a geometry request that was sent to the window manager. The geometry manager performs this by setting the WMShell resource XmNwaitForWm to False and by setting the WMShell resource XmNwmTimeout to 0.

If XmNuseAsyncGeometry is False (default), the geometry manager uses synchronous notification, and so it doesn't change the resources XmNwaitForWm and XmNwmTimeout.

XmNunitType
In Motif 2.0 and later, specifies the units in which size and position resources are calculated. The resource replaces XmNshellUnitType, which is considered obsolete. The values XmFONT_UNITS and Xm100TH_FONT_UNITS have a horizontal and vertical component, calculated from the values of the XmNhorizontalFontUnit and XmNverticalFontUnit resources of the XmScreen object. Possible values:

```
XmPIXELS              /* pixels                          */
XmMILLIMETERS         /* millimeters                     */
Xm100TH_MILLIMETERS   /* 1/100 of a millimeter           */
XmCENTIMETERS         /* centimeters                     */
XmINCHES              /* inches                          */
Xm1000TH_INCHES       /* 1/1000 of an inch               */
XmPOINTS              /* point units (1/72 of an inch)   */
Xm100TH_POINTS        /* 1/100 of a point                */
XmFONT_UNITS          /* depends on XmScreen resources   */
Xm100TH_FONT_UNITS    /* 1/100 of the above              */
```

Inherited Resources

VendorShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. VendorShell resets XmNborderWidth from 1 to 0 and resets XmNinput to True.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNmaxAspectY	WMShell
XmNallowShellResize	Shell	XmNmaxHeight	WMShell
XmNancestorSensitive	Core	XmNmaxWIdth	WMShell
XmNbackground	Core	XmNminAspectX	WMShell
XmNbackgroundPixmap	Core	XmNminAspectY	WMShell
XmNbaseHeight	WMShell	XmNminHeight	WMShell
XmNbaseWidth	WMShell	XmNminWidth	WMShell
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNoverrideRedirect	Shell
XmNborderWidth	Core	XmNpopdownCalback	Shell
XmNchildren	Composite	XmNpopupCallback	Shell
XmNcolormap	Core	XmNsaveUnder	Shell
XmNcreatePopupChildProc	Shell	XmNscreen	Core
XmNdepth	Core	XmNsensitive	Core
XmNdestroyCallback	Core	XmNtitle	WMShell
XmNgeometry	Shell	XmNtitleEncoding	WMShell
XmNheight	Core	XmNtoolTipEnable	VendorShell
XmNheightInc	WMShell	XmNtoolTipPostDuration	VendorShell
XmNiconic	TopLevelShell	XmNtoolTipString	VendorShell
XmNiconMask	WMShell	XmNtransient	WMShell
XmNiconName	TopLevelShell	XmNtranslations	Core
XmNiconNameEncoding	TopLevelShell	XmNvisual	Shell
XmNiconPixmap	WMShell	XmNwaitForWm	WMShell
XmNiconWindow	WMShell	XmNwidth	Core
XmNinitialResourcesPersistent	Core	XmNwidthInc	WMShell
XmNinitialState	WMShell	XmNwindowGroup	WMShell
XmNinput	WMShell	XmNwinGravity	WMShell
XmNinsertPosition	Composite	XmNwmTimeout	WMShell
XmNmappedWhenManaged	Core	XmNx	Core

Resource	Inherited From	Resource	Inherited From
XmNmaxAspectX	WMShell	XmNy	Core

See Also

`Composite`(2), `Core`(2), `Shell`(2), `WMShell`(2).

Name

WMShell widget class – fundamental shell widget that interacts with an ICCCM-compliant window manager.

Synopsis

Public Header:
<Xm/Xm.h>
<X11/Shell.h>

Class Name:
WMShell

Class Hierarchy:
Core → Composite → Shell → WMShell

Class Pointer:
wmShellWidgetClass

Instantiation:
WMShell is an Intrinsics meta-class and is not normally instantiated.

Functions/Macros:
`XtIsWMShell`()

Description

WMShell is a direct subclass of Shell that provides basic window manager interaction. WMShell is not directly instantiated; it encapsulates the application resources that applications use to communicate with window managers.

New Resources

WMShell defines the following resources:

Name	Class	Type	Default	Access
XmNbaseHeight	XmCBaseHeight	int	XtUnspecifiedShellInt	CSG
XmNbaseWidth	XmCBaseWidth	int	XtUnspecifiedShellInt	CSG
XmNheightInc	XmCHeightInc	int	XtUnspecifiedShellInt	CSG
XmNiconMask	XmCIconMask	Pixmap	NULL	CSG
XmNiconPixmap	XmCIconPixmap	Pixmap	NULL	CSG
XmNiconWindow	XmCIconWindow	Window	NULL	CSG
XmNinitialState	XmCInitialState	int	NormalState	CSG
XmNiconX	XmCIconY	int	-1	CSG
XmNiconY	XmCIconY	int	-1	CSG
XmNinput	XmCInput	Boolean	False	CSG
XmNmaxAspectX	XmCMaxAspectX	int	XtUnspecifiedShellInt	CSG

Name	Class	Type	Default	Access
XmNmaxAspectY	XmCMaxAspectY	int	XtUnspecifiedShellInt	CSG
XmNmaxHeight	XmCMaxHeight	int	XtUnspecifiedShellInt	CSG
XmNmaxWIdth	XmCMaxWidth	int	XtUnspecifiedShellInt	CSG
XmNminAspectX	XmCMinAspectX	int	XtUnspecifiedShellInt	CSG
XmNminAspectY	XmCMinAspectY	int	XtUnspecifiedShellInt	CSG
XmNminHeight	XmCMinHeight	int	XtUnspecifiedShellInt	CSG
XmNminWidth	XmCMinWidth	int	XtUnspecifiedShellInt	CSG
XmNtitle	XmCTitle	String	dynamic	CSG
XmNtitleEncoding	XmCTitleEncoding	Atom	dynamic	CSG
XmNtransient	XmCTransient	Boolean	False	CSG
XmNwaitForWm	XmCWaitForWm	Boolean	True	CSG
XmNwidthInc	XmCWidthInc	int	XtUnspecifiedShellInt	CSG
XmNwindowGroup	XmCWindowGroup	Window	dynamic	CSG
XmNwinGravity	XmCWinGravity	int	dynamic	CSG
XmNwmTimeout	XmCWmTimeout	int	5000	CSG

XmNbaseHeight

XmNbaseWidth
The base dimensions from which the preferred height and width can be stepped up or down (as specified by XmNheightInc or XmNwidthInc).

XmNheightInc
The amount by which to increment or decrement the window's height when the window manager chooses a preferred value. The base height is XmNbaseHeight, and the height can decrement to the value of XmNminHeight or increment to the value of XmN-maxHeight. See also XmNwidthInc.

XmNiconMask
A bitmap that the window manager can use in order to clip the application's icon into a non-rectangular shape.

XmNiconPixmap
The application's icon.

XmNiconWindow
The ID of a window that serves as the application's icon.

XmNiconX

XmNiconY
Window manager hints for the root window coordinates of the application's icon.

XmNinitialState
The initial appearance of the widget instance. Possible values are defined in *<X11/Xutil.h>*:

```
NormalState     /* application starts as a window */
IconicState     /* application starts as an icon */
```

XmNinput
A Boolean that, in conjunction with the WM_TAKE_FOCUS atom in the WM_PROTOCOLS property, determines the application's keyboard focus model. The result is determined by the value of XmNinput and the existence of the atom, as described below:

Value of XmNinput Resource	WM_TAKE_FOCUS Atom	Keyboard Focus Model
False	Does not exist	No input allowed
True	Does not exist	Passive
False	Exists	Globally active
True	Exists	Locally active

XmNmaxAspectX

XmNmaxAspectY
The numerator and denominator, respectively, of the maximum aspect ratio requested for this widget.

XmNmaxHeight

XmNmaxWidth
The maximum dimensions for the widget's preferred height or width.

XmNminAspectX

XmNminAspectY
The numerator and denominator, respectively, of the minimum aspect ratio requested for this widget.

XmNminHeight

XmNminWidth
The minimum dimensions for the widget's preferred height or width.

XmNtitle
The string that the window manager displays as the application's name. By default, the icon name is used, but if this isn't specified, the name of the application is used.

XmNtitleEncoding
The property type for encoding the XmNtitle resource.

XmNtransient
If True, this indicates a popup window or some other transient widget. This resource is usually not changed.

XmNwaitForWm
If True (default), the X Toolkit waits for a response from the window manager before acting as if no window manager exists. The waiting time is specified by the XmNwmTimeout resource.

XmNwidthInc
The amount by which to increment or decrement the window's width when the window manager chooses a preferred value. The base width is XmNbaseWidth, and the width can decrement to the value of XmNminWidth or increment to the value of XmN-maxWidth. See also XmNheightInc.

XmNwindowGroup
The window associated with this widget instance. This window acts as the primary window of a group of windows that have similar behavior.

XmNwinGravity
The window gravity used in positioning the widget. Unless an initial value is given, this resource will be set when the widget is realized. The default value is NorthWestGravity (if the Shell resource XmNgeometry is NULL); otherwise, XmNwinGravity assumes the value returned by the XmWMGeometry routine.

XmNwmTimeout
The number of milliseconds that the X Toolkit waits for a response from the window manager. This resource is meaningful when the XmNwaitForWm resource is set to True.

Inherited Resources

WMShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinitialResourcesPersistent	Core
XmNallowShellResize	Shell	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNmappedWhenManaged	Core
XmNbackground	Core	XmNnumChildren	Composite
XmNbackgroundPixmap	Core	XmNoverrideRedirect	Shell
XmNborderColor	Core	XmNpopdownCalback	Shell
XmNborderPixmap	Core	XmNpopupCallback	Shell
XmNborderWidth	Core	XmNsaveUnder	Shell
XmNchildren	Composite	XmNscreen	Core

Resource	Inherited From	Resource	Inherited From
XmNcolormap	Core	XmNsensitive	Core
XmNcreatePopupChildProc	Shell	XmNtranslations	Core
XmNdepth	Core	XmNvisual	Shell
XmNdestroyCallback	Core	XmNwidth	Core
XmNgeometry	Shell	XmNx	Core
XmNheight	Core	XmNy	Core

See Also

`Composite`(2), `Core`(2), `Shell`(2).

Name

XmArrowButton widget class –a directional arrow-shaped button widget.

Synopsis

Public Header:
<Xm/ArrowB.h>

Class Name:
XmArrowButton

Class Hierarchy:
Core → XmPrimitive → XmArrowButton

Class Pointer:
xmArrowButtonWidgetClass

Instantiation:
widget = `XmCreateArrowButton` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmArrowButtonWidgetClass,...)

Functions/Macros:
`XmCreateArrowButton`(), `XmIsArrowButton`()

Description

An ArrowButton is a directional arrow-shaped button that includes a shaded border. The shading changes to make the ArrowButton appear either pressed in when selected or raised when unselected.

Traits

ArrowButton holds the XmQTactivatable trait, which is inherited by any derived classes.

New Resources

ArrowButton defines the following resources:

Name	Class	Type	Default	Access
XmNarrowDirection	XmCArrowDirection	unsigned char	XmARROW_UP	CSG
XmNdetailShadowThickness	XmCShadowThickness	Dimension	dynamic	CSG
XmNmultiClick	XmCMultiClick	unsigned char	dynamic	CSG

XmNarrowDirection
Sets the arrow direction. Possible values:

XmARROW_UP	XmARROW_LEFT
XmARROW_DOWN	XmARROW_RIGHT

XmNdetailShadowThickness
In Motif 2.0 and later, specifies the thickness of the shadow inside the triangle of the ArrowButton. Values of 0 (zero), 1, and 2 are supported. In Motif 2.0, the default is 2. In Motif 2.1 and later, the default depends upon the value of the XmDisplay resource XmNenableThinThickness: if True, the default is 1, otherwise 2.

XmNmultiClick
A flag that determines whether successive button clicks are processed or ignored. Possible values:

```
XmMULTICLICK_DISCARD    /* ignore successive button clicks; */
                        /* default value in a menu system   */
XmMULTICLICK_KEEP       /* count successive button clicks; */
                        /* default value when not in a menu */
```

Callback Resources

ArrowButton defines the following callback resources:

Callback	Reason Constant
XmNactivateCallback	XmCR_ACTIVATE
XmNarmCallback	XmCR_ARM
XmNdisarmCallback	XmCR_DISARM

XmNactivateCallback
List of callbacks that are called when BSelect is pressed and released inside the widget.

XmNarmCallback
List of callbacks that are called when BSelect is pressed while the pointer is inside the widget.

XmNdisarmCallback
List of callbacks that are called when BSelect is released after it has been pressed inside the widget.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
```

```
    XEvent      *event;         /* event structure that triggered callback */
    int         click_count;    /* number of clicks in multi-click sequence */
} XmArrowButtonCallbackStruct;
```

click_count is meaningful only for XmNactivateCallback. Furthermore, if the XmN-multiClick resource is set to XmMULTICLICK_KEEP, then XmN-activate-Callback is called for each click, and the value of click_count is the number of clicks that have occurred in the last sequence of multiple clicks. If the XmNmultiClick resource is set to XmMULTICLICK_DISCARD, then click_count always has a value of 1.

Inherited Resources

ArrowButton inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNhighlightThickness	XmPrimitive
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNlayoutDirection	XmPrimitive
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnavigationType	XmPrimitive
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmPrimitive
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmPrimitive	XmNsensitive	Core
XmNbottomShadowPixmap	XmPrimitive	XmNshadowThickness	XmPrimitive
XmNcolormap	Core	XmNtoolTipString	XmPrimitive
XmNconvertCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNdepth	Core	XmNtopShadowPixmap	XmPrimitive
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNheight	Core	XmNunitType	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNuserData	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNwidth	Core
XmNhighlightOnEnter	XmPrimitive	XmNx	Core
XmNhighlightPixmap	XmPrimitive	XmNy	Core

Translations

The translations of ArrowButton include those of Primitive

Event	Action
BSelect Press	Arm()
BSelect Click	Activate() Disarm()
BSelect Release	Activate() Disarm()
Bselect Press 2+	MultiArm()
BSelect Release 2+	MultiActivate()
KSelect	ArmAndActivate()
MCtrl BSelect Press	ButtonTakeFocus()
KHelp	Help()

Action Routines

ArrowButton defines the following action routines:

Activate()

Displays the ArrowButton as unselected, and invokes the list of callbacks specified by XmNactivateCallback.

Arm()

Displays the ArrowButton as selected, and invokes the list of callbacks specified by XmNarmCallback.

ArmAndActivate()

Displays the ArrowButton as selected, and invokes the list of callbacks specified by XmNarmCallback. After doing this, the action routine displays the ArrowButton as unselected, and invokes the list of callbacks specified by XmNactivateCallback and XmNdisarmCallback.

ButtonTakeFocus()

In Motif 2.0 and later, moves the current keyboard focus to the ArrowButton, without activating the widget.

Disarm()

Displays the ArrowButton as unselected, and invokes the list of callbacks specified by XmNdisarmCallback.

Help()

Invokes the list of callbacks specified by XmNhelpCallback. If the ArrowButton doesn't have any help callbacks, the Help() routine invokes those associated with the nearest ancestor that has them.

MultiActivate()
: Increments the click_count member of XmArrowButtonCallbackStruct, displays the ArrowButton as unselected, and invokes the list of callbacks specified by XmNactivateCallback and XmNdisarmCallback. This action routine takes effect only when the XmNmultiClick resource is set to XmMULTICLICK_KEEP.

MultiArm()
: Displays the ArrowButton as selected, and invokes the list of callbacks specified by XmNarmCallback. This action routine takes effect only when the XmNmultiClick resource is set to XmMULTICLICK_KEEP.

Additional Behavior

ArrowButton has the following additional behavior:

<EnterWindow>
: Displays the ArrowButton as selected if the pointer leaves and re-enters the window while BSelect is pressed.

<LeaveWindow>
: Displays the ArrowButton as unselected if the pointer leaves the window while BSelect is pressed.

See Also

`XmCreateObject`(1), `Core`(2), `XmPrimitive`(2).

Name

XmArrowButtonGadget widget class – a directional arrow-shaped button gadget.

Synopsis

Public Header:
<Xm/ArrowBG.h>

Class Name:
XmArrowButtonGadget

Class Hierarchy:
Object → RectObj → XmGadget → XmArrowButtonGadget

Class Pointer:
xmArrowButtonGadgetClass

Instantiation:
widget = `XmCreateArrowButtonGadget` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmArrowButtonGadgetClass,...)

Functions/Macros:
`XmCreateArrowButtonGadget`(), `XmIsArrowButtonGadget`()

Description

ArrowButtonGadget is the gadget variant of ArrowButton.

ArrowButtonGadget's resources, callback resources, and callback structure are the same as those of ArrowButton.

Traits

ArrowButtonGadget holds the XmQTactivatable, XmQTcareParentVisual, and XmQTaccessColors traits, which are inherited by any derived classes.

Inherited Resources

ArrowButtonGadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of XmNborderWidth is reset to 0 by Gadget.

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	RectObj	XmNlayoutDirection	XmGadget
XmNbackground	XmGadget	XmNnavigationType	XmGadget
XmNbackgroundPixmap	XmGadget	XmNsensitive	RectObj
XmNbottomShadowColor	XmGadget	XmNshadowThickness	XmGadget
XmNbottomShadowPixmap	XmGadget	XmNtoolTipString	XmGadget
XmNborderWidth	RectObj	XmNtopShadowColor	XmGadget

Resource	Inherited From	Resource	Inherited From
XmNdestroyCallback	Object	XmNtopShadowPixmap	XmGadget
XmNforeground	XmGadget	XmNtraversalOn	XmGadget
XmNheight	RectObj	XmNunitType	XmGadget
XmNhelpCallback	XmGadget	XmNuserData	XmGadget
XmNhighlightColor	XmGadget	XmNwidth	RectObj
XmNhighlightOnEnter	XmGadget	XmNx	RectObj
XmNhighlightPixmap	XmGadget	XmNy	RectObj
XmNhighlightThickness	XmGadget		

Behavior

As a gadget subclass, ArrowButtonGadget has no translations associated with it. However, ArrowButtonGadget behavior corresponds to the action routines of the ArrowButton widget. See the ArrowButton action routines for more information.

Event	Action
BSelect Press	Arm()
BSelect Click	Activate() Disarm()
BSelect Release	Activate() Disarm()
Bselect Press 2+	MultiArm()
BSelect Release 2+	MultiActivate()
KSelect	ArmAndActivate()
MCtrl BSelect Press	ButtonTakeFocus()
KHelp	Help()

ArrowButtonGadget has additional behavior associated with <Enter> and <Leave>, which display the ArrowButtonGadget as selected if the pointer leaves and re-enters the gadget while BSelect is pressed or as unselected if the pointer leaves the gadget while BSelect is pressed.

See Also

`XmCreateObject`(1), `Core`(2), `Object`(2), `RectObj`(2), `XmArrowButton`(2), `XmGadget`(2), `XmPrimitive`(2).

Name

XmBulletinBoard widget class – a simple geometry-managing widget.

Synopsis

Public Header:
<Xm/BulletinB.h>

Class Name:
XmBulletinBoard

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmBulletinBoard

Class Pointer:
xmBulletinBoardWidgetClass

Instantiation:
widget = XmCreateBulletinBoard (parent, name,...)
or
widget = XtCreateWidget (name, xmBulletinBoardWidgetClass,...)

Functions/Macros:
`XmCreateBulletinBoard`(), `XmCreateBulletinBoardDialog`()

Description

BulletinBoard is a general-purpose manager that allows children to be placed at arbitrary x, y positions. The simple geometry management of BulletinBoard can be used to enforce margins and to prevent child widgets from overlapping. BulletinBoard is the base widget for most dialog widgets and defines many resources that have an effect only when it is an immediate child of a DialogShell.

Traits

BulletinBoard holds the XmQTspecifyRenderTable and XmQTdialogShellSavvy traits, which are inherited by any derived classes, and uses the XmQTtakesDefault and XmQTspecifyRenderTable traits.

New Resources

BulletinBoard defines the following resources:

Name	Class	Type	Default	Access
XmNallowOverlap	XmCAllowOverlap	Boolean	True	CSG
XmNautoUnmanage	XmCAutoUnmanage	Boolean	True	CSG
XmNbuttonFontList	XmCButtonFontList	XmFontList	dynamic	CSG
XmNbuttonRenderTable	XmCButtonRenderTable	XmRenderTable	dynamic	CSG
XmNcancelButton	XmCWidget	Widget[a]	NULL	SG

Name	Class	Type	Default	Access
XmNdefaultButton	XmCWidget	Widget[b]	NULL	SG
XmNdefaultPosition	XmCDefaultPosition	unsigned char	True	CSG
XmNdialogStyle	XmCDialogStyle	unsigned char	dynamic	CSG
XmNdialogTitle	XmCDialogTitle	XmString	NULL	CSG
XmNlabelFontList	XmCLabelFontList	XmFontList	dynamic	CSG
XmNlabelRenderTable	XmCLabelRenderTable	XmRenderTable	dynamic	CSG
XmNmarginHeight	XmCMarginHeight	Dimension	10	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	10	CSG
XmNnoResize	XmCNoResize	Boolean	False	CSG
XmNresizePolicy	XmCResizePolicy	unsigned char	XmRESIZE_ANY	CSG
XmNshadowType	XmCShadowType	unsigned char	XmSHADOW_OUT	CSG
XmNtextFontList	XmCTextFontList	XmFontList	dynamic	CSG
XmNtextRenderTable	XmCTextRenderTable	XmRenderTable	dynamic	CSG
XmNtextTranslations	XmCTranslations	XtTranslations	NULL	C

a.Erroneously given as Window in 2nd Edition.
b.Erroneously given as Window in 2nd Edition.

XmNallowOverlap
: If True (default), child widgets are allowed to overlap.

XmNautoUnmanage
: If True (default), the BulletinBoard is automatically unmanaged after a button is activated unless the button is an **Apply** or **Help** button.

XmNbuttonFontList
: Specifies the font list used for the button descendants of the BulletinBoard widget. In Motif 2.0 and later, the XmFontList is considered obsolete, and is replaced by the XmRenderTable. The XmNbuttonRenderTable resource is the preferred method of specifying appearance.

XmNbuttonRenderTable
: In Motif 2.0 and later, specifies the render table used for any button descendants of the BulletinBoard widget. If NULL, this is inherited from the nearest ancestor that has the XmQTspecifyRenderTable trait, using the XmBUTTON_RENDER_TABLE value of any ancestor so found.
The button render table resource takes precedence over any specified XmNbuttonFontList.

XmNcancelButton
: The widget ID of the **Cancel** button. The subclasses of BulletinBoard define a **Cancel** button and set this resource.

XmNdefaultButton
: The widget ID of the default button. Some of the subclasses of BulletinBoard define a default button and set this resource. To indicate that it is the default, this button appears different from the others.

XmNdefaultPosition
: If True (default) and if the BulletinBoard is the child of a DialogShell, then the BulletinBoard is centered relative to the DialogShell's parent.

XmNdialogStyle
: The BulletinBoard's dialog style, whose value can be set only if the BulletinBoard is unmanaged. Possible values:

```
XmDIALOG_WORK_AREA          /*default when parent is not a DialogShell */
XmDIALOG_MODELESS           /*default when parent is a DialogShell    */*
XmDIALOG_FULL_APPLICATION_MODAL
XmDIALOG_APPLICATION_MODAL
XmDIALOG_PRIMARY_APPLICATION_MODAL
XmDIALOG_SYSTEM_MODAL
```

  The value XmDIALOG_APPLICATION_MODAL, although maintained for backwards compatibility, is deprecated in Motif 1.2 and later. Use XmDIALOG_PRIMARY_APPLICATION_MODAL instead.

XmNdialogTitle
: The dialog title. Setting this resource also sets the resources XmNtitle and XmN--titleEncoding in a parent that is a subclass of WMShell.

XmNlabelFontList
: Specifies the font list used for the label descendants of the BulletinBoard widget. In Motif 2.0 and later, the XmFontList is considered obsolete, and is replaced by the XmRenderTable. The XmNlabelRenderTable resource is the preferred method of specifying appearance.

XmNlabelRenderTable
: In Motif 2.0 and later, specifies the render table used for any label descendants of the BulletinBoard widget. If NULL, this is inherited from the nearest ancestor that has the XmQTspecifyRenderTable trait, using the XmLABEL_RENDER_TABLE value of any ancestor so found.

XmNmarginHeight
: Minimum spacing between a BulletinBoard's top or bottom edge and any child widget.

XmNmarginWidth
: Minimum spacing between a BulletinBoard's right or left edge and any child widget.

XmNnoResize
If False (default), *mwm* includes resize controls in the window manager frame of the BulletinBoard's shell parent.

XmNresizePolicy
How BulletinBoard widgets are resized. Possible values:

```
XmRESIZE_NONE     /* remain at fixed size       */
XmRESIZE_GROW     /* expand only                */
XmRESIZE_ANY      /* shrink or expand, as needed */
```

XmNshadowType
The style in which shadows are drawn. Possible values:

```
XmSHADOW_IN           /* widget appears inset            */
XmSHADOW_OUT          /* widget appears outset           */
XmSHADOW_ETCHED_IN    /* double line; widget appears inset */
XmSHADOW_ETCHED_OUT   /* double line; widget appears raised */
```

XmNtextFontList
Specifies the font list used for the text descendants of the BulletinBoard widget. In Motif 2.0 and later, the XmFontList is considered obsolete, and is replaced by the XmRenderTable. The XmNtextRenderTable resource is the preferred method of specifying appearance.

XmNtextRenderTable
In Motif 2.0 and later, specifies the render table used for any text descendants of the BulletinBoard widget. If NULL, this is inherited from the nearest ancestor that has the XmQTspecifyRenderTable trait, using the XmTEXT_RENDER_TABLE value of any ancestor so found.

XmNtextTranslations
For any Text widget (or its subclass) that is a child of a BulletinBoard, this resource adds translations.

Callback Resources

BulletinBoard defines the following callback resources:

Callback	Reason Constant
XmNfocusCallback	XmCR_FOCUS
XmNmapCallback	XmCR_MAP
XmNunmapCallback	XmCR_UNMAP

XmNfocusCallback
List of callbacks that are called when the widget or one of its descendants receives the input focus.

XmNmapCallback
List of callbacks that are called when the widget is mapped, if it is a child of a DialogShell.

XmNunmapCallback
List of callbacks that are called when the widget is unmapped, if it is a child of a DialogShell.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
    XEvent      *event;         /* points to event structure that triggered callback */
} XmAnyCallbackStruct;
```

Inherited Resources

BulletinBoard inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. BulletinBoard sets the value of XmNinitialFocus to the value of XmNdefaultButton. When it is a child of a DialogShell, BulletinBoard resets the default XmNshadowThickness from 0 to 1. The default value of XmNborderWidth is reset to 0 by Manager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core

Resource	Inherited From	Resource	Inherited From
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Translations

The translations for BulletinBoard include those of XmManager.

Additional Behavior

BulletinBoard has the following additional behavior:

MAny KCancel
: For a sensitive **Cancel** button, invokes the XmNactivateCallback callbacks.

KActivate
: For the button that has keyboard focus, invokes the XmNactivate-Callback callbacks.

<FocusIn>
: Invokes the XmNfocusCallback callbacks. The widget receives focus either when the user traverses to it (XmNkeyboardFocusPolicy is XmEXPLICIT) or when the pointer enters the window (XmNkeyboardFocusPolicy is XmPOINTER).

<Map>
: Invokes the XmNmapCallback callbacks.

<Unmap>
: Invokes the XmNunmapCallback callbacks.

See Also

`XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmBulletinBoardDialog`(2), `XmDialogShell`(2), `XmManager`(2).

Name

XmBulletinBoardDialog –an unmanaged BulletinBoard as a child of a DialogShell.

Synopsis

Public Header:
<Xm/BulletinB.h>

Instantiation:
widget = `XmCreateBulletinBoardDialog`(...)

Functions/Macros:
`XmCreateBulletinBoardDialog`()

Description

An XmBulletinBoardDialog is a compound object created by a call to XmCreateBulletinBoardDialog() that is useful for creating custom dialogs. A BulletinBoardDialog consists of a DialogShell with an unmanaged BulletinBoard widget as its child. The BulletinBoardDialog does not contain any labels, buttons, or other dialog components; these components are added by the application.

Default Resource Values

A BulletinBoardDialog sets the following default values for BulletinBoard resources:

Name	Default
XmNdialogStyle	XmDIALOG_MODELESS

Widget Hierarchy

When a BulletinBoardDialog is created with a specified *name*, the DialogShell is named *name_popup* and the BulletinBoard is called *name*.

See Also

`XmCreateObject`(1), `XmBulletinBoard`(2), `XmDialogShell`(2).

Name

XmButtonBox – The ButtonBox class

Synopsis

Public Headers:
<Xm/ButtonBox.h>

Class Name:
XmButtonBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmButtonBox

Class Pointer:
xmButtonBoxWidgetClass

Instantiation:

```
widget = XmCreateButtonBox(parent, name,...)
or
widget = XtCreateWidget(name, xmButtonBoxWidget-
Class,...)
```

Functions/Macros:

```
XmCreateButtonBox()
```

Availability

OpenMotif 2.2 and later. (Provisional Widget)

Description

The Button Box widget manages children (usually buttons) in a single row or single column layout.

The Button Box maintains equal spacing between its children at all times and attempts to adjust its height and width as necessary so that all children will fit. If this is not possible, due to parent or application programmer constraints, the Button Box will resize its children as necessary to fit within the available space.

Note: in what follows, major direction refers to the direction of orientation, and minor direction refers to the perpendicular direction.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using

XtSetValues (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNequalSize	XmCEqualSize	Boolean	False	CSG
XmNfillOption	XmCFillOption	unsigned char	XmFillNone	CSG
XmNmarginHeight	XmCMargin	VerticalDimension	0	CSG
XmNmarginWidth	XmCMargin	HorizontalDimension	0	CSG
XmNorientation	XmCOrientation	unsigned char	XmHORIZONTAL	CSG
XmNdefaultButton	XmCXmCWidget	Widget	NULL	SG

XmNequalSize

Specifies whether the children are to be maintained with equal sized heights and widths. The chosen height and width for the children is found by asking each child for its preferred size and taking the largest value in each direction.

XmNfillOption

Specifies how to use any extra space left over once all children have been sized according to either their preference or equalSize. There are four options:

XmFillNone

No automatic filling is performed. Center the children in the minor direction and place the children with equal padding between them in the major direction.

XmFillMinor

Place the children with equal padding between them in the major direction, but force all the children to take on the Button Box minor dimension as their minor direction.

XmFillMajor

Center the children in the minor direction, but expand all the children in their major direction so that there is no padding between them. Expand the children such that their relative sizes remain constant.

XmFillAll

This option combines the placement actions and sizing actions of XmFillMinor and XmFillMajor.

Regardless of the fill mode, the ButtonBox widget will always leave the specified margin between its edge and the nearest child. A new String to fillOption resource converter has been registered to convert the following strings to fill options: "none", "major", "minor", "all." This resource can therefore be set in an application defaults file.

XmNmarginHeight
XmNmarginWidth
: Specifies the number of pixels to use as a margin around the entire group of children. The marginHeight value applies to both the top and bottom margins, while the marginWidth applies to the left and right margins.

XmNorientation
: Specifies whether children are to be placed in a row or a column. The orientation may be either XmHORIZONTAL or XmVERTICAL. If the orientation is XmHORIZONTAL, the children are placed in a row with the major dimension being width and the minor dimension being height. If the value is XmVERTICAL, the children are placed in a column with the major dimension being height and the minor dimension being width. The default value is XmHORIZONTAL.

Inherited Resources

Resource	Inherited from	Resource	Inherited From
XmNbottomShadowColor	XmManager	XmNaccelerators	Core
XmNbottomShadowPixmap	XmManager	XmNancestorSensitive	Core
XmNforeground	XmManager	XmNbackground	Core
XmNhelpCallback	XmManager	XmNbackgroundPixmap	Core
XmNhighlightColor	XmManager	XmNborderColor	Core
XmNhighlightPixmap	XmManager	XmNborderPixmap	Core
XmNinitialFocus	XmManager	XmNborderWidth	Core
XmNlayoutDirection	XmManager	XmNcolormap	Core
XmNnavigationType	XmManager	XmNdepth	Core
XmNpopupHandlerCallback	XmManager	XmNdestroyCallback	Core
XmNshadowThickness	XmManager	XmNheight	Core
XmNstringDirection	XmManager	XmNinitialResourcesPersistent	Core
XmNtopShadowColor	XmManager	XmNmappedWhenManaged	Core
XmNtopShadowPixmap	XmManager	XmNscreen	Core
XmNtraversalOn	XmManager	XmNsensitive	Core
XmNunitType	XmManager	XmNtranslations	Core
XmNuserData	XmManager	XmNwidth	Core
XmNchildren	Composite	XmNx	Core
XmNinsertPosition	Composite	XmNy	Core

Resource	Inherited from	Resource	Inherited From
XmNnumChildren	Composite		

Translations

XmButtonBox inherits translations from XmManager.

See Also

XmCreateObject(1), Composite(2), Constrait(2), Core(2), XmColumn, XmCreateButtonBox(3), XmIconBox, XmManager(2), and XmRowColumn.

Name

XmCascadeButton widget class –a button widget that posts menus.

Synopsis

Public Header:
<Xm/CascadeB.h>

Class Name:
XmCascadeButton

Class Hierarchy:
Core → XmPrimitive → XmLabel → XmCascadeButton

Class Pointer:
xmCascadeButtonWidgetClass

Instantiation:
widget = `XmCreateCascadeButton` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmCascadeButtonWidgetClass,...)

Functions/Macros:
`XmCascadeButtonHighlight`(), `XmCreateCascadeButton`(), `XmIs-CascadeButton`()

Description

CascadeButtons are used in menu systems to post menus. A CascadeButton either links a menu bar to a menu pane or connects a menu pane to another menu pane. The widget can have a menu attached to it as a submenu.

Traits

CascadeButton uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

New Resources

CascadeButton defines the following resources:

Name	Class	Type	Default	Access
XmNcascadePixmap	XmCPixmap	Pixmap	dynamic	CSG
XmNmappingDelay	XmCMappingDelay	int	180	CSG
XmNsubMenuId	XmCMenuWidget	Widget	NULL	CSG

XmNcascadePixmap
The pixmap within the CascadeButton that indicates a submenu. By default, this pixmap is an arrow pointing toward the submenu to be popped up.

XmNmappingDelay
The number of milliseconds it should take for the application to display a submenu after its CascadeButton has been selected.

XmNsubMenuId
The widget ID of the pulldown menu pane associated with the CascadeButton. The menu pane is displayed when the CascadeButton is selected. The pulldown menu pane and the CascadeButton must have a common parent.

Callback Resources

CascadeButton defines the following callback resources:

Callback	Reason Constant
XmNactivateCallback	XmCR_ACTIVATE
XmNcascadingCallback	XmCR_CASCADING

XmNactivateCallback
List of callbacks that are called when BSelect is pressed and released while the pointer is inside the widget and there is no submenu to post.

XmNcascadingCallback
List of callbacks that are called before the submenu associated with the CascadeButton is mapped.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int     reason;     /* the reason that the callback was called */
    XEvent  *event;     /* event structure that triggered callback */
} XmAnyCallbackStruct;
```

Inherited Resources

CascadeButton inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. CascadeButton sets the default values of XmNmarginBottom, XmNmarginRight, XmNmarginTop, XmNmarginWidth, and XmN-traversalOn dynamically. In Motif 2.0 and earlier, the default values of XmNhighlightThickness and XmNshadowThickness are reset to 2. In Motif 2.1, the default values depend upon the XmDisplay XmNenableThinThickness resource: if True the default is 1, otherwise 2. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNaccelerator	XmLabel	XmNlabelType	XmLabel
XmNaccelerators	Core	XmNmappedWhenManaged	Core
XmNacceleratorText	XmLabel	XmNmarginBottom	XmLabel
XmNalignment	XmLabel	XmNmarginHeight	XmLabel
XmNancestorSensitive	Core	XmNmarginLeft	XmLabel
XmNbackground	Core	XmNmarginRight	XmLabel
XmNbackgroundPixmap	Core	XmNmarginTop	XmLabel
XmNborderColor	Core	XmNmarginWidth	XmLabel
XmNborderPixmap	Core	XmNmnemonicCharSet	XmLabel
XmNborderWidth	Core	XmNmnemonic	XmLabel
XmNbottomShadowColor	XmPrimitive	XmNnavigationType	XmPrimitive
XmNbottomShadowPixmap	XmPrimitive	XmNpopupHandlerCallback	XmPrimitive
XmNcolormap	Core	XmNrecomputeSize	XmLabel
XmNconvertCallback	XmPrimitive	XmNrenderTable	XmLabel
XmNdepth	Core	XmNscreen	Core
XmNdestroyCallback	Core	XmNsensitive	Core
XmNfontList	XmLabel	XmNshadowThickness	XmPrimitive
XmNforeground	XmPrimitive	XmNstringDirection	XmLabel
XmNheight	Core	XmNtoolTipString	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNtopShadowPixmap	XmPrimitive
XmNhighlightOnEnter	XmPrimitive	XmNtranslations	Core
XmNhighlightPixmap	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNhighlightThickness	XmPrimitive	XmNunitType	XmPrimitive
XmNinitialResourcesPersistent	Core	XmNuserData	XmPrimitive
XmNlabelInsensitivePixmap	XmLabel	XmNwidth	Core
XmNlabelPixmap	XmLabel	XmNx	Core
XmNlabelString	XmLabel	XmNy	Core
XmNlayoutDirection	XmPrimitive		

Translations

The translations of CascadeButton include the menu traversal translations of Label.

Event	Action
BSelect Press	MenuBarSelect() (in a menu bar) StartDrag() (in a popup or pulldown menu)
BSelect Release	DoSelect()
MCtrl BSelect Press	MenuButtonTakeFocus()
MCtrl BSelect Release	MenuButtonTakeFocusUp()
KActivate	KeySelect()
KSelect	KeySelect()
KHelp	Help()
MAny KCancel	CleanupMenuBar()

Action Routines

CascadeButton defines the following action routines:

CleanupMenuBar()

Unposts any menus and restores the keyboard focus to the group of widgets (tab group) that had the focus before the CascadeButton was armed.

DoSelect()

Posts the CascadeButton's submenu and allows keyboard traversal. If there is no submenu attached to the CascadeButton, this action routine activates the CascadeButton and unposts all the menus in the cascade.

Help()

Similar to CleanupMenuBar() in that the Help() routine unposts any menus and restores keyboard focus. This routine also invokes the list of callbacks specified by XmNhelpCallback. If the CascadeButton doesn't have any help callbacks, the Help() routine invokes those associated with the nearest ancestor that has them.

KeySelect()

Posts the CascadeButton's submenu, provided that keyboard traversal is allowed. If there is no submenu attached to the CascadeButton, this action routine activates the CascadeButton and unposts all the menus in the cascade.

MenuBarSelect()
: Unposts any previously posted menus, posts the submenu associated with the CascadeButton, and enables mouse traversal.

MenuButtonTakeFocus()
: In Motif 2.0 and later, moves the current keyboard focus to the CascadeButton, without activating the widget.

StartDrag()
: Posts the submenu associated with the CascadeButton and enables mouse traversal.

Additional Behavior

CascadeButton has the following additional behavior:

<EnterWindow>	Arms the CascadeButton and posts its submenu.
<LeaveWindow>	Disarms the CascadeButton and unposts its submenu.

See Also

`XmCascadeButtonHighlight`(1), `XmCreateObject`(2), `Core`(2), `XmLabel`(2), `XmPrimitive`(2), `XmRowColumn`(2).

Name

XmCascadeButtonGadget widget class –a button gadget that posts menus.

Synopsis

Public Header:
<Xm/CascadeBG.h>

Class Name:
XmCascadeButtonGadget

Class Hierarchy:
Object → RectObj → XmGadget → XmLabelGadget → XmCascadeButtonGadget

Class Pointer:
xmCascadeButtonGadgetClass

Instantiation:
widget = `XmCreateCascadeButtonGadget` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmCascadeButtonGadgetClass,...)[1]

Functions/Macros:
`XmCascadeButtonGadgetHighlight`(), `XmCreateCascadeButtonGadget`(), `XmIsCascadeButtonGadget`(), `XmOptionButtonGadget`()

Description

CascadeButtonGadget is the gadget variant of CascadeButton.

CascadeButtonGadget's new resources, callback resources, and callback structure are the same as those for CascadeButton.

Traits

CascadeButtonGadget uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

Inherited Resources

CascadeButtonGadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. CascadeButtonGadget sets the default values of XmNmarginBottom, XmNmarginRight, XmNmarginTop, and XmNmarginWidth dynamically. It also sets the default value of XmNhighlightThickness to 0. The default value of XmNborderWidth is reset to 0 by Gadget.

1.Erroneously given as xmCascadeButtonWidgetClass in 2nd Edition.

Resource	Inherited From	Resource	Inherited From
XmNaccelerator	XmLabelGadget	XmNmarginHeight	XmLabelGadget
XmNacceleratorText	XmLabelGadget	XmNmarginLeft	XmLabelGadget
XmNalignment	XmLabelGadget	XmNmarginRight	XmLabelGadget
XmNancestorSensitive	RectObj	XmNmarginTop	XmLabelGadget
XmNbackground	XmGadget	XmNmarginWidth	XmLabelGadget
XmNbackgroundPixmap	XmGadget	XmNmnemonic	XmLabelGadget
XmNbottomShadowColor	XmGadget	XmNmnemonicCharSet	XmLabelGadget
XmNbottomShadowPixmap	XmGadget	XmNnavigationType	XmGadget
XmNborderWidth	RectObj	XmNrecomputeSize	XmLabelGadget
XmNdestroyCallback	Object	XmNrenderTable	XmLabelGadget
XmNfontList	XmLabelGadget	XmNsensitive	RectObj
XmNforeground	XmGadget	XmNshadowThickness	XmGadget
XmNheight	RectObj	XmNstringDirection	XmLabelGadget
XmNhelpCallback	XmGadget	XmNtoolTipString	XmGadget
XmNhighlightColor	XmGadget	XmNtopShadowColor	XmGadget
XmNhighlightOnEnter	XmGadget	XmNtopShadowPixmap	XmGadget
XmNhighlightPixmap	XmGadget	XmNtraversalOn	XmGadget
XmNhighlightThickness	XmGadget	XmNunitType	XmGadget
XmNlabelInsensitivePixmap	XmLabelGadget	XmNuserData	XmGadget
XmNlabelPixmap	XmLabelGadget	XmNwidth	RectObj
XmNlabelType	XmLabelGadget	XmNx	RectObj
XmNlayoutDirection	XmGadget	XmNy	RectObj
XmNmarginBottom	XmLabelGadget		

Behavior

As a gadget subclass, CascadeButtonGadget has no translations associated with it. However, CascadeButtonGadget behavior corresponds to the action routines of the CascadeButton widget. See the CascadeButton action routines for more information.

Event	Action
BSelect Press	MenuBarSelect() (in a menu bar) StartDrag() (in a popup or pulldown menu)
BSelect Release	DoSelect()

Event	Action
MCtrl BSelect Press	MenuButtonTakeFocus()
MCtrl BSelect Release	MenuButtonTakeFocusUp()
KActivate	KeySelect()
KSelect	KeySelect()
KHelp	Help()
MAny KCancel	CleanupMenuBar()

In a menu bar that is armed, CascadeButtonGadget has additional behavior associated with <Enter>, which arms the CascadeButtonGadget and posts its submenu, and with <Leave>, which disarms the CascadeButtonGadget and unposts its submenu.

See Also

`XmCascadeButtonHighlight`(1), `XmCreateObject`(1), `XmOptionButtonGadget`(1), `Object`(2), `RectObj`(2), `XmCascadeButton`(2), `XmGadget`(2), `XmLabelGadget`(2), `XmRowColumn`(2).

Name

XmCheckBox –a RowColumn that contains ToggleButtons.

Synopsis

Public Header:
<Xm/RowColumn.h>

Class Name:
XmRowColumn

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmRowColumn

Class Pointer:
xmRowColumnWidgetClass

Instantiation:
widget = `XmCreateSimpleCheckBox` (parent, name,...)

Functions/Macros:
`XmCreateRowColumn`(), `XmCreateSimpleCheckBox`(), `XmIsRowColumn`(),
`XmVaCreateSimpleCheckBox`()

Description

An XmCheckBox is an instance of a RowColumn widget that contains ToggleButton or ToggleButtonGadget children, any number of which may be selected at a given time. A CheckBox is a RowColumn widget with its XmNrowColumnType resource set to XmWORK_AREA and XmNradioAlwaysOne set to False.

A CheckBox can be created by making a RowColumn with these resource values. When it is created in this way, a CheckBox does not automatically contain ToggleButton children; they are added by the application.

A CheckBox can also be created by a call to XmCreateSimpleCheckBox() or XmVaCreateSimpleCheckBox(). These routines automatically create the CheckBox with ToggleButtonGadgets as children. The routines use the RowColumn resources associated with the creation of simple menus. For a CheckBox, the only type allowed in the XmNbuttonType resource is XmCHECKBUTTON. The name of each ToggleButtonGadget is *button_n*, where *n* is the number of the button, ranging from 0 to 1 less than the number of buttons in the CheckBox.

Default Resource Values

A CheckBox sets the following default values for its resources:

Name	Default
XmNnavigationType	XmTAB_GROUP

Name	Default
XmNradioBehavior	False
XmNrowColumnType	XmWORK_AREA
XmNtraversalOn	True

See Also

XmCreateObject(2), XmVaCreateSimpleCheckBox(2), XmRowColumn(2), XmToggleButton(2), XmToggleButtonGadget(2).

Name

XmColorSelector – The ColorSelector class

Synopsis

Public Headers:
<Xm/ColorS.h>

Class Name:
XmColorSelector

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmColorSelector

Class Pointer:
xmColorSelectorWidgetClass

Instantiation:

```
widget = XtCreateColorSelector(parent, name,...)
or
widget = XtCreateWidget(name, xmColorSelectorWidget-
Class,...)
```

Functions/Macros:

```
XmCreateColorSelector()
```

Availability

OpenMotif 2.2 and later. (Contributed Widget)

Description

The Color Selector widget allows users to choose a color by using either a set of RGB sliders or choosing from a list of all colors available in the rgb database. The name or rgb value, as well as the color selected, are dynamically displayed to the user as they pick and choose different colors.

The color selector is composed of many sub-widgets. As with all widgets, most values are passed to this widget through the argument list at creation time or via set values and are passed to each of this widget's children, although get values requests must be made on a child-by-child basis. The children of the color selector are listed below. The documentation for each of the children should be consulted for a list of resources for each child.

XmColorSelector <named by application>

XmScrolledWindow		scrolled
	XmScrollBar	ListvScrollBar
	XmScrollBar	ListhScrollBar
	XmList	list
XmButtonBox		buttonBox
	XmScale	scale
	XmLabelGadget	scale_title

```
            XmScrollBar       scale_scrollbar
    XmRowColumn  radioBox
            XmToggleButton colorListToggle
            XmToggleButton colorSlidersToggle
    XmFrame          colorFrame
            XmLabel           colorWindow
```

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words).The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNblueSliderLabel	XmCSliderLabel	XmString	"Blue"	CSG
XmNcolorListTogLabel	XmCTogLabel	XmString	"Color List"	CSG
XmNcolorMode	XmCColorMode	XmColorMode	XmScaleMode	CSG
XmNcolorName	XmCString	String	"White"	CSG
XmNfileReadError	XmCFileReadError	XmString	"Could not read rgb.txt file"	CSG
XmNgreenSliderLabel	XmCSliderLabel	XmString	"Green"	CSG
XmNmarginHeight	XmCMarginHeight	VerticalDimension	2	CSG
XmNmarginWidth	XmCMarginWidth	HorizontalDimension	2	CSG
XmNnoCellError	XmCNoCellError	XmString	"No Color Cell Available"	CSG
XmNredSliderLabel	XmCSliderLabel	XmString	"Red"	CSG
XmNrgbFile	XmCString	String	"/usr/lib/X11/rgb.txt"	CSG
XmNsliderTogLabel	XmCTogLabel	XmString	"Color Sliders"	CSG

XmNblueSliderLabel

The string appearing for the label of the blue slider

XmNcolorListTogLabel

The string appearing for the label of the color list toggle

XmNcolorMode
: The color list can be used in either slider or list mode. Acceptable values are XmListMode and XmScaleMode. This resource allows the application to determine the mode that the color selector should use when it is created. After this point, the user may freely change modes by utilizing a pair of radio buttons in the color selector. A type converter is registered to convert the strings "ScaleMode" and "ListMode" to color modes for use with the resource database.

XmNcolorName
: This resource controls the color name that is currently displayed to the user. This value can be modified to change the color displayed in the color selector or queried to find the color the user has selected. The string returned here is either a color name or a pound sign (#) followed by a set of rgb values as specified in the Xlib specification.

XmNfileReadError
: The message which is displayed when the Color Selector cannot read the rgb.txt file. The message is displayed at the top of the window in which the color list would normally appear.

XmNgreenSliderLabel
: The string appearing for the label of the green slider

XmNmarginHeight

XmNmarginWidth
: This is the amount of space left between each of the children in the color selector and between the outside children and the edge of the color selector widget.

XmNnoCellError
: This resource controls the message which is displayed in the sample color field when the Color Selector cannot allocate a read/write color cell redSliderLabel The string appearing for the label of the red slider

XmNrgbFile
: This is the name of the file to be loaded, which contains the valid color names. Each of these names is sorted and the duplicates removed before being shown to the user.

XmNsliderTogLabel
: The string appearing for the label of the color slider toggle

Inherited Resources

Color Selector inherits behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass.

Resource	Inherited from	Resource	Inherited from
XmNbottomShadowColor	XmManager	XmNaccelerators	Core
XmNbottomShadowPixmap	XmManager	XmNancestorSensitive	Core
XmNforeground	XmManager	XmNbackground	Core
XmNhelpCallback	XmManager	XmNbackgroundPixmap	Core
XmNhighlightColor	XmManager	XmNborderColor	Core
XmNhighlightPixmap	XmManager	XmNborderPixmap	Core
XmNinitialFocus	XmManager	XmNborderWidth	Core
XmNlayoutDirection	XmManager	XmNcolormap	Core
XmNnavigationType	XmManager	XmNdepth	Core
XmNpopupHandlerCallback	XmManager	XmNdestroyCallback	Core
XmNshadowThickness	XmManager	XmNheight	Core
XmNstringDirection	XmManager	XmNinitialResourcesPersistent	Core
XmNtopShadowColor	XmManager	XmNmappedWhenManaged	Core
XmNtopShadowPixmap	XmManager	XmNscreen	Core
XmNtraversalOn	XmManager	XmNsensitive	Core
XmNunitType	XmManager	XmNtranslations	Core
XmNuserData	XmManager	XmNwidth	Core
XmNchildren	Composite	XmNx	Core
XmNinsertPosition	Composite	XmNy	Core
XmNnumChildren	Composite		

Translations

XmFontSelector inherits translations from XmManager.

See Also

`XmCreateObject(1), Composite(2), Constraint(2), Core(2), and XmManager(2).`

Name

XmColumn – The Column class

Synopsis

Public Headers:
<Xm/Column.h>

Class Name:
XmColumn

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmBulletinBoard → XmColumn

Class Pointer:
xmColumnWidgetClass

Instantiation:

```
widget = XmCreateColumn (parent, name,...)
or
widget = XtCreateWidget(name, xmColumnWidgetClass,...)
```

Functions/Macros:

```
XmCreateColumn(), XmColumnGetChildLabel()
```

Availability

OpenMotif 2.2 and later. (Provisional Widget).

Description

The Column widget displays its children stacked in a column, each with an optional associated label: labels appear in one column, and children in another. This is useful for displaying, for example, labeled data-entry fields. It can also display all label-child pairs in a horizontal orientation.This widget offers several constraint resources that allow specification of characteristics of the label, such as displaying text or a pixmap, alignment of text, font to use, and so forth. It also offers several resources for setting defaults for children that specify no specific values.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining

letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNdefaultEntryLabelAlignment	XmCAlignment	unsigned char	XmALIGNMENT_BEGINNING	CSG
XmNdefaultEntryLabelFontList	XmCFontList	XmFontList	dynamic	CSG
XmNdefaultFillStyle	XmCFillStyle	unsigned char	XmFILL_RAGGED	CSG
XmNdistribution	XmCDistribution	unsigned char	XmDISTRIBUTE_TIGHT	CSG
XmNitemSpacing	XmCItemSpacing	Dimension	2	CSG
XmNlabelSpacing	XmCLabelSpacing	Dimension	10	CSG
XmNorientation	XmCOrientation	unsigned char	XmVERTICAL	CSG

XmNdefaultEntryLabelAlignment
: Specifies the default XmNentryLabelAlignment to use when a child specifies no significant value. Resources that specify Alignment have values of XmALIGNMENT_BEGINNING, XmALIGNMENT_CENTER, XmALIGNMENT_END, and XmALIGNMENT_ UNSPECIFIED. Valid string values that can be used in a resources file are: alignment_ unspecified, unspecified, alignment_beginning, beginning, alignment_center, center, alignment_end, end.

XmNdefaultEntryLabelFontList
: Specifies the default XmNentryLabelFontList to use when a child specifies no significant value. If unspecified, uses XmNlabelFontList resource of the XmBulletinBoard.

XmNdefaultFillStyle
: Specifies the default XmNfillStyle to use when a child specifies no significant value.

XmNdistribution
: Specifies whether the spacing between each pair of rows should be increased equally (XmDISTRIBUTE_ SPREAD) or remain constant

(XmDISTRIBUTE_TIGHT) when the column isresized vertically to be larger than its natural size. This resource has no effect if any child has its

XmNitemSpacing
Specifies the spacing between each pair of rows (in vertical orientation) or between pairs of labels and children (in horizontal orientation).

XmNlabelSpacing
Specifies the spacing between the column containing the labels and the column containing the XmColumn's children.

XmNorientation
Specifies the layout direction of the XmColumn. When XmVERTICAL, the widgets and their associated labels are laid out in two vertical columns. When XmHORIZONTAL, the widgets and their associated labels are laid out in a single row.

Inherited Resources

Column inherits behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass.

Resource	Inherited from	Resource	Inherited from
XmNentryLabelAlignment	XmBulletinBoard	XmNuserData	XmManager
XmNentryLabelFontList	XmBulletinBoard	XmNchildren	Composite
XmNentryLabelPixmap	XmBulletinBoard	XmNinsertPosition	Composite
XmNentryLabelString	XmBulletinBoard	XmNnumChildren	Composite
XmNentryLabelType	XmBulletinBoard	XmNaccelerators	Core
XmNfillStyle	XmBulletinBoard	XmNancestorSensitive	Core
XmNshowEntryLabel	XmBulletinBoard	XmNbackground	Core
XmNstretchable	XmBulletinBoard	XmNbackgroundPixmap	Core
XmNbottomShadowColor	XmManager	XmNborderColor	Core
XmNbottomShadowPixmap	XmManager	XmNborderPixmap	Core
XmNforeground	XmManager	XmNborderWidth	Core
XmNhelpCallback	XmManager	XmNcolormap	Core
XmNhighlightColor	XmManager	XmNdepth	Core
XmNhighlightPixmap	XmManager	XmNdestroyCallback	Core
XmNinitialFocus	XmManager	XmNheight	Core
XmNlayoutDirection	XmManager	XmNinitialResourcesPersistent	Core

Resource	Inherited from	Resource	Inherited from
XmNnavigationType	XmManager	XmNmappedWhenManaged	Core
XmNpopupHandlerCall-back	XmManager	XmNscreen	Core
XmNshadowThickness	XmManager	XmNsensitive	Core
XmNstringDirection	XmManager	XmNtranslations	Core
XmNtopShadowColor	XmManager	XmNwidth	Core
XmNtopShadowPixmap	XmManager	XmNx	Core
XmNtraversalOn	XmManager	XmNy	Core
XmNunitType	XmManager		

Translations

XmColumn inherits translations from XmBulletinBoard.

See Also

`XmButtonBox, XmCreateObject(1), XmIconBox, Compos-ite(2), Constraint(2), Core(2), XmManager(2), XmRow-Column.`

Name

XmComboBox widget class – a composite widget which combines a text widget with a list of choices.

Synopsis

Public Header:
<Xm/ComboBox.h>

Class Name:
XmComboBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmComboBox

Class Pointer:
xmComboBoxWidgetClass

Instantiation:
widget = `XmCreateComboBox` (parent, name,...)
or
widget = `XmCreateDropDownComboBox` (parent, name,...)
or
widget = `XmCreateDropDownList` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmComboBoxWidgetClass,...)

Functions/Macros:
`XmComboBoxAddItem`(), `XmComboBoxDeletePos`(), `XmComboBoxSelectItem`(),
`XmComboBoxSetItem`(), `XmComboBoxUpdate`(), `XmCreateComboBox`(),
`XmCreateDropDownComboBox`(), `XmCreateDropDownList`(), `XmIsComboBox`()

Availability

Motif 2.0 and later.

Description

ComboBox is a composite widget that combines both text entry and scrolled list selection. The ComboBox can be configured in various ways, depending on whether the text field is to be editable, and whether the scrolled list is to be permanently visible. The text field contains the currently selected item; an item selected from the list is automatically placed into the text field. Whether entering data into the text field, or selecting from the list, the user can select only one item at any given time. Data typed directly into the text field does not, however, automatically select any matching item in the list.

By default, both the text field and the list are fully visible, and the list is placed underneath the text widget. The user can either enter characters directly into the text field, or select an item from the list. Alternatively, the ComboBox widget can be configured such that the list is hidden until required. In this case, the ComboBox widget draws an arrow button adjacent to the text field. Clicking on the arrow displays the hidden list underneath the text field. Selecting an item from the displayed popup list automatically pops down the list.

The XmNcomboBoxType resource configures the type of the ComboBox. You may specify one of the following types: combo box (XmCOMBO_BOX), drop-down combo box (XmDROP_DOWN_COMBO_BOX), or drop-down list (XmDROP_DOWN_LIST). The combo box type has a permanently visible list, and the text field is editable. A drop-down combo box has an editable text field, the list is hidden, and the arrow button is drawn. A drop-down list is identical to a drop-down combo box except that the text field is not editable: the user must select from the list to change the selection.

If the ComboBox widget has a non-editable text field, any characters typed do not appear directly in the text field. Instead, any characters entered may be compared against items in the list, depending on the value of the XmNmatchBehavior resource. When enabled, and the list has the input focus, characters typed are compared against the first character of each item in the list. If a match is found, the matched list item is automatically selected. Subsequently typing the same character progresses cyclically through the list to find any further matching item.

The position of the drawn arrow button relative to the text field depends upon the XmNlayoutDirection resource of the Shell ancestor of the ComboBox widget. If XmNlayoutDirection is XmLEFT_TO_RIGHT, the arrow button is drawn to the right of the text field. Otherwise, a value XmRIGHT_TO_LEFT draws the arrow button to the left.

Traits

ComboBox uses the XmQTaccessTextual and XmQTspecifyRenderTable traits.

New Resources

ComboBox defines the following resources:

Name	Class	Type	Default	Access
XmNarrowSize	XmCArrowSize	Dimension	dynamic	CSG
XmNarrowSpacing	XmCArrowSpacing	Dimension	dynamic	CSG
XmNcolumns	XmCColumns	short	dynamic	CSG
XmNcomboBoxType	XmCComboBoxType	unsigned char	XmCOMBO_BOX	CG
XmNfontList	XmCFontList	XmFontList	NULL	CSG
XmNhighlightThickness	XmCHighlightThickness	Dimension	dynamic	CSG
XmNitemCount	XmCItemCount	int	dynamic	CSG
XmNitems	XmCItems	XmStringTable	dynamic	CSG
XmNlist	XmCList	Widget	dynamic	G
XmNmarginHeight	XmCMarginHeight	Dimension	2	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	2	CSG
XmNmatchBehavior	XmCMatchBehavior	unsigned char	dynamic	CSG
XmNpositionMode	XmCPositionMode	XtEnum	XmZERO_BASED	CG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNselectedItem	XmCSelectedItem	XmString	NULL	CSG
XmNselectedPosition	XmCSelectedPosition	int	0	CSG
XmNtextField	XmCTextField	Widget	dynamic	G
XmNvisibleItemCount	XmCVisibleItemCount	int	10	CSG

XmNarrowSize

The size of the drawn arrow button used to display the hidden list. The default size depends upon the size of the text field, and the size of the ComboBox widget. Attempting to change the arrow size may result in a geometry request to the parent of the ComboBox widget. If this request is refused, the arrow size is set to the maximum size that fits into the space allowed by the parent of the ComboBox widget.

XmNarrowSpacing

The horizontal spacing, in pixels, between the drawn arrow button and the text field. The default value is calculated from the value of the XmNmarginWidth resource.

XmNcolumns

Specifies the number of columns in the text field. The default value is that of XmTextField.

XmNcomboBoxType
Specifies the type of the ComboBox. Possible values:

```
XmCOMBO_BOX                 /* visible list; editable text field     */
XmDROP_DOWN_COMBO_BOX       /* hidden list; editable text field      */
XmDROP_DOWN_LIST            /* hidden list; non-editable text field */
```

XmNfontList
The font list used for the items in the list and text field. In Motif 2.0 and later, the XmFontList is obsolete as a data type, and the resource is maintained for backwards compatibility through an implementation as an XmRenderTable. The XmNrenderTable resource is the preferred method of specifying appearance. If both a render table and font list are specified, the render table takes precedence. The default font list is taken from the default render table.

XmNhighlightThickness
The thickness of the highlighting rectangle. In Motif 2.0, the default is 2. In Motif 2.1 and later, the default depends upon the value of the XmDisplay resource XmNenableThinThickness: if True, the default is 1, otherwise 2.

XmNitemCount
The total number of items. If unspecified, the value is taken from the internal list. The ComboBox widget updates this resource every time a list item is added or removed through the ComboBox convenience functions.

XmNitems
A pointer to an array of compound strings, representing the items to display in the list. A call to XtGetValues() returns the actual list items, not a copy. Applications should not directly free any items fetched in this manner. If the resource is unspecified, the value is taken from the internal list. The ComboBox widget updates this resource every time a list item is added or removed through the ComboBox convenience functions.

XmNlist
The list widget created by the ComboBox. Applications may not change the value of this resource, but may fetch the value to perform required operations on the internal list.

XmNmarginHeight
The minimum spacing between the ComboBox top or bottom edge and the child list and text field widgets.

XmNmarginWidth
The minimum spacing between the ComboBox left or right edge and the child list and text field widgets.

XmNmatchBehavior

Determines whether matching behavior is enabled, where characters typed are compared against items in the list. Possible values:

```
XmNONE                  /* No match behavior     */
XmQUICK_NAVIGATE        /* Match behavior enabled */
```

The value XmQUICK_NAVIGATE may only be specified if the XmNcomboBoxType resource has value XmDROP_DOWN_LIST. XmQUICK_NAVIGATE is the default when XmNcomboBoxType is XmDROP_DOWN_LIST. Otherwise, XmNONE is the default.

XmNpositionMode

Specifies the way in which the position of the selected item is reported in callbacks, and controls the initial index value of the XmNselectedPosition resource. Possible values:

```
XmZERO_BASED            /* first item in list is position zero */
XmONE_BASED             /* first item in list is position one */
```

A value of XmZERO_BASED configures callback data on the XmNselectionCallback list such that the item_position element of the XmComboBoxCallbackStruct is indexed from zero: selecting the first list item has item_position set to zero, selecting the second item has item_position as one, and so forth. Similarly, fetching the XmNselectedPosition resource when the first item in the list is selected will return the value zero.

A value of XmONE_BASED sets the item_position element such that selecting the first item in the list is reported at position one, the second item is reported at position two, and so on. By analogy, fetching the XmNselectedPosition resource when the first item in the list is selected will return the value one.

In all cases, changes to the text field are reported with item_position set to zero.

A XmNpositionMode of XmONE_BASED therefore makes it easier to distinguish between text field and list selection, since item_position set to zero is not ambiguous.

The ComboBox convenience functions for adding, deleting, or selecting items in the list are unaffected by the value of this resource: these functions always assume that the first item is at position one.

This resource is provided for CDE compatibility. In particular, setting the value to XmZERO_BASED makes the ComboBox selection behavior consistent with that of the DtComboBox widget.

XmNrenderTable
: Specifies the render table for the ComboBox, and is used in both the text field and list children. If NULL, this is inherited from the nearest ancestor that has the XmQTspecifyRenderTable trait. The BulletinBoard, VendorShell, and MenuShell widgets and derived classes set this trait.
The render table resource takes precedence over any specified XmNfontList.

XmNselectedItem
: A compound string representing the currently selected item contained within the ComboBox text field.

XmNselectedPosition
: Identifies the index of the XmNselectedItem in the list. The interpretation of the index depends upon the value of the XmNpositionMode resource. If XmNpositionMode is XmZERO_BASED, a XmNselectedPosition value of 0 (zero) indicates that the first list item is selected, a value of 1 indicates the second item is selected, and so forth. If XmNpositionMode is XmONE_BASED, the value 1 indicates that the first list item is selected, the value 2 indicates the second list item, and so on, with value 0 (zero) indicating that no list item is selected.

XmNtextField
: The text field widget created by the ComboBox. Applications may not change the value of this resource, but may fetch the value to perform required operations on the internal text field.

XmNvisibleItemCount
: The number of items to display in the work area of the list. If specified, this resource overrides the XmNvisibleItemCount resource of the internal list widget. The resource may affect the height of the list widget, and hence the ComboBox itself, depending upon whether the list is permanently visible. If the XmNvisibleItemCount value is less than the number of items in the list, the list is automatically configured with a vertical ScrollBar.

Callback Resources

ComboBox defines the following callback resources:

Callback	Reason Constant
XmNselectionCallback	XmCR_SELECT

XmNselectionCallback
: List of callbacks that are called when a selection occurs in the ComboBox widget.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called        */
    XEvent      *event;         /* points to event structure that triggered callback */
    XmString    item_or_text;   /* the selected item                              */
    int         item_position;  /* the index of the item in the list              */
} XmComboBoxCallbackStruct;
```

The item_or_text element is a compound string representing the ComboBox selected item. It points to allocated memory that is reclaimed after the callbacks on the XmNselectionCallback list have returned. If the item is to be cached, the application should copy the item using XmStringCopy().

The item_position element specifies the index of the selected item within the XmNitems array of the list. The interpretation of the value will depend upon the XmNpositionMode resource of the ComboBox widget. With XmNpositionMode set to XmONE_BASED, an item_position of 1 refers to the first item in the list, and an item_position of zero indicates that the selected item has been entered directly into the text field (no list selection). With a XmNpositionMode of XmZERO_BASED, an item_position of zero could either mean that the first item in the list is selected, or that the selection is from direct text entry, and an item_position of 1 refers to the second list item.

Default Resource Values

A ComboBox sets the following default values for the scrolled list resources:

Name	Default
XmNborderWidth	0
XmNhighlightThickness	dynamic
XmNlistSizePolicy	XmVARIABLE
XmNnavigationType	XmNONE
XmNrenderTable	dynamic
XmNselectionPolicy	XmBROWSE_SELECT
XmNspacing	0
XmNtraversalOn	dynamic
XmNvisualPolicy	XmVARIABLE

If the ComboBox is a drop down list, XmNcursorPositionVisible and XmNeditable are set to False, and the XmNshadowThickness is set to zero. Otherwise, XmNcursorPositionVisible and XmNeditable are set True, and XmNeditMode is set to XmSINGLE_LINE_EDIT.

Inherited Resources

ComboBox inherits the resources shown below. The resources are listed alphabetically, along with the superclass that defines them. ComboBox sets the default values of XmNmarginWidth, and XmNmarginHeight to 2, and XmNnavigationType to XmSTICKY_TAB_GROUP. The default value of XmNborderWidth is reset to 0 (zero) by Manager. The default values for XmNhighlightThickness and XmNshadowThickness depend upon the XmDisplay XmNenableThinThickness resource: if True, the default is 1, otherwise 2.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Widget Hierarchy

The ComboBox creates the text field with the name Text, and the scrolled list is named List. If the ComboBox is of a drop down type, an XmGrabShell named GrabShell is created as parent to the scrolled list.

Translations

The translations for ComboBox include those of XmManager.

Event	Action
BSelect Press	CBArmAndDropDownList()
BSelect Release	CBDisarm()

ComboBox places the following translations upon the list:

Event	Action
KDown	CBDropDownList()
KUp	CBDropDownList()
KCancel	CBCancel()
KActivate	CBActivate()
MShift KBeginData	CBListAction(ListBeginData)
MShift KEndData	CBListAction(ListEndData)
KPageUp	CBListAction(ListPrevPage)
KPageDown	CBListAction(ListNextPage)

ComboBox places the following translations upon the text field:

Event	Action
<FocusOut>	CBTextFocusOut()

Action Routines

ComboBox defines the following action routines:

CBArmAndDropDownList()
: If the mouse is over the drawn arrow button, draws the arrow button as though selected, and posts the drop-down list.

CBDisarm()
: Draws the arrow button in unselected state.

CBDropDownList()
: Posts the drop-down list

CBFocusIn()
: Draws focus highlighting around the ComboBox widget.

CBFocusOut()
: Erases focus highlighting around the ComboBox widget. If the text field has changed, invokes the list of callbacks specified by XmNselectionCallback.

CBCancel()
: Pops down the drop-down list, and draws the arrow button in unselected state.

CBActivate()
: Fetches the value from the text field. If the XmNcomboBoxType is XmCOMBO_BOX, invokes the list of callbacks specified by XmNdefaultActionCallback for the internal list, passing the text field value as the selected item within the callback data. Regardless of XmNcomboBoxType, if the value matches an item within the list, the list item is selected, otherwise all list items are deselected. Lastly, the list of callbacks specified by XmNselectionCallback is invoked.

CBListAction(type)
: A generic action to perform operations on the internal list. The action type may be one of Up, Down, ListPrevPage, ListNextPage, ListBeginData, or ListEndData. The types Up and Down simply select the relevant item in the list in the required direction relative to the currently selected item. The remaining types directly invoke the ListPrevPage, ListNextPage, ListBeginData, or ListEndData actions of the list. The types Up and Down differ from the corresponding List actions in that the ComboBox actions will wrap around the items in the internal list.

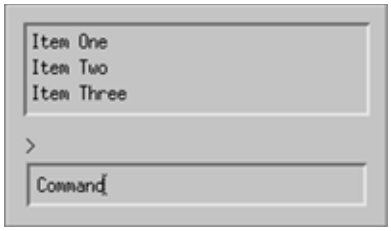
CBTextFocusOut()
: Turns off text field cursor blinking.

See Also

`XmComboBoxAddItem`(1), `XmComboBoxDeletePos`(1), `XmComboBoxSelectItem`(1), `XmComboBoxSetItem`(1), `XmComboBoxUpdate`(1), `XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `DropDown`(2), `XmManager`(2).

Name

XmCommand widget class – a composite widget for command entry.



Synopsis

Public Header:
<Xm/Command.h>

Class Name:
XmCommand

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmBulletinBoard → XmSelectionBox → XmCommand

Class Pointer:
xmCommandWidgetClass

Instantiation:
widget = `XmCreateCommand` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmCommandWidgetClass,...)

Functions/Macros:
`XmCommandAppendValue`(), `XmCommandError`(), `XmCommandGetChild`(),
`XmCommandSetValue`(), `XmCreateCommand`(), `XmIsCommand`()

Description

Command is a composite widget that handles command entry by providing a prompt, a command input field, and a history list region. Many of the Command widget's new resources are in fact renamed resources from SelectionBox.

New Resources

Command defines the following resources:

Name	Class	Type	Default	Access
XmNcommand	XmCTextString	XmString	NULL	CSG
XmNhistoryItems	XmCItems	XmStringTable	NULL	CSG

Name	Class	Type	Default	Access
XmNhistoryItemCount	XmCItemCount	int	0	CSG
XmNhistoryMaxItems	XmCMaxItems	int	100	CSG
XmNhistoryVisibleItemCount	XmCVisibleItemCount	int	dynamic	CSG
XmNpromptString	XmCPromptString	XmString	dynamic	CSG

XmNcommand
The text currently displayed on the command line. Synonymous with the XmNtextString resource in SelectionBox. XmNcommand can be changed using the routines `XmCommandSetValue`() and `XmCommandAppendValue`().

XmNhistoryItems
The items in the history list. Synonymous with the XmNlistItems resource in SelectionBox. A call to `XtGetValues`() returns the actual list items (not a copy), so don't have your application free these items.

XmNhistoryItemCount
The number of strings in XmNhistoryItems. Synonymous with the XmNlistItemCount resource in SelectionBox.

XmNhistoryMaxItems
The history list's maximum number of items. When this number is reached, the first history item is removed before the new command is added to the list.

XmNhistoryVisibleItemCount
The number of history list commands that will display at one time. Synonymous with the XmNvisibleItemCount resource in SelectionBox.

XmNpromptString
The command-line prompt. Synonymous with the XmNselectionLabelString resource in SelectionBox.

Callback Resources

Command defines the following callback resources:

Callback	Reason Constant
XmNcommandEnteredCallback	XmCR_COMMAND_ENTERED
XmNcommandChangedCallback	XmCR_COMMAND_CHANGED

XmNcommandChangedCallback
List of callbacks that are called when the value of the command changes.

XmNcommandEnteredCallback
List of callbacks that are called when a command is entered in the widget.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;     /* the reason that the callback was called        */
    XEvent      *event;     /* points to event structure that triggered callback */
    XmString    value;      /* the string contained in the command area       */
    int         length;     /* the size of this string                        */
} XmCommandCallbackStruct;
```

Inherited Resources

Command inherits the resources shown below. The resources are listed alphabetically, along with the superclass that defines them. Command sets the default values of XmNautoUnmanage and XmNdefaultPosition to False, XmNdialogType to XmDIALOG_COMMAND, and XmNlistLabelString to NULL. In versions of Motif prior to 2.1.10, XmNresizePolicy is reset to XmRESIZE_NONE.

In Motif 2.1.10 and later, it is reset to XmRESIZE_ANY: this undocumented change is a bug which persists in Motif 2.1.20. The default value of XmNborderWidth is reset to 0 by Manager. BulletinBoard sets the value of XmNinitialFocus to XmNdefaultButton and resets the default XmNshadowThickness from 0 to 1 if the Command widget is a child of a DialogShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNlistItemCount	XmSelectionBox
XmNallowOverlap	XmBulletinBoard	XmNlistItems	XmSelectionBox
XmNancestorSensitive	Core	XmNlistLabelString	XmSelectionBox
XmNapplyCallback	XmSelectionBox	XmNlistVisibleItemCount	XmSelectionBox
XmNapplyLabelString	XmSelectionBox	XmNmapCallback	XmBulletinBoard
XmNautoUnmanage	XmBulletinBoard	XmNmappedWhenManaged	Core
XmNbackground	Core	XmNmarginHeight	XmBulletinBoard
XmNbackgroundPixmap	Core	XmNmarginWidth	XmBulletinBoard
XmNborderColor	Core	XmNminimizeButtons	XmSelectionBox
XmNborderPixmap	Core	XmNmustMatch	XmSelectionBox
XmNborderWidth	Core	XmNnavigationType	XmManager
XmNbottomShadowColor	XmManager	XmNnoMatchCallback	XmSelectionBox
XmNbottomShadowPixmap	XmManager	XmNnoResize	XmBulletinBoard
XmNbuttonFontList	XmBulletinBoard	XmNnumChildren	Composite
XmNbuttonRenderTable	XmBulletinBoard	XmNokCallback	XmSelectionBox
XmNcancelButton	XmBulletinBoard	XmNokLabelString	XmSelectionBox

Resource	Inherited From	Resource	Inherited From
XmNcancelCallback	XmSelectionBox	XmNpopupHandlerCallback	XmManager
XmNcancelLabelString	XmSelectionBox	XmNresizePolicy	XmBulletinBoard
XmNchildren	Composite	XmNscreen	Core
XmNchildPlacement	XmSelectionBox	XmNselectionLabelString	XmSelectionBox
XmNcolormap	Core	XmNsensitive	Core
XmNdefaultButton	XmBulletinBoard	XmNshadowThickness	XmManager
XmNdefaultPosition	XmBulletinBoard	XmNshadowType	XmBulletinBoard
XmNdepth	Core	XmNstringDirection	XmManager
XmNdestroyCallback	Core	XmNtextAccelerators	XmSelectionBox
XmNdialogStyle	XmBulletinBoard	XmNtextColumns	XmSelectionBox
XmNdialogTitle	XmBulletinBoard	XmNtextFontList	XmBulletinBoard
XmNdialogType	XmSelectionBox	XmNtextRenderTable	XmBulletinBoard
XmNfocusCallback	XmBulletinBoard	XmNtextString	XmSelectionBox
XmNforeground	XmManager	XmNtextTranslations	XmBulletinBoard
XmNheight	Core	XmNtopShadowColor	XmManager
XmNhelpCallback	XmManager	XmNtopShadowPixmap	XmManager
XmNhelpLabelString	XmSelectionBox	XmNtranslations	Core
XmNhighlightColor	XmManager	XmNtraversalOn	XmManager
XmNhighlightPixmap	XmManager	XmNunitType	XmManager
XmNinitialFocus	XmManager	XmNunmapCallback	XmBulletinBoard
XmNinitialResourcesPersistent	Core	XmNuserData	XmManager
XmNinsertPosition	Composite	XmNwidth	Core
XmNlabelFontList	XmBulletinBoard	XmNx	Core
XmNlabelRenderTable	XmBulletinBoard	XmNy	Core
XmNlayoutDirection	XmManager		

Translations

The translations for Command are inherited from XmSelectionBox.

Action Routines

Command defines the following action routines:

SelectionBoxUpOrDown(flag)

Selects a command from the history list, replaces the current command-line text with this list item, and invokes the callbacks specified by XmNcommandChangedCallback. The value of flag determines which history list command is selected. With a flag

value of 0, 1, 2, or 3, this action routine selects the list's previous, next, first, or last item, respectively.

Additional Behavior

Command has the following additional behavior:

MAny KCancel
: The event is passed to the parent if it is a manager widget.

KActivate
: In the Text widget, invokes the XmNactivateCallback callbacks, appends the text to the history list, and invokes the XmNcommandEnteredCallback callbacks.

<Key>
: In the Text widget, any keystroke that changes text invokes the XmNcommandChangedCallback callbacks.

KActivate or <DoubleClick>
: In the List widget, invokes the XmNdefaultActionCallback callbacks, appends the selected item to the history list, and invokes the XmNcommandEnteredCallback callbacks.

<FocusIn>
: Invokes the XmNfocusCallback callbacks.

<MapWindow>
: If the widget is a child of a DialogShell, invokes the XmNmapCallback callbacks when the widget is mapped.

<UnmapWindow>
: If the widget is a child of a DialogShell, invokes the XmNunmapCallback callbacks when the widget is unmapped.

See Also

`XmCommandAppendValue`(1), `XmCommandError`(1), `XmCommandGetChild`(1), `XmCommandSetValue`(1), `XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmBulletinBoard`(2), `XmManager`(2), `XmSelectionBox`(2).

Name

XmContainer widget class – a widget which controls the layout and selection of a set of items

Synopsis

Public Header:
<Xm/Container.h>

Class Name:
XmContainer

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmContainer

Class Pointer:
xmContainerWidgetClass

Instantiation:
widget = `XmCreateContainer` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmContainerWidgetClass,...)

Functions/Macros:
`XmContainerCopy`(), `XmContainerCopyLink`(), `XmContainerCut`(), `XmContainerGetItemChild`(), `XmContainerPaste`(), `XmContainerPasteLink`(), `XmContainerRelayout`(), `XmContainerReorder`(), `XmCreateContainer`()

Availability

Motif 2.0 and later.

Description

A Container is a constraint widget which controls the layout and selection of container items. Container is intended to provide an object-oriented view of the world: an application object can be represented in the Container as a container item. The user can subsequently manipulate the item by moving, copying, selecting, or deleting it, or perform drag and drop operations between applications using the item. New items can be dropped into the Container. At each stage, callbacks indicate to the application the operation performed upon each item, and hence the requested operation upon the object which it represents. The Container recognises as a container item any child widget which holds the XmQTcontainerItem trait. The IconGadget is the only standard Motif widget to hold this trait.

The Container provides three styles in which items can be displayed, specified through the XmNlayoutType resource. The resource can have the values XmOUTLINE, XmDETAIL, or XmSPATIAL.

The XmOUTLINE layout style provides a tree view onto the items, and is appropriate when application objects exist in a parent/child relationship to each other. The logical relationship between items is specified by setting the XmNentryParent constraint resource of an item to point to the logical parent. The order of items within the tree depends upon the XmNpositionIndex constraint value for each item. The Container draws connecting lines between items to indicate the relationships. The Container creates additional PushButtonGadgets which are used for folding and unfolding portions of the tree.

The XmDETAIL layout style gives a tabular format, where each row of the table represents an object, each cell within the row possibly representing a property of the object. The data attached to an object and displayed in the row is specified through the XmNdetail resources of the associated container item. Column headings can be specified through the XmNdetailColumnHeading resources of the Container.

The XmSPATIAL layout style provides generic layout, where the exact positioning of items is controlled through further resources. This can take the form of a grid layout, where items can span single or multiple cells of the grid, or a free format where items can be positioned at absolute x, y coordinates of the Container. At the simplest, the grid layout can be used to construct a general purpose icon box. The resources XmNspatialStyle, XmNspatialIncludeModel, and XmNspatialSnapModel control the positioning of items.

The Container controls the way in which items are selected, and provides selection notification. The widget supports single, browse, multiple, and extended selection. Selection of items within the Container can be performed by including them within a rubberband rectangle called a Marquee, which the user specifies using the mouse. Alternatively, selection can be performed by simply swiping the mouse over an item. The style of selection is specified through the XmNselectionTechnique resource.

The user can only move container items if the XmNlayoutType is XmSPATIAL.

Traits

Container holds the XmQTtransfer, XmQTtraversalControl, and XmQTcontainer traits, and uses the XmQTscrollFrame, XmQTcontainerItem, XmQTnavigator, XmQTspecifyRenderTable, and XmQTpointIn traits.

New Resources

Container defines the following resources:

Name	Class	Type	Default	Access
XmNautomaticSelection	XmCAutomaticSelection	unsigned char	XmAUTO_SELECT	CSG

Name	Class	Type	Default	Access
XmNcollapsedStatePixmap	XmCCollapsedStatePix-map	Pixmap	dynamic	CSG
XmNdetailColumnHeading	XmCDetailColumnHead-ing	XmStringTable	NULL	CSG
XmNdetailColumnHeading-Count	XmCDetailColumnHead-ingCount	Cardinal	0	CSG
XmNdetailOrder	XmCDetailOrder	Cardinal *	NULL	CSG
XmNdetailOrderCount	XmCDetailOrderCount	Cardinal	0	CSG
XmNdetailTabList	XmCDetailTabList	XmTabList	NULL	CSG
XmNentryViewType	XmCEntryViewType	unsigned char	XmANY_ICON	CSG
XmNexpandedStatePixmap	XmCExpandedStatePix-map	Pixmap	dynamic	CSG
XmNfontList	XmCFontList	XmFontList	dynamic	CSG
XmNlargeCellHeight	XmCCellHeight	Dimension	0	CSG
XmNlargeCellWidth	XmCCellWidth	Dimension	0	CSG
XmNlayoutType	XmCLayoutType	unsigned char	XmSPATIAL	CSG
XmNmarginHeight	XmCMarginHeight	Dimension	0	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	0	CSG
XmNoutlineButtonPolicy	XmCOutlineButtonPolicy	unsigned char	XmOUTLINE_BUTTON_PRESENT	CSG
XmNoutlineColumnWidth	XmCOutlineColumnWidth	Dimension	0	CSG
XmNoutlineIndentation	XmCOutlineIndentation	Dimension	40	CSG
XmNoutlineLineStyle	XmCOutlineLineStyle	unsigned char	XmSINGLE	CSG
XmNprimaryOwnership	XmCPrimaryOwnership	unsigned char	XmOWN_POSSIBLE_MULTIPLE	CSG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNselectColor	XmCSelectColor	Pixel	XmREVERSED_GROUND_COLORS	CSG
XmNselectedObjectCount	XmCSelectedObjectCount	int	0	SG
XmNselectedObjects	XmCSelectedObjects	WidgetList	NULL	SG
XmNselectionPolicy	XmCSelectionPolicy	unsigned char	XmEXTENDED_SELECT	CSG
XmNselectionTechnique	XmCSelectionTechnique	unsigned char	XmTOUCH_OVER	CSG
XmNsmallCellHeight	XmCCellHeight	Dimension	0	CSG
XmNsmallCellWidth	XmCCellWidth	Dimension	0	CSG
XmNspatialIncludeModel	XmCSpatialIncludeModel	unsigned char	XmAPPEND	CSG
XmNspatialResizeModel	XmCSpatialResizeModel	unsigned char	XmGROW_MINOR	CSG

Name	Class	Type	Default	Access
XmNspatialSnapModel	XmCSpatialSnapModel	unsigned char	XmNONE	CSG
XmNspatialStyle	XmCSpatialStyle	unsigned char	XmGRID	CSG

XmNautomaticSelection
Specifies whether selection callbacks are invoked immediately and each time that an item is selected, or whether callbacks are invoked when the user has completed the selection action. Possible values:

```
XmAUTO_SELECT       /* callbacks called immediately on selection    */
XmNO_AUTO_SELECT    /* callbacks delayed until user action completed
*/
```

XmNcollapsedStatePixmap
Specifies the pixmap to display on the PushButtonGadget to indicate that logical child items are folded (hidden) within an XmOUTLINE layout. The resource has no effect unless resource XmNoutlineButtonPolicy is XmOUTLINE_BUTTON_PRESENT. Otherwise, if the resource is unspecified, a default pixmap with an upwards pointing arrow is displayed.

XmNdetailColumnHeading
Specifies an XmString array to use as column headings in an XmDETAIL layout. No column headings are displayed if the value is NULL.

XmNdetailColumnHeadingCount
Specifies the length of the array associated with the XmNdetailColumnHeading resource.

XmNdetailOrder
Specifies an array of Cardinal values that represents which column, and in which order, the detail data associated with container items is to be displayed. The resource has no effect unless XmNlayoutType is XmDETAIL. If NULL, the XmNdetailOrderCount resource is used to determine the column detail data associated with each item.

XmNdetailOrderCount
Specifies the length of the array associated with the XmNdetailOrder resource. If XmNdetailOrder is NULL, and XmNdetailOrderCount is not zero, each container item displays any detail information in order starting from column 1, up to the value of XmNdetailOrderCount. Otherwise, with a value of zero, a default algorithm inspects the XmQTcontainerItem trait of each item to determine the columnar data.

XmNdetailTabList
Specifies an XmTabList which indicates the start of each column in an XmDETAIL layout. If NULL, the Container calculates a default XmTabList.

XmNentryViewType
Specifies the view type for all Container children. If the value is XmANY_ICON, then the XmQTcontainerItem trait of each child specifies the individual view type. Possible values:

```
XmANY_ICON      /* children use their own view type    */
XmLARGE_ICON    /* all children forced to XmLARGE_ICON */
XmSMALL_ICON    /* all children forced to XmSMALL_ICON */
```

XmNexpandedStatePixmap
Specifies the pixmap to display on the PushButtonGadget to indicate that logical child items are unfolded (displayed). The resource has no effect unless XmNoutlineButtonPolicy is XmOUTLINE_BUTTON_PRESENT. Otherwise, if the resource is unspecified, a default pixmap with a downwards pointing arrow is displayed.

XmNfontList
The XmFontList is considered obsolete in Motif 2.0 and later, and has been subsumed into the XmRenderTable. Any specified XmNrenderTable resource takes precedence.

XmNlargeCellHeight
Specifies the height of a cell when the Container is using a grid layout. The resource is not used when XmNentryViewType is XmSMALL_ICON.

XmNlargeCellWidth
Specifies the width of a cell when the Container is using a grid layout. The resource is not used when XmNentryViewType is XmSMALL_ICON.

XmNlayoutType
Specifies the way in which the Container lays out children. Possible values:

```
XmOUTLINE    /* items are displayed in a tree arrangement          */
XmSPATIAL    /* items displayed according to XmNspatialStyle resource */
XmDETAIL     /* items displayed in tabular row/column format       */
```

XmNmarginHeight
Specifies the spacing at the top and bottom of the Container widget.

XmNmarginWidth
Specifies the spacing at the left and right of the Container widget.

XmNoutlineButtonPolicy
Specifies whether a PushButtonGadget, used for folding/unfolding items, is displayed with each container item that has logical children, specified by the XmNentryParent resource. The resource has no effect if XmNspatialStyle is not XmOUTLINE. Possible values:

```
XmOUTLINE_BUTTON_ABSENT    /* display fold/unfold buttons */
```

```
XmOUTLINE_BUTTON_PRESENT    /* no PushButtonGadget buttons */
```

XmNoutlineColumnWidth
Specifies the width of the first column within an XmDETAIL layout, and the preferred width of the Container within an XmOUTLINE layout. If zero, the Container will deduce a default value based upon the width of the widest item pixmap and the XmNoutlineIndentation resource.

XmNoutlineIndentation
Specifies an indentation for container items. The resource has no effect when XmNlayoutType[1] is XmSPATIAL.

XmNoutlineLineStyle
Specifies whether to draw connecting lines between container items in an XmOUTLINE or XmDETAIL layout. Possible values:

```
XmNO_LINE                   /* no line is drawn between items      */
XmSINGLE                    /* a line one pixel wide connects items
*/
```

XmNprimaryOwnership
Specifies whether the Container takes possession of the primary selection when the user makes a selection from the items within the widget. Possible values:

```
XmOWN_NEVER                 /* never own the primary selection
*/
XmOWN_ALWAYS                /* always own the primary selection
*/
XmOWN_MULTIPLE              /* own if more than one item is selected
*/
XmOWN_POSSIBLE_MULTIPLE     /* own if multiple selection possible
*/
```

XmNrenderTable
Specifies the render table that is used for all children of the Container. If NULL, the nearest ancestor holding the XmQTspecifyRenderTable trait is searched, using the XmLABEL_FONTLIST value.

XmNselectColor
Specifies a color which container item children can use to indicate selected state. In addition to allocated Pixel values, the constant XmDEFAULT_SELECT_COLOR specifies a color between the XmNbackground and XmNbottomShadowColor, XmHIGHLIGHT_COLOR makes the select color the same as the XmNhighlightColor value, and

1.Erroneously listed as XmNlayoutStyle in 2nd Edition.

XmREVERSED_GROUND_COLORS makes the XmNselectColor the same as the XmNforeground, using the XmNbackground color to render any text.

XmNselectedObjectCount
Specifies the number of widgets in the array of selected container items, represented by the XmNselectedObjects resource.

XmNselectedObjects
Specifies an array of widgets representing the set of container items currently selected.

XmNselectionPolicy
Specifies the way in which container items can be selected. Possible values:

```
XmSINGLE_SELECT         /* only one selected item permitted
*/
XmBROWSE_SELECT         /* as above, except items are selected by
dragging */
XmMULTIPLE_SELECT       /* items in contiguous range are selecta-
ble      */
XmEXTENDED_SELECT       /* items in discontinuous range are selecta-
ble    */
```

XmNselectionTechnique
Specifies the way in which items are selected. Possible values:

```
XmMARQUEE                       /* items must be wholly enclosed
within Marquee          */ XmMARQUEE_EXTEND_START/* includes item
containing Marquee start coordinate     */ XmMARQUEE_EXTEND_BOTH/*
includes items containing Marquee start/end coordinates */ XmTOUCH_ONLY/*
select items between start and end location         */
XmTOUCH_OVER                    /* select only items the mouse passes
through         */
```

XmNsmallCellHeight
Specifies the height of a cell when the Container spatial style is XmGRID, and the XmNentryViewType resource is XmSMALL_ICON.

XmNsmallCellWidth
Specifies the width of a cell when the Container spatial style is XmGRID, and the XmNentryViewType resource is XmSMALL_ICON.

XmNspatialIncludeModel
Specifies the layout of an item within a grid XmSPATIAL layout type, when the item is managed. Possible values:

```
XmAPPEND        /* place after the last occupied cell     */
                /* according to XmNlayoutDirection */
XmCLOSEST       /* place in the free cell */
```

```
                        /* nearest the x, y coordinates of the item */
    XmFIRST_FIT         /* place the item in the first free cell   */
                        /* according to XmNlayoutDirection */
```

XmNspatialResizeModel

Specifies how the Container will attempt to resize itself when there is insufficient space to contain a new item. The resource only has effect within a grid XmSPATIAL layout type. The definition of XmGROW_MAJOR and XmGROW_MINOR depend upon the value of the XmNlayoutDirection resource. The major dimension is width when the XmNlayoutDirection is horizontally oriented, and height when the direction is vertically oriented. Similarly, the minor dimension is height when the XmNlayoutDirection is horizontally oriented, and width when the direction is vertically oriented. Possible values:

```
XmGROW_BALANCED     /* request both width and height from parent */
XmGROW_MAJOR        /* request growth in the major dimension     */
XmGROW_MINOR        /* request growth in the minor dimension     */
```

XmNspatialSnapModel

Specifies how the Container will position an item within a cell, when the XmNlayoutType is XmSPATIAL. A value of XmSNAP_TO_GRID positions the item at the upper left or upper right of the cell, depending upon the value of XmNlayoutDirection. A value of XmNONE positions the item depending upon the value of XmNx, XmNy: if these fall outside the cell, then layout is performed according to the XmSNAP_TO_GRID method. A value of XmCENTER centers the item.

XmNspatialStyle

Specifies the layout of container items, when the XmNlayoutType is XmSPATIAL. Possible values:

```
XmCELLS     /* grid layout of same-sized cells. an item can occupy many cells
*/
XmGRID      /* grid layout of same-sized cells. an item can occupy one cell
*/
XmNONE      /* lay out according to XmNx, XmNy resources                  */
```

New Constraint Resources

Container defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNentryParent	XmCEntryParent	Widget	NULL	CSG
XmNoutlineState	XmCOutlineState	unsigned char	XmCOLLAPSED	CSG
XmNpositionIndex	XmCPositionIndex	int	XmLAST_POSITION	CSG

XmNentryParent
Specifies a logical parent for the item. The root of a hierarchy has the value NULL. Used when the XmNlayoutType is XmOUTLINE or XmDETAIL.

XmNoutlineState
Specifies whether to display logical child items when the XmNlayoutType is XmOUTLINE or XmDETAIL. Possible values:

```
XmCOLLAPSED     /* does not display child items */
XmEXPANDED      /* displays child items         */
```

XmNpositionIndex
Specifies the order of the item within the Container, when XmNlayoutType is XmOUTLINE or XmDETAIL. Items are firstly ordered by XmNentryParent, and by XmNpositionIndex within those items sharing the same XmNentryParent. If unspecified, the highest such index for all other items sharing the same logical parent is calculated, and then incremented. If no other item shares the same logical parent, the default is zero.

Callback Resources

Container defines the following callback resources:

Callback	Reason Constant
XmNconvertCallback	XmCR_OK
XmNdefaultActionCallback	XmCR_DEFAULT_ACTION
XmNdestinationCallback	XmCR_OK
XmNoutlineChangedCallback	XmCR_COLLAPSED XmCR_EXPANDED
XmNselectionCallback	XmCR_SINGLE_SELECT XmCR_BROWSE_SELECT XmCR_MULTIPLE_SELECT XmCR_EXTENDED_SELECT

XmNconvertCallback
List of callbacks called when a request is made to convert a selection.

XmNdefaultActionCallback
List of callbacks called when an item is double-clicked, or KActivate is pressed.

XmNdestinationCallback
List of callbacks called when the Container is the destination of a transfer.

XmNoutlineChangedCallback
List of callbacks called when a change is made to the XmNoutlineState value of an item.

XmNselectionCallback
List of callbacks called when an item is selected.

Callback Structure

Each callback on the XmNoutlineChangedCallback list is passed the following structure:

```
typedef struct
    int             reason;          /* the reason that the callback was called */
    XEvent          *event;          /* points to event that triggered callback */
    Widget          item;            /* container item associated with event    */
    unsigned char   new_outline_state;       /* the requested state */
} XmContainerOutlineCallbackStruct;
```

new_outline_state specifies an XmNoutlineState for item. The value may be changed within the callback to force a particular state.

Each callback on the XmNselectionCallback and XmNdefaultActionCallback list is passed the following structure:

```
typedef struct {
    int             reason;          /* the reason that the callback was called */
    XEvent          *event;          /* points to event that triggered callback */
    WidgetList      selected_items;          /* the list of selected items          */
    int             selected_item_count;     /* the number of selected items     */
    unsigned char       auto_selection_type;         /* type of selection event    */
} XmContainerSelectCallbackStruct;
```

selected_items is the array of container items selected by the event. The number of such items is specified by selected_item_count. auto_selection_type indicates the type of automatic selection event. If XmNautomaticSelection is False, auto_selection_type has the value XmAUTO_UNSET. Otherwise, the range of possible values is given by:

```
XmAUTO_BEGIN        /* event is the beginning of automatic selection    */
XmAUTO_CANCEL       /* current selection is cancelled                   */
XmAUTO_CHANGE       /* current selection differs from initial selection */
XmAUTO_MOTION       /* current selection is caused by button drag */
XmAUTO_NO_CHANGE    /* current selection same as the initial selection */
```

Convert callbacks are fully described within the sections covering the Uniform Transfer Model. See *XmTransfer*(1) for more details. For quick reference, a pointer to the following structure is passed to callbacks on the XmNconvertCallback list:

```
typedef struct {
    int             reason;          /* reason that the callback is invoked     */
    XEvent          *event;          /* points to event that triggered callback */
    Atom            selection;       /* requested conversion selection          */
    Atom            target;          /* the conversion target                   */
    XtPointer       source_data;     /* selection source information            */
    XtPointer       location_data;   /* information on data to be transferred */
    int             flags;           /* input status of the conversion          */
    XtPointer       parm;            /* parameter data for the target           *
    int             parm_format;     /* format of parameter data                */
    unsigned long   parm_length;     /* the number of elements                  */
                                     /*       in parameter data                 */
    Atom            parm_type;       /* the type of the parameter data     */
    int             status;          /* output status of the conversion         */
    XtPointer       value;           /* returned conversion data                */
    Atom            type;            /* type of conversion data returned        */
    int             format;          /* format of the conversion data           */
    unsigned long   length;     /* number of elements in conversion data */
} XmConvertCallbackStruct;
```

Destination callbacks are fully described within the sections covering the Uniform Transfer Model. For quick reference, a pointer to the following structure is passed to callbacks on the XmNdestinationCallback list:

```
typedef struct {
    int             reason;             /* reason that the callback is invoked     */
    XEvent          *event;             /* points to event that triggered callback */
    Atom            selection;          /* requested selection type, as an Atom    */
    XtEnum          operation;          /* the type of transfer requested          */
    int             flags;             /* whether destination and source are same */
    XtPointer       transfer_id;        /* unique identifier for the request       */
    XtPointer       destination_data;   /* information about the destination       */
    XtPointer       location_data;      /* information about the data              */
    Time            time;               /* time when transfer operation started    */
} XmDestinationCallbackStruct;
```

Inherited Resources

Container inherits the resources shown below. The resources are listed alphabetically, along with the superclass that defines them.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Widget Hierarchy

The PushButtonGadget children created by an outline style Container are all named OutlineButton.

Translations

The translations for Container include those of XmManager.

Event	Action
BSelect Press	ContainerHandleBtn1Down(ContainerBeginSelect, Copy)
BToggle Press	ContainerHandleBtn1Down(ContainerBeginToggle, Copy)
MLink BSelect Press	ContainerHandleBtn1Down(ContainerNoop, Link)
BExtend Press	ContainerHandleBtn1Down(ContainerBeginExtend Move)
BExtend Motion	ContainerHandleBtn1Motion(ContainerButtonMotion)
BSelect Release	ContainerHandleBtn1Up(ContainerEndSelect)
BToggle Release	ContainerHandleBtn1Up(ContainerEndToggle)
BExtend Release	ContainerHandleBtn1Up(ContainerEndExtend)

Event	Action
BTransfer Press	ContainerHandleBtn2Down(ContainerStartTransfer, Copy)
MLink BTransfer Press	ContainerHandleBtn2Down(ContainerStartTransfer, Link)
MMove BTransfer Press	ContainerHandleBtn2Down(ContainerStartTransfer, Move)
BTransfer Motion	ContainerHandleBtn2Motion(ContainerButtonMotion)
BTransfer Release	ContainerHandleBtn2Up(ContainerEndTransfer)
MShift KPrimaryCopy	ContainerPrimaryLink()
KPrimaryCopy	ContainerPrimaryCopy()
KPrimaryCut	ContainerPrimaryMove()
KCancel	ContainerCancel()
KExtend	ContainerExtend()
KSelect	ContainerSelect()
KSelectAll	ContainerSelectAll
KDeselectAll	ContainerDeselectAll()
KAddMode	ContainerToggleMode()
KActivate	ContainerActivate()
MShift KBeginData	ContainerExtendCursor(First)
MShift KEndData	ContainerExtendCursor(Last)
KBeginData	ContainerMoveCursor(First)
KEndData	ContainerMoveCursor(Last)
MCtrl KLeft	ContainerExpandOrCollapse(Left)
MCtrl KRight	ContainerExpandOrCollapse(Right)
MShift KUp	ContainerExtendCursor(Up)
MShift KDown	ContainerExtendCursor(Down)
MShift KLeft	ContainerExtendCursor(Left)
MShift KRight	ContainerExtendCursor(Right)
KUp	ContainerMoveCursor(Up)
KDown	ContainerMoveCursor(Down)
KLeft	ContainerMoveCursor(Left)
KRight	ContainerMoveCursor(Right)

Action Routines

Container defines the following action routines:

ContainerActivate()

Calls the procedures associated with the XmNdefaultActionCallback resource.

ContainerBeginExtend()

The action has no effect if the XmNlayoutType is XmSPATIAL, or if the XmNselectionPolicy is either XmSINGLE_SELECT or XmBROWSE_SELECT.

Otherwise, the location cursor is set to the object under the pointer, and if no object is there, or if there is no anchor, the action returns. Any items between the anchor and the location cursor are selected. Finally, if automatic selection is enabled, the list of callbacks specified by the XmNselectionCallback resource is invoked.

ContainerBeginSelect()

Single selection: if the object under the pointer is the anchor item, the selected state of the object is reversed. Otherwise, all items are deselected, the object at the pointer is made the anchor item, and the location cursor is set to it.

Browse selection: if the object under the pointer is not the anchor item, all items are deselected, the object is made the anchor item and selected, and the location cursor is set to it. If automatic selection is enabled, the list of callbacks specified by the XmNselectionCallback resource is invoked.

Multiple selection: sets the anchor item to the object under the pointer, and sets the location cursor to it. The selected state of the item is reversed. If the selection technique is XmTOUCH_OVER, and the anchor item is NULL, the Marquee start point is initialized. If automatic selection is enabled, the list of callbacks specified by the XmNselectionCallback resource is invoked.

Extended selection: as for multiple selection, except initially all items are deselected.

ContainerBeginToggle()

The action has no effect if the XmNlayoutType is XmSPATIAL, or if the XmNselectionPolicy is either XmSINGLE_SELECT or XmBROWSE_SELECT.

Multiple or Extended selection: the anchor item is set to the object under the pointer, and the location cursor is set to it. The selected state of the item is reversed. If automatic selection is enabled, the list of callbacks specified by the XmNselectionCallback resource is invoked. Lastly, if XmNselectionTechnique is XmMARQUEE_EXTEND_START or

XmMARQUEE_EXTEND_BOTH, the Marquee rectangle is drawn around the item.

ContainerButtonMotion()

The action has no effect if XmNselectionPolicy is XmSINGLE_SELECT.

Browse selection: if the action follows ContainerBeginExtend() *or ContainerBeginToggle()* action, or if the pointer is over the current anchor item, the routine simply returns. Otherwise, the selected state of the anchor item is reversed, the selected state of any item under the pointer is also reversed, and the anchor item is reset to point to it.

Multiple and extended selection: if a previous action has initiated the Marquee, the rectangle is redrawn around the start point and the current pointer location. The selected state of all items within the Marquee are set to that of the anchor item. For non-Marquee selection, in a spatial layout the selected state of the item under the pointer is reversed, and the anchor item is reset to it. In a non-spatial layout, the selected state of all items between the anchor item and the item under the pointer are set to match the selected state of the anchor item.

In all cases, if automatic selection is enabled, the list of callbacks specified by the XmNselectionCallback resource is invoke*d.*

ContainerCancel()

The selected state of all items reverts to the pre-selection state. If automatic selection is enabled, the list of callbacks specified by the XmNselectionCallback resource is invoked.

ContainerDeselectAll()

All items are deselected, and the callbacks on the XmNselection-Callback list are invoked.

ContainerEndExtend()

The action has no effect if XmNlayoutType is XmSPATIAL.

Multiple or Extended selection: the callbacks specified by XmNselectionCallback are invoked.

ContainerEndSelect()

Single selection: simply invokes the callbacks specified by the XmNselectionCallback *resource.*

Browse selection: if the pointer is not over the current anchor item, the selected state of the current anchor, and the selected state of any item under the pointer are reversed. Callbacks specified by XmNselectionCallback are invoked.

Multiple and extended selection: similar to the ContainerButtonMotion() action, except that the auto_selection_type element within XmNselectionCallback procedures is XmAUTO_CHANGE or XmAUTO_NO_CHANGE rather than XmAUTO_MOTION.

ContainerEndToggle()

The action has no effect if XmNselectionPolicy is XmSINGLE_SELECT or XmBROWSE_SELECT.

Multiple or extended selection: the procedure directly invokes the ContainerEndSelect() action.

ContainerEndTransfer()

If the current transfer operation is XmLINK, the ContainerPrimaryLink() action is called. If the transfer operation is XmMOVE, the procedure invokes ContainerPrimaryMove(). If the operation is XmCOPY, the ContainerPrimaryCopy() action is called.

ContainerExpandOrCollapse(type)

The action has no effect if layout type is XmSPATIAL().

Otherwise, the outline state of the container item which has the focus is changed. Possible values for type:

```
Collapse/* outline state set to XmCOLLAPSED */
Expand/* outline state set to XmEXPANDED */
Left/* depends upon layout direction    */
Right/* depends upon layout direction    */
```

If XmNlayoutDirection is XmLEFT_TO_RIGHT, Left is interpreted as XmCOLLAPSED, and Right as XmEXPANDED. This is reversed if the layout direction is XmRIGHT_TO_LEFT.

ContainerExtend()

The action has no effect if layout type is XmSPATIAL(), or if XmNselectionPolicy is XmSINGLE_SELECT or XmBROWSE_SELECT.

Multiple selection: the selected state of all items between the anchor item and the location cursor is set to that of the anchor item. The callbacks specified by XmNselectionCallback *are invoked.*

Extended selection: in Normal mode, all items are deselected, then any items between the anchor item and the location cursor are selected. In Add mode, the selected state of all items between the anchor item and the location cursor is set to that of the anchor item. The callbacks specified by XmNselectionCallback are invoked.

ContainerExtendCursor(type)

The action has no effect if layout type is XmSPATIAL(), or if XmNselectionPolicy is XmSINGLE_SELECT or XmBROWSE_SELECT.

The location cursor is moved. If type is Left, Right, Up, Down, First, or Last, the cursor is moved one item in the specified direction if possible (or to the first/last item).
Thereafter, the ContainerExtend() procedure is directly invoked.

ContainerHandleBtn1Down(string)

The XmDisplay resource XmNenableBtn1Transfer configures the integration of selection and transfer operations on Button 1.

If XmNenableBtn1Transfer is not XmOFF, and the pointer is over an unselected item, the actions ContainerBeginSelect() and ContainerEndSelect() are invoked in order to select the item. If thereafter there is no selected item, the Marquee start point is initialized, otherwise the action becomes a data transfer operation, and the ContainerStartTransfer() action is invoked.

If XmNenableBtn1Transfer is XmOFF, and if no data transfer has been initialized, the action specified by string is invoked to initiate selection. Possible values for string:

ContainerBeginSelect,Copy
ContainerBeginToggle,Copy
ContainerNoop,Link
ContainerBeginExtend,Move

ContainerHandleBtn1Motion(string)

If the XmDisplay XmNenableBtn1Transfer resource is not XmOFF, and a selection is in progress, a drag action is initiated. Otherwise, the action as specified by string is invoked, typically ContainerButtonMotion.

ContainerHandleBtn1Up(string)

If a Button1 transfer is in progress, the transfer is cancelled. Otherwise, the action as specified by string is invoked. Possible values for string:

ContainerEndSelect
ContainerEndToggle
ContainerEndExtend

ContainerHandleBtn2Down(string)
If the XmDisplay XmNenableBtn1Transfer resource is XmBUTTON2_ADJUST, the action ContainerBeginExtend is directly invoked. Otherwise, the action as specified by string is invoked. Possible values for string:

ContainerStartTransfer, Copy
ContainerStartTransfer, Link
ContainerStartTransfer, Move

ContainerHandleBtn2Motion(string)
If the XmDisplay XmNenableBtn1Transfer resource is not XmBUTTON2_ADJUST, and a selection is in progress, a drag action is initiated. Otherwise, the action as specified by string is invoked, typically ContainerButtonMotion.

ContainerHandleBtn2Up(string)
If the XmDisplay XmNenableBtn1Transfer resource is XmBUTTON2_ADJUST, the action directly invokes the ContainerEndExtend() action. Otherwise, the action as specified by string is invoked, typically ContainerEndTransfer.

ContainerMoveCursor(string)
Moves the location cursor to the container item in a given direction, if possible. Valid values of type: Up, Down, Left, Right, First, Last.

If the number of selected items is greater than 1, all items are deselected. The item at the location cursor is selected, and callbacks associated with the XmNselectionCallback resource are invoked.

ContainerPrimaryCopy()
Requests a primary selection copy to the Container. Any XmNdestinationCallback procedures are invoked. By default, the Container performs no data transfer: the programmer must provide a callback for the task.

ContainerPrimaryLink()
Requests a primary selection link to the Container. Any XmNdestinationCallback procedures are invoked. By default, the Container performs no data transfer: the programmer must provide a callback for the task.

ContainerPrimaryMove()

Requests a primary selection copy to the Container. Any XmNdestinationCallback procedures are invoked. By default, the Container performs no data transfer: the programmer must provide a callback for the task. Subsequently, the selection owner's XmNconvertCallback procedures are notified for the primary selection, with the target DELETE.

ContainerSelect()

Single or browse selection: deselects any selected item, and selects the item at the location cursor.

Multiple selection: reverses the selected state of the item at the location cursor, and makes this the anchor for any further operations.

Extended selection: in Normal mode, deselects all items, and selects the item at the location cursor. In Add mode, reverses the selected state of the item, which becomes the anchor for further operations.

In each case, callbacks associated with the XmNselectionCallback resource are invoked.

ContainerSelectAll()

Single or browse selection: deselects any selected item, and selects the item at the location cursor.

Multiple or extended selection: selects all container items.

In all cases, callbacks associated with the XmNselectionCallback resource are invoked.

ContainerStartTransfer(type)

The action saves the value of the parameter type for reference by later transfer operations. In XmSPATIAL layout, a DragContext is created, and a transfer operation is initiated. By default, unless overridden by a customized XmNconvertCallback procedure, if the drop occurs within the Container, then any dragged unselected item is moved to the pointer location, or if the item is selected, then all selected items are relocated to the pointer. Possible values for type are: Copy, Link, Move.

ContainerToggleMode()

Extended selection: toggles between Normal and Add mode.

Additional Behavior

Container has the following additional behavior:

<Double Click>
: Calls the XmNdefaultActionCallback callbacks.

<FocusIn>
: If the keyboard focus policy is explicit, sets the focus and draws the location cursor.

<FocusOut>
: If the keyboard focus policy is explicit, removes the focus and erases the location cursor.

See Also

`XmContainerCopy`(1), `XmContainerCopyLink`(1), `XmContainerCut`(1), `XmContainerGetItemChildren`(1), `XmContainerPaste`(1), `XmContainerPasteLink`(1), `XmContainerRelayout`(1), `XmContainerReorder`(1), `XmCreateObject`(1), `XmTransfer`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmIconGadget`(2), `XmManager`(2).

Name

XmDataField – The DataField widget class

Synopsis

Public Header:
<Xm/DataF.h>

Class Name:
XmDataField

Class Hierarchy:
Core → XmPrimitive → XmTextField → XmDataField

Class Pointer:
xmDataFieldWidgetClass

Instantiation:

```
widget = XmCreateDataField(parent, name,...)
or
widget = XtCreateWidget(name, xmDataFieldWidget-
Class,...)
```

Functions/Macros:

```
XmCreateDataField(), XmDataFieldSetString(), XmDa-
taFieldGetString(), XmDataFieldGetStringWes(), XmDa-
taFieldSetHighlight(), XmDataFieldSetAddMode(),
XmDataFieldGetSelection(), XmDataFieldSetSelection(),
XmDataFieldGetSelectionPosition(), XmDataFieldXYTo-
Pos(), XmDataFieldShowPosition(), XmDataFieldXYTo-
Pos(), XmDataFieldShowPosition(), XmDataFieldCut(),
XmDataFieldCopy(), XmDataFieldPaste(), XmDataFieldSe-
tEditable(), XmDataFieldSetInsertionPosition().
```

Availability

OpenMotif 2.2 and later. (Provisional Widget).

Description

The DataField widget is a Data Presentation widget that handles display and entry of data as text. The DataField widget is a subclass of the Motif XmTextField widget intended for data entry applications. In addition to all of the normal XmTextField functionality, it supports regular expression-based parsing and acceptance/rejection of its input through the XmNpicture resource, and right justification through the XmNalignment resource.

The DataField widget provides added capability by supporting several types of validation: a DataField widget containing an invalid value will not give-up focus; the user must enter a correct value before proceeding to another field.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNalignment	XmCAlignment	unsigned char	XmALIGNMENT_BEGINNING	CSG
XmNautoFill	XmCAutoFill	Boolean	True	CSG
XmNpicture	XmCPicture	String	NULL	CSG
XmNpictureErrorCallback	XmCCallback	XtCallbackList	NULL	C
XmNvalidateCallback	XmCCallback	XtCallbackList	NULL	C

XmNalignment

When set to XmALIGNMENT_END, the widget aligns all its text with the right hand side of the input area.

XmNautoFill

When set to True, the widget "auto-fills" its contents when it can determine that the next character in the string must be a particular literal. For instance, the picture "###-####" automatically inserts a ’-’ character after receiving three numeric digits as input.

XmNpicture

Specifies a picture for data entry in the widget. A picture acts as a template that formats the value you enter in a field. An example would be the US Phone Number picture: (###)###-####. The picture is used to convert characters entered into the field to a formatted value.

Character interpretations

The following lists and defines the characters you can use in a picture, and how the DataField widget interprets them.

Any numeric digit

? Case insensitive letter
& Uppercase letter (forces lowercase to uppercase)
@ Case insensitive character
! Uppercase character
; Interpret the following character literally
* Repeat the following character some number of times
[] Characters within brackets are optional
{} Characters within braces are grouped
, Alternative values
Other characters are interpreted literally.

Set XmNpicture to NULL to disable regular expression processing. The DataField widget is cleared whenever the XmNpicture resource is changed.

XmNpictureErrorCallback

Specifies a list of callbacks to be called when the XmDataField widget determines that data is being entered that does not match the format specified by the XmNpicture resource.

XmNvalidateCallback

Specifies a list of callbacks to be called when data has been entered in the XmDataField widget and the user has moved out of the XmDataField widget (usually by pressing the Tab key). The callbacks can reject the movement of focus.

Inherited Resources

DataField inherits behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass.

Resource	Inherited from	Resource	Inherited from
XmNactivateCallback	XmTextField	XmNhighlightColor	Primitive
XmNblinkRate	XmTextField	XmNhighlightOnEnter	Primitive
XmNcolumns	XmTextField	XmNhighlightPixmap	Primitive
XmNcursorPosition	XmTextField	XmNhighlightThickness	Primitive
XmNcursorPositionVisible	XmTextField	XmNlayoutDirection	Primitive
XmNdestinationCallback	XmTextField	XmNnavigationType	Primitive
XmNeditable	XmTextField	XmNpopupHandlerCallback	Primitive
XmNfocusCallback	XmTextField	XmNshadowThickness	Primitive
XmNfontList	XmTextField	XmNtoolTipString	Primitive
XmNgainPrimaryCallback	XmTextField	XmNtopShadowColor	Primitive

Resource	Inherited from	Resource	Inherited from
XmNlosePrimaryCallback	XmTextField	XmNtopShadowPixmap	Primitive
XmNlosingFocusCallback	XmTextField	XmNtraversalOn	Primitive
XmNmarginHeight	XmTextField	XmNunitType	Primitive
XmNmarginWidth	XmTextField	XmNuserData	Primitive
XmNmaxLength	XmTextField	XmNaccelerators	Core
XmNmodifyVerifyCallback	XmTextField	XmNancestorSensitive	Core
XmNmodifyVerifyCallback-Wcs	XmTextField	XmNbackground	Core
XmNmotionVerifyCallback	XmTextField	XmNbackgroundPixmap	Core
XmNpendingDelete	XmTextField	XmNborderColor	Core
XmNrenderTable	XmTextField	XmNborderPixmap	Core
XmNresizeWidth	XmTextField	XmNborderWidth	Core
XmNselectionArray	XmTextField	XmNcolormap	Core
XmNselectionArrayCount	XmTextField	XmNdepth	Core
XmNselectThreshold	XmTextField	XmNdestroyCallback	Core
XmNvalue	XmTextField	XmNheight	Core
XmNvalueChangedCallback	XmTextField	XmNinitialResourcesPersistent	Core
XmNvalueWes	XmTextField	XmNmappedWhenManaged	Core
XmNverifyBell	XmTextField	XmNscreen	Core
XmNbottomShadowColor	Primitive	XmNsensitive	Core
XmNbottomShadowPixmap	Primitive	XmNtranslations	Core
XmNconvertCallback	Primitive	XmNwidth	Core
XmNforeground	Primitive	XmNx	Core
XmNhelpCallback	Primitive	XmNy	Core

See Also

XmCreateObject(1), Core(2), XmPrimitive(2), XmTest, XmTextField(2).

Name

XmDialogShell widget class – the Shell parent for dialog boxes.

Synopsis

Public Header:
<Xm/DialogS.h>

Class Name:
XmDialogShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → TransientShell → XmDialogShell

Class Pointer:
xmDialogShellWidgetClass

Instantiation:
widget = `XmCreateDialogShell` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmDialogShellWidgetClass,...)

Functions/Macros:
`XmCreateDialogShell`(), `XmIsDialogShell`()

Description

DialogShell is the parent for dialog boxes. A DialogShell cannot be iconified separately, but only when the main application shell is iconified. The child of a DialogShell is typically a subclass of BulletinBoard and much of the functionality of DialogShell is based on this assumption.

Traits

DialogShell uses the XmQTdialogShellSavvy trait.

New Resources

DialogShell does not define any new resources.

Inherited Resources

DialogShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. DialogShell sets the default values of XmNdeleteResponse to XmUNMAP and XmNinput and XmNtransient to True. The default value of XmNborderWidth is reset to 0 by VendorShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNmaxAspectX	WMShell
XmNallowShellResize	Shell	XmNmaxAspectY	WMShell

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	Core	XmNminAspectX	WMShell
XmNaudibleWarning	VendorShell	XmNminAspectY	WMShell
XmNbackground	Core	XmNminHeight	WMShell
XmNbackgroundPixmap	Core	XmNminWidth	WMShell
XmNbaseHeight	WMShell	XmNmwmDecorations	VendorShell
XmNbaseWidth	WMShell	XmNmwmFunctions	VendorShell
XmNborderColor	Core	XmNmwmInputMode	VendorShell
XmNborderPixmap	Core	XmNmwmMenu	VendorShell
XmNborderWidth	Core	XmNnumChildren	Composite
XmNbuttonFontList	VendorShell	XmNoverrideRedirect	Shell
XmNbuttonRenderTable	VendorShell	XmNpopdownCalback	Shell
XmNchildren	Composite	XmNpopupCallback	Shell
XmNcolormap	Core	XmNpreeditType	VendorShell
XmNcreatePopupChildProc	Shell	XmNsaveUnder	Shell
XmNdefaultFontList	VendorShell	XmNscreen	Core
XmNdeleteResponse	VendorShell	XmNsensitive	Core
XmNdepth	Core	XmNshellUnitType	VendorShell
XmNdestroyCallback	Core	XmNtextFontList	VendorShell
XmNgeometry	Shell	XmNtextRenderTable	VendorShell
XmNheight	Core	XmNtitle	WMShell
XmNheightInc	WMShell	XmNtitleEncoding	WMShell
XmNiconMask	WMShell	XmNtoolTipEnable	VendorShell
XmNiconPixmap	WMShell	XmNtoolTipPostDuration	VendorShell
XmNiconWindow	WMShell	XmNtoolTipString	VendorShell
XmNinitialResourcesPersistent	Core	XmNtransient	WMShell
XmNinitialState	WMShell	XmNtransientFor	TransientShell
XmNinput	WMShell	XmNtranslations	Core
XmNinputMethod	VendorShell	XmNvisual	Shell
XmNinputPolicy	VendorShell	XmNwaitForWm	WMShell
XmNinsertPosition	Composite	XmNwidth	Core
XmNkeyboardFocusPolicy	VendorShell	XmNwidthInc	WMShell
XmNlabelFontList	VendorShell	XmNwindowGroup	WMShell
XmNlabelRenderTable	VendorShell	XmNwinGravity	WMShell
XmNlayoutDirection	VendorShell	XmNwmTimeout	WMShell

Resource	Inherited From	Resource	Inherited From
XmNmappedWhenManaged	Core	XmNx	Core
XmNmaxHeight	WMShell	XmNy	Core
XmNmaxWIdth	WMShell		

See Also

`XmCreateObject`(1), `Composite`(2), `Core`(2), `Shell`(2), `TransientShell`(2), `VendorShell`(2), `WMShell`(2), `XmBulletinBoardDialog`(2), `XmErrorDialog`(2), `XmFileSelectionDialog`(2), `XmFormDialog`(2), `XmInformationDialog`(2), `XmMessageDialog`(2), `XmPromptDialog`(2), `XmQuestionDialog`(2), `XmSelectionDialog`(2), `XmTemplateDialog`(2), `XmWarningDialog`(2), `XmWorkingDialog`(2).

Name

XmDisplay widget class –an object to store display-specific information.

Synopsis

Public Header:
<Xm/Display.h>

Class Name:
XmDisplay

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → TopLevelShell → ApplicationShell → XmDisplay

Class Pointer:
xmDisplayClass

Instantiation:
widget = `XtAppInitialize`(...)

Functions/Macros:
`XmGetXmDisplay`(), `XmIsDisplay`()

Availability

Motif 1.2 and later.

Description

The Display object stores display-specific information for use by the toolkit. An application has a Display object for each display it accesses. When an application creates its first shell on a display, typically by calling `XtAppInitialize`() or `XtAppCreateShell`(), a Display object is created automatically. There is no way to create a Display independently. The function `XmGetXmDisplay`() can be used to get the widget ID of the Display object.

The XmNdragInitiatorProtocolStyle and XmNdragReceiverProtocolStyle resources specify the drag protocol for an application that performs drag and drop operations. The two protocol styles are Dynamic and Preregister. Under the dynamic protocol, the initiator and receiver pass messages back and forth to handle drag and drop visuals. Under the Preregister protocol, the initiator handles drag and drop visuals by reading information that is preregistered and stored in properties. The actual protocol that is used by a specific initiator and receiver is based on the requested protocol styles of the receiver and initiator:

Drag Initiator	**Drag Receiver Protocol Style**			
Protocol Style	**Preregister**	**Prefer Preregister**	**Prefer Dynamic**	**Dynamic**
Preregister	PREREGISTER	PREREGISTER	PREREGISTER	DROP_ONLY
Prefer Preregister	PREREGISTER	PREREGISTER	PREREGISTER	DYNAMIC
Prefer Receiver	PREREGISTER	PREREGISTER	DYNAMIC	DYNAMIC
Prefer Dynamic	PREREGISTER	DYNAMIC	DYNAMIC	DYNAMIC
Dynamic	DROP_ONLY	DYNAMIC	DYNAMIC	DYNAMIC

New Resources

Display defines the following resources:

Name	Class	Type	Default	Access
XmNdefaultButtonEmphasis	XmCDefaultButtonEmphasis	XtEnum	XmEXTERNAL_HIGHLIGHT	SG
XmNdefaultVirtualBindings	XmCDefaultVirtualBindings	String	dynamic	SG
XmNdragInitiatorProtocolStyle	XmCDragInitiatorProtocolStyle	unsigned char	XmDRAG_PREFER_RECEIVER	SG
XmNdragReceiverProtocolStyle	XmCDragReceiverProtocolStyle	unsigned char	XmDRAG_PREFER_PREREGISTER	SG
XmNenableBtn1Transfer	XmCEnableBtn1Transfer	XtEnum	XmOFF	CG
XmNenableButtonTab	XmCEnableButtonTab	Boolean	False	CG
XmNenableDragIcon	XmCEnableDragIcon	Boolean	False	CG
XmNenableEtchedInMenu	XmCEnableEtchedinMenu	Boolean	False	CG
XmNenableMultiKeyBindings	XmCEnableMultiKeyBindings	Boolean	False	CG
XmNenableThinThickness	XmCEnableThinThickness	Boolean	False	CG
XmNenableToggleColor	XmCEnableToggleColor	Boolean	False	CG
XmNenableToggleVisual	XmCEnableToggleVisual	Boolean	False	CG
XmNenableUnselectableDrag	XmCEnableUnselectableDrag	Boolean	True	CG
XmNenableWarp	XmCEnableWarp	XtEnum	True	CG
XmNmotifVersion	XmCMotifVersion	int	XmVersion	CSG
XmNuserData	XmCUserData	XtPointer	NULL	CSG

XmNdefaultButtonEmphasis
In Motif 2.0 and later, specifies the manner in which button widgets and gadgets which have the XmNshowAsDefault resource set are displayed. A button which is the default has a double border. If XmNdefaultButtonEmphasis is XmINTERNAL_HIGHLIGHT, the location cursor is drawn between the double border. Otherwise, with a value of XmEXTERNAL_HIGHLIGHT, the location cursor is drawn outside of the double border. An internal indication uses less space for the button.

XmNdefaultVirtualBindings
In Motif 2.0 and later, specifies the default virtual bindings for the display.

XmNdragInitiatorProtocolStyle
The client's drag and drop protocol requirements or preference when it is the initiator of a drag and drop operation. Possible values:

```
XmDRAG_PREREGISTER          /* can only use the preregister protocol */
XmDRAG_DYNAMIC              /* can only use the dynamic protocol     */
XmDRAG_NONE                 /* drag and drop is disabled             */
XmDRAG_DROP_ONLY            /* only supports dragging                */
XmDRAG_PREFER_DYNAMIC       /* supports both but prefers dynamic     */
XmDRAG_PREFER_PREREGISTER
                            /* supports both but prefers preregister */
XmDRAG_PREFER_RECEIVER      /* supports both; prefers receiver's     */
                            /* protocol */
```

XmNdragReceiverProtocolStyle
The client's drag and drop protocol requirements or preference when it is the receiver. Possible values:

```
XmDRAG_PREREGISTER          /* can only use the preregister protocol */
XmDRAG_DYNAMIC              /* can only use the dynamic protocol     */
XmDRAG_NONE                 /* drag and drop is disabled             */
XmDRAG_DROP_ONLY            /* only supports dropping                */
XmDRAG_PREFER_DYNAMIC       /* supports both but prefers dynamic     */
XmDRAG_PREFER_PREREGISTER
                            /* supports both but prefers preregister*/
```

XmNenableBtn1Transfer
In Motif 2.0 and later, configures selection and transfer actions for Button1. The Container, Text, TextField, and List actions are affected by this resource. Possible values:

```
XmOFF                   /* selection and transfer disabled for button 1 */
XmBUTTON2_TRANSFER      /* selection on button 1, transfer on button 2 */
XmBUTTON2_ADJUST        /* selection on button 1, adjust on button 2    */
```

XmNenableButtonTab
: In Motif 2.0 and later, configures the action of the Tab key with respect to keyboard navigation. If True, KNextField and KPrevField will behave like an Arrow key, moving the focus between widgets within a Tab Group, until the boundary of a Tab group is reached, at which point a subsequent navigation will move the focus into the next or previous Tab group. If False, KNextField and KPrevField move the focus to the next or previous tab group respectively.

XmNenableDragIcon
: In Motif 2.0 and later, a set of alternative icons representing the drag and drop default cursors is available. A value of True specifies that the newer icons are the default.

XmNenableEtchedInMenu
: In Motif 2.0 and later, specifies the way in which buttons within menus are shadowed when the widget is activated. The value False results in an etched out appearance, True gives an etched in shadowing, which is consistent with the appearance of activated buttons outside of a menu system. The resource affects PushButton, ToggleButton, CascadeButton widgets and gadget counterparts.

XmNenableMultiKeyBindings
: In Motif 2.1, merges an additional set of translations into the resource database which are compatible with CDE cancel translations.

XmNenableThinThickness
: Introduced in Motif 1.2.5 to provide CDE style shadowing and highlighting, used originally only by the ScrollBar. In Motif 2.1, the number of widgets sensitive to the resource is considerably expanded. If True, the default shadow thickness is 1, otherwise the default is 2.

XmNenableToggleColor
: In Motif 2.0 and later, specifies how the default value of a toggle's XmNselectColor is determined. True means that the default is taken from the XmNhighlightColor value, False uses the XmNbackground. XmNenableToggleColor is ignored if an explicit XmNselectColor is supplied to the toggle widget or gadget. The resource only takes effect if the indicator type of the toggle is XmONE_OF_MANY or XmONE_OF_MANY_ROUND.

XmNenableToggleVisual
: In Motif 2.0 and later, controls the default appearance of toggles. If False, a toggle with the indicator type of XmONE_OF_MANY is drawn as a diamond, and a toggle with indicator type XmN_OF_MANY is drawn square. If True, a toggle within a radio box has the default indicator type XmONE_OF_MANY_ROUND, which is rendered as a circle. A toggle outside of a radio box has the default indicator type XmN_OF_MANY, which is rendered square, and a check mark is displayed when XmNindicatorOn is True.

XmNenableUnselectableDrag
: In Motif 2.0 and later, specifies whether it is possible to initiate a drag operation from Label, LabelGadget, or Scale widgets. The value True enables drag operations from the widgets.

XmNenableWarp
: In Motif 2.0 and later, specifies if the application is permitted to warp the pointer away from the user. The value True enables warping.

XmNmotifVersion
: In Motif 2.0 and later, specifies the current version of Motif.

XmNuserData
: In Motif 2.0 and later, specifies a pointer to data that the application can attach to the XmDisplay object. The resource is unused internally.

Callback Resources

In Motif 2.0 and later, Display defines the following callback resources:

Callback	Reason Constant
XmNdragStartCallback	XmCR_DRAG_START
XmNnoFontCallback	XmCR_NO_FONT
XmNnoRenditionCallback	XmCR_NO_RENDITION

XmNdragStartCallback
: List of callbacks that are called when the procedure `XmDragStart`() is invoked.

XmNnoFontCallback
: List of callbacks that are called when an XmRendition object fails in an attempt to load a font. This may happen if the object is created with an XmNloadModel of XmLOAD_IMMEDIATE, and the font cannot be loaded there and then, or if the XmNloadModel is XmLOAD_DEFERRED and a later attempt is made to render a compound string using an unloadable font. A callback can be supplied to rectify the situation: it can find or specify an alternative font, and invoke the function `XmRenditionUpdate`() upon the rendition object.

XmNnoRenditionCallback
: List of callbacks that are called when an attempt is made to render using a rendition tag which does not match any entry within a given render table. A callback can be supplied to rectify the problem: it can create a new rendition with the problematic tag, and augment the render table.

Callback Structure

Each XmNnoFontCallback or XmNnoRenditionCallback procedure is passed the following structure:

```
typedef struct {
    int             reason;         /* the reason that the callback was called   */
    XEvent          *event;         /* points to event that triggered callback   */
    XmRendition     rendition;      /* the rendition with a missing font         */
    char            *font_name;     /* the font which is not loadable            */
    XmRenderTable   render_table;   /* the render table with a missing rendition */
    XmString        tag;            /* the tag of the missing rendition          */
} XmDisplayCallbackStruct;
```

The render_table and tag elements are only applicable to callbacks on the XmNnoRenditionCallback list. rendition and font_name are valid only for XmNnoFontCallback callbacks.

In addition, an XmNdragStartCallback procedure is passed the following structure:

```
typedef struct {
    int             reason;     /* the reason that the callback was called         */
    XEvent          *event;     /* points to event structure that triggered callback */
    Widget          widget;     /* the ID of the widget where the drag initiated   */
    Boolean         doit;       /* do the action (True) or undo it (False)         */
} XmDragStartCallbackStruct;
```

Inherited Resources

None of the resources inherited by Display can be set by the programmer or user.

See Also

`XmGetXmDisplay`(1), `ApplicationShell`(2), `Composite`(2), `Core`(2), `Shell`(2), `TopLevelShell`(2), `VendorShell`(2), `WMShell`(2), `XmScreen`(2).

Name

XmDragContext widget class –an object used to store information about a drag transaction.

Synopsis

Public Header:
<Xm/DragDrop.h>

Class Name:
XmDragContext

Class Pointer:
xmDragContextClass

Class Hierarchy:
Core → DragContext

Instantiation:
widget = `XmDragStart`(...)

Functions/Macros:
`XmDragCancel`(), `XmDragStart`()

Availability

Motif 1.2 and later.

Description

The DragContext object stores information that the toolkit needs to process a drag transaction. An application does not explicitly create a DragContext widget, but instead initiates a drag and drop operation by calling `XmDragStart`(), which initializes and returns a DragContext widget. The DragContext stores information about the types of data and operations of the drag source, the drag icons that are used during the drag, and the callbacks that are called during different parts of the drag. These characteristics can be specified as resources when the DragContext is created using `XmDragStart`().

Each drag operation has a unique DragContext that is freed by the toolkit when the operation is complete. The initiating and receiving clients in a drag and drop operation both use the DragContext to keep track of the state of the operation. The drag-over visual effects that are used during a drag operation depend on the drag protocol that is being used. Under the preregister protocol, either a cursor or a pixmap can be used, since the server is grabbed. Under the dynamic protocol, the X cursor is used.

New Resources

DragContext defines the following resources:

Name	Class	Type	Default	Access
XmNblendModel	XmCBlendModel	unsigned char	XmBLEND_ALL	CG
XmNclientData	XmCClientData	XtPointer	NULL	CSG
XmNconvertProc	XmCConvertProc	XtConvertSelection-IncrProc	NULL	CSG
XmNcursorBackground	XmCCursorBackground	Pixel	dynamic	CSG
XmNcursorForeground	XmCCursorForeground	Pixel	dynamic	CSG
XmNdragOperations	XmCDragOperations	unsigned char	XmDROP_COPY | XmDROP_MOVE	C
XmNexportTargets	XmCExportTargets	Atom *	NULL	CSG
XmNincremental	XmCIncremental	Boolean	False	CSG
XmNinvalidCursorForeground	XmCCursorForeground	Pixel	dynamic	CSG
XmNnoneCursorForeground	XmCCursorForeground	Pixel	dynamic	CSG
XmNnumExportTargets	XmCNumExportTargets	Cardinal	0	CSG
XmNoperationCursorIcon	XmCOperationCursorIcon	Widget	dynamic	CSG
XmNsourceCursorIcon	XmCSourceCursorIcon	Widget	dynamic	CSG
XmNsourcePixmapIcon	XmCSourcePixmapIcon	Widget	dynamic	CSG
XmNstateCursorIcon	XmCStateCursorIcon	Pixel	dynamic	CSG
XmNvalidCursorForeground	XmCCursorForeground	Pixel	dynamic	CSG

XmNblendModel

The combination of DragIcons that are blended to produce a drag-over visual. Possible values:

```
XmBLEND_ALL             /* source, state, and operation */
XmBLEND_STATE_SOURCE    /* source and state             */
XmBLEND_JUST_SOURCE     /* source only                  */
XmBLEND_NONE            /* no drag-over visual          */
```

XmNclientData

The client data that is passed to the XmNconvertProc.

XmNconvertProc

A procedure of type XtConvertSelectionIncrProc that converts the data to the format(s) specified by the receiving client. The widget argument passed to this procedure is the DragContext widget and the selection atom is _MOTIF_DROP. If XmNincremental is False, the conversion procedure should process the conversion atomically and ignore the max_length, client_data, and request_-id argu-

ments. Allocate any data returned by XmNconvertProc using `XtMalloc`() and it will be freed automatically by the toolkit after the transfer.

XmNcursorBackground
The background color of the cursor.

XmNcursorForeground
The foreground color of the cursor when the state icon is not blended. The default value is the foreground color of the widget passed to XmDragStart().

XmNdragOperations
The valid operations for the drag. The value is a bit mask that is formed by combining one or more of these possible values:

```
XmDROP_COPY     /* copy operations are valid */
XmDROP_LINK     /* link operations are valid */
XmDROP_MOVE     /* move operations are valid */
XmDROP_NOOP     /* no operations are valid   */
```

For Text and TextField widgets, the default value is XmDROP_COPY | XmDROP_MOVE. For List widgets and Label and subclasses, the default is XmDROP_COPY.

XmNexportTargets
The list of target atoms that the source data can be converted to.

XmNincremental
If True, the initiator uses the Xt incremental selection transfer mechanism. If False (default), the initiator uses atomic transfer.

XmNinvalidCursorForeground
The foreground color of the cursor when the state is invalid. The default value is the value of the XmNcursorForeground resource.

XmNnoneCursorForeground
The foreground color of the cursor when the state is none. The default value is the value of the XmNcursorForeground resource.

XmNnumExportTargets
The number of atoms in the XmNexportTargets list.

XmNoperationCursorIcon
The drag icon used to show the type of drag operation being performed. If the value is NULL, the default Screen icons are used.

XmNsourceCursorIcon
The drag icon used to represent the source data under the dynamic protocol. If the value is NULL, the default Screen icon is used.

XmNsourcePixmapIcon
The drag icon used to represent the source data under the preregister protocol. If the value is NULL, XmNsourceCursorIcon is used.

XmNstateCursorIcon
The drag icon used to show the state of a drop site. If the value is NULL, the default Screen icons are used.

XmNvalidCursorForeground
The foreground color of the cursor when the state is valid. The default value is the value of the XmNcursorForeground resource.

Callback Resources

DragContext defines the following callback resources:

Callback	Reason Constant
XmNdragDropFinishCallback	XmCR_DRAG_DROP_FINISH
XmNdragMotionCallback	XmCR_DRAG_MOTION
XmNdropFinishCallback	XmCR_DROP_FINISH
XmNdropSiteEnterCallback	XmCR_DROP_SITE_ENTER
XmNdropSiteLeaveCallback	XmCR_DROP_SITE_LEAVE
XmNdropStartCallback	XmCR_DROP_START
XmNoperationChangedCallback	XmCR_OPERATION_CHANGED
XmNtopLevelEnterCallback	XmCR_TOP_LEVEL_ENTER
XmNtopLevelLeaveCallback	XmCR_TOP_LEVEL_LEAVE

XmNdragDropFinishCallback
List of callbacks that are called when the entire transaction is finished.

XmNdragMotionCallback
List of callbacks that are called when the pointer moves during a drag.

XmNdropFinishCallback
List of callbacks that are called when the drop is finished.

XmNdropSiteEnterCallback
List of callbacks that are called when the pointer enters a drop site.

XmNdropSiteLeaveCallback
List of callbacks that are called when the pointer leaves a drop site.

XmNdropStartCallback
List of callbacks that are called when a drop is started.

XmNoperationChangedCallback
List of callbacks that are called when the user changes the operation during a drag.

XmNtopLevelEnterCallback
List of callbacks that are called when the pointer enters a top-level window or root window.

XmNtopLevelLeaveCallback
List of callbacks that are called when the pointer leaves a top-level window or the root window.

Callback Structure

The XmNdragDropFinishCallback is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason the callback was called     */
    XEvent      *event;         /* event structure that triggered callback */
    Time        timeStamp;      /* time at which operation completed      */
} XmDragDropFinishCallbackStruct, *XmDragDropFinishCallback;
```

The XmNdragMotionCallback is passed the following structure:

```
typedef struct {
    int             reason;             /* reason the callback was called */
    XEvent          *event;             /* event that triggered callback */
    Time            timeStamp;          /* timestamp of logical event    */
    unsigned char   operation;          /* current operation             */
    unsigned char   operations;         /* supported operations          */
    unsigned char   dropSiteStatus;     /* valid, invalid, or none       */
    Position        x;                  /* x-coordinate of pointer       */
    Position        y;                  /* y-coordinate of pointer       */
} XmDragMotionCallbackStruct, *XmDragMotionCallback;
```

The XmNdropFinishCallback is passed the following structure:

```
typedef struct {
    int             reason;             /* reason the callback was called  */
    XEvent          *event;             /* event that triggered callback   */
    Time            timeStamp;          /* time at which drop completed    */
    unsigned char   operation;          /* current operation               */
    unsigned char   operations;         /* supported operations            */
    unsigned char   dropSiteStatus;     /* valid, invalid, or none         */
    unsigned char   dropAction;         /* drop, cancel, help, or interrupt */
    unsigned char   completionStatus;   /* success or failure              */
} XmDropFinishCallbackStruct, *XmDropFinishCallback;
```

The XmNdropSiteEnterCallback is passed the following structure:

```
typedef struct {
    int             reason;          /* reason the callback was called */
    XEvent          *event;          /* event that triggered callback */
    Time            timeStamp;       /* time of crossing event        */
    unsigned char   operation;       /* current operation             */
    unsigned char   operations;      /* supported operations          */
    unsigned char   dropSiteStatus;  /* valid, invalid, or none       */
    Position        x;               /* x-coordinate of pointer       */
    Position        y;               /* y-coordinate of pointer       */
} XmDropSiteEnterCallbackStruct, *XmDropSiteEnterCallback;
```

The XmNdropSiteLeaveCallback is passed the following structure:

```
typedef struct {
    int             reason;          /* reason the callback was called */
    XEvent          *event;          /* event that triggered callback */
    Time            timeStamp;       /* time of crossing event        */
} XmDropSiteLeaveCallbackStruct, *XmDropSiteLeaveCallback;
```

The XmNdropStartCallback is passed the following structure:

```
typedef struct {
    int             reason;          /* reason the callback was called   */
    XEvent          *event;          /* event that triggered callback    */
    Time            timeStamp;       /* time at which drag completed     */
    unsigned char   operation;       /* current operation                */
    unsigned char   operations;      /* supported operations             */
    unsigned char   dropSiteStatus;  /* valid, invalid, or none          */
    unsigned char   dropAction;      /* drop, cancel, help, or interrupt */
    Position        x;               /* x-coordinate of pointer          */
    Position        y;               /* y-coordinate of pointer          */
} XmDropStartCallbackStruct, *XmDropStartCallback;
```

The XmNoperationChangedCallback is passed the following structure:

```
typedef struct {
    int             reason;          /* reason the callback was called */
    XEvent          *event;          /* event that triggered callback */
    Time            timeStamp;       /* timestamp of logical event    */
    unsigned char   operation;       /* current operation             */
    unsigned char   operations;      /* supported operations          */
    unsigned char   dropSiteStatus;  /* valid, invalid, or none       */
} XmOperationChangedCallbackStruct, *XmOperationChangedCallback;
```

The XmNtopLevelEnterCallback is passed the following structure:

```
typedef struct {
    int             reason;             /* reason callback was called    */
    XEvent          *event;             /* event that triggered callback */
    Time            timestamp;          /* timestamp of logical event    */
    Screen          screen;             /* screen of top-level window    */
    Window          window;             /* window being entered          */
    Position        x;                  /* x-coordinate of pointer       */
    Position        y;                  /* y-coordinate of pointer       */
    unsigned char   dragProtocolStyle;  /* drag protocol of initiator    */
} XmTopLevelEnterCallbackStruct, *XmTopLevelEnterCallback;
```

The XmNtopLevelLeaveCallback is passed the following structure:

```
typedef struct {
    int             reason;             /* reason the callback was called */
    XEvent          *event;             /* event that triggered callback */
    Time            timestamp;          /* timestamp of logical event    */
    Screen          screen;             /* screen of top-level window    */
    Window          window;             /* window being left             */
} XmTopLevelLeaveCallbackStruct, *XmTopLevelLeaveCallback;
```

The operations field in these structures specifies the set of operations supported for the data being dragged. The toolkit initializes the value based on the operations field of the XmDragProcCallbackStruct, the XmNdropSiteOperations resource of the DropSite, the XmNdragOperations resource of the DragContext and the operation selected by the user. The operation field in these structures specifies the current operation. The toolkit initializes the value based on the value of the operation field of the XmDragProcCallbackStruct, operations, and the XmNdropSiteOperations resource of the Drop Site.

The dropSiteStatus field in these structures specifies whether or not the drop site is valid. The toolkit initializes the value based on the XmNimportTargets resource of the DropSite and the XmNexportTargets resource of the DragContext and the location of the pointer. The possible values are XmDROP_SITE_VALID, XmDROP_SITE_INVALID, and XmNO_DROP_SITE.

The dropAction field in these structures specifies the action associated with the drop. The possible values are XmDROP, XmDROP_CANCEL,

XmDROP_INTERRUPT, and XmDROP_HELP[1]. XmDROP_INTERRUPT is unsupported and is interpreted as XmDROP_CANCEL.

The completionStatus field in the XmDropFinishCallbackStruct specifies the status of the drop transaction, which determines the drop visual effect. The value of this field can be changed by the XmNdropFinishCallbackStruct. The possible values are XmDROP_SUCCESS[2] and XmDROP_FAILURE[3].

Inherited Resources

DragContext inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. DragContext sets the default value of XmNborderWidth to 0.

Name	Inherited From	Name	Inherited From
XmNaccelerators	Core	XmNheight	Core
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNscreen	Core
XmNborderColor	Core	XmNsensitive	Core
XmNborderPixmap	Core	XmNtranslations	Core
XmNborderWidth	Core	XmNwidth	Core
XmNcolormap	Core	XmNx	Core
XmNdepth	Core	XmNy	Core
XmNdestroyCallback	Core		

Translations

Event	Action
BDrag Motion	DragMotion()
BDrag Release	FinishDrag()
KCancel	CancelDrag()
KHelp	HelpDrag()

Action Routines

DragContext defines the following action routines:

1.Erroneously given as DROP_HELP in 1st and 2nd edition.

2.Erroneously given as XmSUCCESS in 1st and 2nd edition.

3.Erroneously given as XmFAILURE in 1st and 2nd edition.

CancelDrag()
: Cancels the drag operation and frees the associated DragContext.

DragMotion()
: Drags the selected data as the pointer is moved.

FinishDrag()
: Completes the drag operation and initiates the drop operation.

HelpDrag()
: Starts a conditional drop that allows the receiving client to provide help information to the user. The user can cancel or continue the drop operation in response to this information.

See Also

`XmDragCancel`(1), `XmDragStart`(1), `XmGetDragContext`(1), `Core`(2), `XmDisplay`(2), `XmDragIcon`(2), `XmDropSite`(2), `XmDropTransfer`(2), `XmScreen`(2).

Name

XmDragIcon widget class – an object used to represent the data in a drag and drop operation.

Synopsis

Public Header:
<Xm/DragDrop.h>

Class Name:
XmDragIcon

Class Pointer:
xmDragIconObjectClass

Class Hierarchy:
Object → DragIcon

Instantiation:
widget = `XmCreateDragIcon`(...)

Functions/Macros:
`XmCreateDragIcon`(), `XmIsDragIconObjectClass`()

Availability

Motif 1.2 and later.

Description

A DragIcon is an object that represents the source data in a drag and drop transaction. During a drag operation, the cursor changes into a visual that is created by combining the various DragIcons specified in the DragContext associated with the operation. A DragIcon is created using the `XmCreateDragIcon`() function or from entries in the resource database.

A drag-over visual can have both a static and a dynamic part. The static part of the visual is the DragIcon that represents the source data. The dynamic parts can be DragIcons that change to indicate the type of operation that is being performed and whether the pointer is over a valid or an invalid drop site. The XmNblendModel resource of the DragContext for a drag and drop operation specifies which icons are blended to produce the drag-over visual. DragIcon resources specify the relative positions of the operation and state icons if they are used. When a DragIcon is not specified, the default DragIcons from the appropriate Screen object are used.

New Resources

DragIcon defines the following resources:

Name	Class	Type	Default	Access
XmNattachment	XmCAttachment	unsigned char	XmATTACH_NORTH_WEST	CSG
XmNdepth	XmCDepth	int	1	CSG
XmNheight	XmCHeight	Dimension	0	CSG
XmNhotX	XmCHot	Position	0	CSG
XmNhotY	XmCHot	Position	0	CSG
XmNmask	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNoffsetX	XmCOffset	Position	0	CSG
XmNoffsetY	XmCOffset	Position	0	CSG
XmNpixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNwidth	XmCWidth	Dimension	0	CSG

XmNattachment
: The relative location on the source icon where the state or operation icon is attached. Possible values:

XmATTACH_NORTH_WEST	XmATTACH_NORTH
XmATTACH_NORTH_EAST	XmATTACH_EAST
XmATTACH_SOUTH_EAST	XmATTACH_SOUTH
XmATTACH_SOUTH_WEST	XmATTACH_WEST
XmATTACH_CENTER	XmATTACH_HOT

XmNdepth
: The depth of the pixmap.

XmNheight
: The height of the pixmap.

XmNhotX
: The x-coordinate of the hotspot of the cursor.

XmNhotY
: The y-coordinate of the hotspot of the cursor.

XmNmask
: The mask for the DragIcon pixmap.

XmNoffsetX
: The horizontal offset in pixels of the origin of the state or operation icon relative to the attachment point on the source icon.

XmNoffsetY
The vertical offset in pixels of the origin of the state or operation icon relative to the attachment point on the source icon.

XmNpixmap
The pixmap for the DragIcon.

XmNwidth
The width of the pixmap.

Inherited Resources

DragIcon inherits the following resource:

Resource	Inherited From
XmNdestroyCallback	Object

See Also

`XmCreateObject`(1), `Object`(2), `XmDisplay`(2), `XmDragContext`(2), `XmDropSite`(2), `XmDropTransfer`(2), `XmScreen`(2).

Name

XmDrawingArea widget class –a simple manager widget for interactive drawing.

Synopsis

Public Header:
<Xm/DrawingA.h>

Class Name:
XmDrawingArea

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmDrawingArea

Class Pointer:
xmDrawingAreaWidgetClass

Instantiation:
widget = `XmCreateDrawingArea` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmDrawingAreaWidgetClass,...)

Functions/Macros:
`XmCreateDrawingArea`(), `XmIsDrawingArea`()

Description

DrawingArea provides a blank canvas for interactive drawing. The widget does not do any drawing of its own. Since DrawingArea is a subclass of Manager, it can provide simple geometry management of multiple widget or gadget children. The widget does not define any behavior except for invoking callbacks that notify an application when it receives input events, exposure events, and resize events.

New Resources

DrawingArea defines the following resources:

Name	Class	Type	Default	Access
XmNmarginHeight	XmCMarginHeight	Dimension	10	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	10	CSG
XmNresizePolicy	XmCResizePolicy	unsigned char	XmRESIZE_ANY	CSG

XmNmarginHeight
The spacing between a DrawingArea's top or bottom edge and any child widget.

XmNmarginWidth
The spacing between a DrawingArea's right or left edge and any child widget.

XmNresizePolicy
How DrawingArea widgets are resized. Possible values:

```
XmRESIZE_NONE       /* remain at fixed size       */
XmRESIZE_GROW       /* expand only                */
XmRESIZE_ANY        /* shrink or expand, as needed */
```

Callback Resources

DrawingArea defines the following callback resources:

Callback	Reason Constant
XmNconvertCallback	XmCR_OK
XmNdestinationCallback	XmCR_OK
XmNexposeCallback	XmCR_EXPOSE
XmNinputCallback	XmCR_INPUT
XmNresizeCallback	XmCR_RESIZE

XmNconvertCallback
In Motif 2.0 and later, specifies the list of callbacks called when a request is made to convert a selection.

XmNdestinationCallback
In Motif 2.0 and later, specifies the list of callbacks called when the DrawingArea is the destination of a data transfer.

XmNexposeCallback
List of callbacks that are called when the DrawingArea receives an exposure event.

XmNinputCallback
List of callbacks that are called when the DrawingArea receives a keyboard or mouse event.

XmNresizeCallback
List of callbacks that are called when the DrawingArea receives a resize event.

Callback Structure

Each expose, resize, and input callback function is passed the following structure:

```
typedef struct {
    int         reason;     /* the reason that the callback was called */
    XEvent      *event;     /* event structure that triggered callback; */
                            /* for XmNresizeCallback, this is NULL     */
    Window      window;     /* the widget's window                     */
} XmDrawingAreaCallbackStruct;
```

Convert callbacks are fully described within the sections covering the Uniform Transfer Model. See `XmTransfer`(1) for more details. For quick reference, a pointer to the following structure is passed to callbacks on the XmNconvertCallback list:

```
typedef struct {
    int           reason;          /* reason that the callback is invoked     */
    XEvent        *event;          /* points to event that triggered callback */
    Atom          selection;       /* requested conversion selection          */
    Atom          target;          /* the conversion target                   */
    XtPointer     source_data;     /* selection source information            */
    XtPointer     location_data;   /* information about data to be transferred*/
    int           flags;           /* input status of the conversion          */
    XtPointer     parm;            /* parameter data for the target           */
    int           parm_format;     /* format of parameter data                */
    unsigned long parm_length;     /* number of elements in                   */
                                   /*       parameter data                    */
    Atom          parm_type;       /* the type of the parameter data          */
    int           status;          /* output status of the conversion         */
    XtPointer     value;           /* returned conversion data                */
    Atom          type;            /* type of conversion data returned        */
    int           format;          /* format of the conversion data           */
    unsigned long length;          /* number of elements in conversion data */
} XmConvertCallbackStruct;
```

Destination callbacks are fully described within the sections covering the Uniform Transfer Model. See `XmTransfer`(1) for more details. For quick reference, a pointer to the following structure is passed to callbacks on the XmNdestinationCallback list:

```
typedef struct {
    int           reason;          /* reason that the callback is invoked     */
    XEvent        *event;          /* points to event that triggered callback */
    Atom          selection;       /* the requested selection type, as an Atom */
    XtEnum        operation;       /* the type of transfer requested          */
    int           flags;           /* whether destination and source are same */
    XtPointer     transfer_id;     /* unique identifier for the request       */
    XtPointer     destination_data; /* information about the destination */
    XtPointer     location_data; /* information about the data        */
    Time          time;            /* the time when transfer operation started */
} XmDestinationCallbackStruct;
```

Inherited Resources

DrawingArea inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of XmNborderWidth is reset to 0 by Manager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Translations

The translations for DrawingArea include those of Manager. All of the events in the inherited translations except <BtnMotion>, <EnterWindow>, <LeaveWindow>, <FocusIn>, and <FocusOut> call the DrawingAreaInput() action before calling the Manager actions.

DrawingArea has the following additional translations:

Event	Action
MAny Bany Press	DrawingAreaInput()

Event	Action
Many Bany Release	DrawingAreaInput()
Many KAny Press	DrawingAreaInput() ManagerGadgetKeyInput()
MAny KAny Release	DrawingAreaInput()

Action Routines

DrawingArea defines the following action routines:

DrawingAreaInput()
: When a widget child of a DrawingArea receives a keyboard or mouse event, this action routine invokes the list of callbacks specified by XmNinputCallback.

ManagerGadgetKeyInput()
: When a gadget child of a DrawingArea receives a keyboard or mouse event, this action routine processes the event.

Additional Behavior

DrawingArea has the following additional behavior:

<Expose>
: Invokes the XmNexposeCallback callbacks.

<WidgetResize>
: Invokes the XmNresizeCallback callbacks.

See Also

`XmCreateObject`(1), `XmTransfer`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmManager`(2).

Name

XmDrawnButton widget class – a button widget that provides a graphics area.

Synopsis

Public Header:
<Xm/DrawnB.h>

Class Name:
XmDrawnButton

Class Hierarchy:
XmPrimitive → XmLabel → XmDrawnButton

Class Pointer:
xmDrawnButtonWidgetClass

Instantiation:
widget = `XmCreateDrawnButton` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmDrawnButtonWidgetClass,...)

Functions/Macros:
`XmCreateDrawnButton`(), `XmIsDrawnButton`()

Description

DrawnButton is an empty widget window, surrounded by a shaded border. The widget provides a graphics area that can act like a PushButton. The graphics can be dynamically updated by the application.

Traits

DrawnButton holds the XmQTactivatable trait, which is inherited by any derived classes, and uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

New Resources

DrawnButton defines the following resources:

Name	Class	Type	Default	Access
XmNmultiClick	XmCMultiClick	unsigned char	dynamic	CSG
XmNpushButtonEnabled	XmCPushButtonEnabled	Boolean	False	CSG
XmNshadowType	XmCShadowType	unsigned char	XmSHADOW_ETCHED_IN	CSG

XmNmultiClick
A flag that determines whether successive button clicks are processed or ignored. Possible values:

```
XmMULTICLICK_DISCARD    /* ignore successive button clicks; */
                        /* default value in a menu system   */
```

```
XmMULTICLICK_KEEP       /* count successive button clicks; */
                        /* default value when not in a menu */
```

XmNpushButtonEnabled
If False (default), the shadow drawing doesn't appear three dimensional; if True, the shading provides a pushed in or raised appearance as for the PushButton widget.

XmNshadowType
The style in which shadows are drawn. Possible values:

```
XmSHADOW_IN             /* widget appears inset          */
XmSHADOW_OUT            /* widget appears outset         */
XmSHADOW_ETCHED_IN      /* double line; widget appears inset */
XmSHADOW_ETCHED_OUT     /* double line; widget appears raised */
```

Callback Resources

DrawnButton defines the following callback resources:

Callback	Reason Constant
XmNactivateCallback	XmCR_ACTIVATE
XmNarmCallback	XmCR_ARM
XmNdisarmCallback	XmCR_DISARM
XmNexposeCallback	XmCR_EXPOSE
XmNresizeCallback	XmCR_RESIZE

XmNactivateCallback
List of callbacks that are called when BSelect is pressed and released inside of the widget.

XmNarmCallback
List of callbacks that are called when BSelect is pressed while the pointer is inside the widget.

XmNdisarmCallback
List of callbacks that are called when BSelect is released after it has been pressed inside of the widget.

XmNexposeCallback
List of callbacks that are called when the widget receives an exposure event.

XmNresizeCallback
List of callbacks that are called when the widget receives a resize event.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;      /* the reason that the callback was called */
    XEvent      *event;      /* event structure that triggered callback */
    Window      window;      /* ID of window in which the event occurred */
    int         click_count; /* number of multi-clicks                   */
} XmDrawnButtonCallbackStruct;
```

event is NULL for XmNresizeCallback and is sometimes NULL for XmNactivateCallback.

click_count is meaningful only for XmNactivateCallback. Furthermore, if the XmNmultiClick resource has the value XmMULTICLICK_KEEP, then XmNactivateCallback is called for each click, and the value of *click_count* is the number of clicks that have occurred in the last sequence of multiple clicks. If the XmNmultiClick resource is set to XmMULTICLICK_DISCARD, then *click_count* always has a value of 1.

Inherited Resources

DrawnButton inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. DrawnButton sets default value of XmNlabelString to XmUNSPECIFIED[1]. The default value of XmNborderWidth is reset to 0 by Primitive. In Motif 2.0 and earlier, the default value of XmNhighlightThickness and XmNshadowThickness are reset to 2. In Motif 2.1 and later, the default values depend upon the XmDisplay XmNenableThinThickness resource: if True, the default is 1, otherwise 2.

Resource	Inherited From	Resource	Inherited From
XmNaccelerator	XmLabel	XmNlayoutDirection	XmPrimitive
XmNaccelerators	Core	XmNmappedWhenManaged	Core
XmNacceleratorText	XmLabel	XmNmarginBottom	XmLabel
XmNalignment	XmLabel	XmNmarginHeight	XmLabel
XmNancestorSensitive	Core	XmNmarginLeft	XmLabel
XmNbackground	Core	XmNmarginRight	XmLabel
XmNbackgroundPixmap	Core	XmNmarginTop	XmLabel
XmNborderColor	Core	XmNmarginWidth	XmLabel
XmNborderPixmap	Core	XmNmnemonicCharSet	XmLabel
XmNborderWidth	Core	XmNmnemonic	XmLabel
XmNbottomShadowColor	XmPrimitive	XmNnavigationType	XmPrimitive

1.Given as “” in 1st and 2nd editions. This is imprecise. The XmLabel superclass treats XmUNSPECIFIED as a special value, which maps to an empty XmString.

Resource	Inherited From	Resource	Inherited From
XmNbottomShadowPixmap	XmPrimitive	XmNpopupHandlerCallback	XmPrimitive
XmNcolormap	Core	XmNrecomputeSize	XmLabel
XmNconvertCallback	XmPrimitive	XmNrenderTable	XmLabel
XmNdepth	Core	XmNscreen	Core
XmNdestroyCallback	Core	XmNsensitive	Core
XmNfontList	XmLabel	XmNshadowThickness	XmPrimitive
XmNforeground	XmPrimitive	XmNstringDirection	XmLabel
XmNheight	Core	XmNtoolTipString	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNtopShadowPixmap	XmPrimitive
XmNhighlightOnEnter	XmPrimitive	XmNtranslations	Core
XmNhighlightPixmap	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNhighlightThickness	XmPrimitive	XmNunitType	XmPrimitive
XmNinitialResourcesPersistent	Core	XmNuserData	XmPrimitive
XmNlabelInsensitivePixmap	XmLabel	XmNwidth	Core
XmNlabelPixmap	XmLabel	XmNx	Core
XmNlabelString	XmLabel	XmNy	Core
XmNlabelType	XmLabel		

Translations

Event	Action
BSelect Press	Arm()
MCtrl BSelect Press	ButtonTakeFocus()
BSelect Click	Activate() Disarm()
BSelect Release	Activate() Disarm()
BSelect Press 2+	MultiArm()
BSelect Release 2+	MultiActivate()
KSelect	ArmAndActivate()
KHelp	Help()

Action Routines

DrawnButton defines the following action routines:

Activate()
: Displays the DrawnButton as unselected if XmNpushButtonEnabled is True or displays the shadow according to XmNshadowType. Invokes the list of callbacks specified by XmNactivateCallback.

Arm()
: Displays the DrawnButton as selected if XmNpushButtonEnabled is True or displays the shadow according to XmNshadowType. Invokes the list of callbacks specified by XmNarmCallback.

ArmAndActivate()
: Displays the DrawnButton as selected if XmNpushButtonEnabled is True or displays the shadow according to XmNshadowType. Invokes the list of callbacks specified by XmNarmCallback. After doing this, the action routine displays the DrawnButton as unselected if XmNpushButtonEnabled is True or displays the shadow according to XmNshadowType and invokes the list of callbacks specified by XmNactivateCallback and XmNdisarmCallback.

ButtonTakeFocus()
: In Motif 2.0 and later, moves the current keyboard focus to the DrawnButton, without activating the widget.

Disarm()
: Displays the DrawnButton as unselected and invokes the list of callbacks specified by XmNdisarmCallback.

Help()
: Invokes the list of callbacks specified by XmNhelpCallback. If the DrawnButton doesn't have any help callbacks, the Help() routine invokes those associated with the nearest ancestor that has them.

MultiActivate()
: Increments the click_count member of XmDrawnButtonCallbackStruct, displays the DrawnButton as unselected if XmNpushButtonEnabled is True or displays the shadow according to XmNshadowType, and invokes the list of callbacks specified by XmNactivateCallback and XmNdisarmCallback. This action routine takes effect only when the XmNmultiClick resource is set to XmMULTICLICK_KEEP.

MultiArm()
: Displays the DrawnButton as selected if XmNpushButtonEnabled is True or displays the shadow according to XmNshadowType, and invokes the list of callbacks specified by XmNarmCallback. This

action routine takes effect only when the XmNmultiClick resource is set to XmMULTICLICK_KEEP.

Additional Behavior

DrawnButton has the following additional behavior:

<EnterWindow>

Displays the DrawnButton as selected if XmNpushButtonEnabled is True and the pointer leaves and re-enters the window while BSelect is pressed.

<LeaveWindow>

Displays the DrawnButton as unselected if XmNpushButtonEnabled is True and the pointer leaves the window while BSelect is pressed.

See Also

`XmCreateObject`(1), `Core`(2), `XmLabel`(2), `XmPrimitive`(2), `XmPushButton`(2).

Name

XmDropDown – The DropDown Widget Class

Synopsis

Public Headers:
<Xm/DropDown.h>

Class Name:
XmDropDown

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmDropDown

Class Pointer:
xmDropDownWidgetClass

Instantiation:

```
widget = XmCreateDropDown(parent, name,...)
or
widget = XtCreateWidget(name, xmDropDownWidget-
Class,...)
```

Functions/Macros:

```
XmCreateDropDown(), XmDropDownGetLabel(),
XmDropDownGetArrow(), XmDropDownGetText(),
XmDropDownGetList(), XmDropDownGetChild()
```

Availability

OpenMotif 2.2 and later. Known as the CombinationBox2 until OpenMotif 2.3 (Provisional Widget).

Description

The DropDown widget allows the user to select elements from a list of choices, and enter their own values in a text widget. To conserve screen space, the list of choices is shown only when the user selects the down arrow button. The choices may then be selected from this list. If the list widget is in Browse Select mode (the default) or Single Select mode, then the list will automatically be removed when the user selects an item in the list. When the list is in other modes, multiple items may be selected and the list may be popped down by either another click on the arrow button, a click outside the list or double-clicking an item. When using keyboard traversal, the list may be popped down by selecting the arrow button or Alt - down arrow, and popped back up by typing either the osfActivate key, a carriage return, or Alt - up arrow. In any case, when the list is removed the item or items that were selected will be placed in the text widget separated by commas. Typing the escape key when the list is up, cancels the list popup, restoring the combination box to the state it was in before the list was popped up. If the text

field area is non-editable, clicking anywhere in the text field the list will also pop down.

Resources are available to change the margin sizes, the location of the left edge of the popup list, whether or not the label is shown, whether the text field widget may be edited, and whether or not the text in the list is verified against the choices available in the list. By setting resources that occur in the children of the combination box, the contents of the list, the number of items visible in the list, the initial value of the text field, and the value of the label may be set or changed dynamically.

If a developer wishes the popup list's shell to be resizable, they should set the allowShellResize to True at creation time.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNcustomizedCombinationBox	XmCBoolean	Boolean	False	CSG
XmNeditable	XmCBoolean	Boolean	True	CSG
XmNhorizontalMargin	XmCMargin	Dimension	2	CSG
XmNitemCount	list	int	0	CSG
XmNitems	list	XmStringTable	NULL	CSG
XmNlabelString	label	XmString	"label"	CSG
XmNpopupCursor	XmCCursor	Cursor	left_ptr	CSG
XmNpopupOffset	XmCPopupOffset	int	15	CSG
XmNpopupShellWidget	XmCWidget	Widget	NULL	CSG
XmNshowLabel	XmCBoolean	Boolean	True	CSG
XmNupdateShellCallback	XmCCallback	XtCallbackList	NULLS	CSG
XmNupdateTextCallback	XmCCallback	XtCallbackList	NULL	CSG

Name	Class	Type	Default	Access
XmNuseTextField	XmCUseTextField	Boolean	True	CSG
XmNvalue	text	String	""	CSG
XmNverify	XmCVerify	Boolean	True	CSG
XmNverifyTextCallback	XmCCallback	XtCallbackList	NULL	CSG
XmNverticalMargin	XmCMargin	Dimension	2	CSG
XmNvisibleItemCount	XmCVisibleItemCount	int	0	CSG

XmNcustomizedCombinationBox
: If this resource is True then the widget will not automatically create a popup shell and list widget. This resource can be used, as the name implies, to create a custom combination box that has something other than a list in it. If this resource is true then a shell must be provided to the combination box via the popupShellWidget resource. Just before the shell is popped up the updateShellCallback is called. Just after the shell is popped down the updateTextCallback is called. If verify is true then the verifyTextCallback is called when the combo box needs to verify the contents of the text widget against the allowable values in the custom shell.

XmNeditable
: This boolean value determines whether the user is allowed to type into the DropDown's TextField widget. If this value is False, selecting the text field will popup the combo box list XmNhorizontalMargin Specifies the horizontal spacing between the child widgets and the boundary of the DropDown.

XmNverticalMargin
: Specifies the vertical spacing between the child widgets and the boundary of the DropDown.

XmNitemCount
: The number of items in the popup list.

XmNitems
: The list of all choice that will be displayed in the popup list.

XmNlabelString
: The string displayed as the label of the DropDown.

XmNpopupCursor
: The cursor to display to the user when the DropDown's list is popped up. See X Window System, Robert Scheiffler et al., Appendix B, for choices. 2 DropDown(library call) DropDown(library call)

XmNpopupOffset
: The amount of space in pixels between the left edge of the Text widget and the left edge of the list when the list is displayed. Positive values mean the text's left edge is farther to the left, negative values mean the list's edge is farther to the

left. If this is a non-custom combination box the right edge of the text and the right edge of the arrow button will always line up.

XmNpopupShellWidget
: The widget identifier for the shell that is popped up when the arrow is clicked. If customized- DropDown is False then this widget is automatically created by the DropDown.

XmNshowLabel
: A boolean value that specifies whether or not to display the DropDownLabel widget.

XmNupdateShellCallback
XmNupdateTextCallback
: The callback routine to call when either the shell widget contents or the Text widget need to be updated to correspond with the other. The shell is updated just before it is popped up. The text is updated just after the shell is popped down. If customized DropDown is False then the updates are done automatically by the combo box. These routines are called to inform the application that an action has been taken, in case it would like to do any further processing.

XmNuseTextField
: A boolean value that specifies if an XmTextField or an XmText widget should be used for the text entry portion of the combination box.

XmNvalue
: The string displayed in the Text widget.

XmNverify
: If this resource is true the DropDown will verify its value against the list whenever it loses focus or the user types <Carriage Return>. If the verification fails, an XtWarning is generated with a name of XmNtextVerifyFailed. To trap this message register an XtWarningMsgHandler.

XmNverifyTextCallback
: This routine is called whenever the Text widget's contents may need to be verified against the popup shell's contents. If the customized DropDown resource is False then the DropDown has already performed the verification when this routine is called.

XmNvisibleItemCount
: The number of items visible in the popup list at one time.

Inherited Resources

XmDropDown inherits behavior and resources from the superclass described in the following table. For a complete description of each resource, refer to the reference page for that superclass.

Resource	Inherited from	Resource	Inherited from
XmNbottomShadowColor	XmManager	XmNaccelerators	Core
XmNbottomShadowPixmap	XmManager	XmNancestorSensitive	Core
XmNforeground	XmManager	XmNbackground	Core
XmNhelpCallback	XmManager	XmNbackgroundPixmap	Core
XmNhighlightColor	XmManager	XmNborderColor	Core
XmNhighlightPixmap	XmManager	XmNborderPixmap	Core
XmNinitialFocus	XmManager	XmNborderWidth	Core
XmNlayoutDirection	XmManager	XmNcolormap	Core
XmNnavigationType	XmManager	XmNdepth	Core
XmNpopupHandlerCallback	XmManager	XmNdestroyCallback	Core
XmNshadowThickness	XmManager	XmNheight	Core
XmNstringDirection	XmManager	XmNinitialResourcesPersistent	Core
XmNtopShadowColor	XmManager	XmNmappedWhenManaged	Core
XmNtopShadowPixmap	XmManager	XmNscreen	Core
XmNtraversalOn	XmManager	XmNsensitive	Core
XmNunitType	XmManager	XmNtranslations	Core
XmNuserData	XmManager	XmNwidth	Core
XmNchildren	Composite	XmNx	Core
XmNinsertPosition	Composite	XmNy	Core
XmNnumChildren	Composite		

See Also

XmComboBox, Composite(2), Constraint(2), Core(3), XmCreateObject(1), XmManager(3), XmTextField(2).

Name

XmDropSite registry – an object that defines the characteristics of a drop site.

Synopsis

Public Header:
<Xm/DragDrop.h>

Class Hierarchy:
DropSite does not inherit from any widget class.

Instantiation:
`XmDropSiteRegister`(...)

Functions/Macros:
`XmDropSiteConfigureStackingOrder`(), `XmDropSiteEndUpdate`(), `XmDropSiteQueryStackingOrder`(), `XmDropSiteRegister`(), `XmDropSiteRetrieve`(), `XmDropSiteStartUpdate`(), `XmDropSiteUpdate`(), `XmDropSiteUnregister`()

Availability

Motif 1.2 and later.

Description

An XmDropSite is an object that stores data about a drop site for drag and drop operations. A DropSite is associated with a particular widget or gadget in an application. An application registers a widget or gadget as a DropSite using `XmDropSiteRegister`(). The DropSite stores information about the shape of the drop site, the animation effects used when the pointer enters the drop site, the types of data supported by the drop site, and the callback that is activated when a drop occurs. These characteristics can be specified as resources when the DropSite is created.

The functions `XmDropSiteUpdate`() and `XmDropSiteRetrieve`() set and get the drop site resources for a widget that is registered as a DropSite. Use these routines instead of `XtSetValues`() and `XtGetValues`().

New Resources

DropSite defines the following resources:

Name	Class	Type	Default	Access
XmNanimationMask	XmCAnimationMask	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNanimationPixmap	XmCAnimationPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG

Name	Class	Type	Default	Access
XmNanimationPixmapDepth	XmCAnimationPixmapDepth	int	0	CSG
XmNanimationStyle	XmCAnimationStyle	unsigned char	XmDRAG_UNDER_HIGHLIGHT	CSG
XmNdropRectangles	XmCDropRectangles	XRectangle *	dynamic	CSG
XmNdropSiteActivity	XmCDropSiteActivity	unsigned char	XmDROP_SITE_ACTIVE	CSG
XmNdropSiteOperations	XmCDropSiteOperations	unsigned char	XmDROP_MOVE | XmDROP_COPY	CSG
XmNdropSiteType	XmCDropSiteType	unsigned char	XmDROP_SITE_SIMPLE	CG
XmNimportTargets	XmCImportTargets	Atom *	NULL	CSG
XmNnumDropRectangles	XmCNumDropRectangles	Cardinal	1	CSG
XmNnumImportTargets	XmCNumImportTargets	Cardinal	0	CSG

XmNanimationMask
The mask for the XmNanimationPixmap when the animation style is XmDRAG_UNDER_PIXMAP.

XmNanimationPixmap
The pixmap used for drag-under animation when the animation style is XmDRAG_UNDER_PIXMAP.

XmNanimationPixmapDepth
The depth of the pixmap specified by XmNanimationPixmap.

XmNanimationStyle
The style of drag-under animation used when the pointer enters a valid drop site during a drag operation. Possible values:

```
XmDRAG_UNDER_HIGHLIGHT      /* drop site highlighted                */
XmDRAG_UNDER_SHADOW_OUT     /* drop site shown with outset shadow */
XmDRAG_UNDER_SHADOW_IN      /* drop site shown with inset shadow */
XmDRAG_UNDER_PIXMAP         /* drop site displays pixmap            */
XmDRAG_UNDER_NONE           /* no animation effects unless in XmNdragProc */
```

XmNdropRectangles
A list of rectangles that specify the shape of the drop site. When the value is NULL, the drop site is the entire widget.

XmNdropSiteActivity
Specifies the state of the drop site. Possible values:

```
XmDROP_SITE_ACTIVE          /* participates in drop operations */
```

```
XmDROP_SITE_INACTIVE      /* does not participate in drop operations */
```

XmNdropSiteOperations
The valid operations for a drop site. The value is a bit mask that is formed by combining one or more of these possible values:

```
XmDROP_COPY               /* copy operations are valid */
XmDROP_LINK               /* link operations are valid */
XmDROP_MOVE               /* move operations are valid */
XmDROP_NOOP               /* no operations are valid   */
```

XmNdropSiteType
The type of the drop site. Possible values:

```
XmDROP_SITE_SIMPLE        /* no children are registered as drop sites */
XmDROP_SITE_COMPOSITE     /* has children registered as drop sites    */
```

XmNimportTargets
The list of target atoms that the drop site accepts.

XmNnumDropRectangles
The number of rectangles in the XmNdropRectangles list.

XmNnumImportTargets
The number of atoms in the XmNimportTargets list.

Callback Resources

DropSite defines the following callback resources:

Callback	Reason Constant
XmNdragProc	XmCR_DROP_SITE_ENTER_MESSAGE XmCR_DROP_SITE_LEAVE_MESSAGE XmCR_DROP_SITE_MOTION_MESSAGE XmCR_OPERATION_CHANGED_MESSAGE
XmNdropProc	XmCR_DROP_MESSAGE

XmNdragProc
The procedure that is called when the drop site receives a crossing, motion, or operation changed message under the dynamic protocol. The reason passed to the procedure depends on the type of message that is received.

XmNdropProc
The procedure that is called when a drop operation occurs on the drop site.

Callback Structure

The XmNdragProc is passed the following structure:

```
typedef struct {
    int             reason;             /* reason the callback was called      */
    XEvent          *event;             /* event that triggered callback       */
    Time            timeStamp;          /* timestamp of logical event          */
    Widget          dragContext;        /* DragContext associated with operation */
    Position        x;                  /* x-coordinate of pointer             */
    Position        y;                  /* y-coordinate of pointer             */
    unsigned char   dropSiteStatus;     /* valid or invalid                    */
    unsigned char   operation;          /* current operation                   */
    unsigned char   operations;         /* supported operations                */
    Boolean         animate;            /* toolkit or receiver does animation  */
} XmDragProcCallbackStruct, *XmDragProcCallback;
```

The XmNdragProc can change the value of the dropSiteStatus, operation, and operations fields in this structure. When the drag procedure completes, the toolkit uses the resulting values to initialize the corresponding fields in the callback structure passed to the initiating client's callbacks.

The XmNdropProc is passed the following structure:

```
typedef struct {
    int             reason;             /* reason the callback was called      */
    XEvent          *event;             /* event that triggered callback       */
    Time            timeStamp;          /* timestamp of logical event          */
    Widget          dragContext;        /* DragContext associated with operation */
    Position        x;                  /* x-coordinate of pointer             */
    Position        y;                  /* y-coordinate of pointer             */
    unsigned char   dropSiteStatus;     /* valid or invalid                    */
    unsigned char   operation;          /* current operation                   */
    unsigned char   operations;         /* supported operations                */
    unsigned char   dropAction;         /* drop or help                        */
} XmDropProcCallbackStruct, *XmDropProcCallback;
```

The XmNdropProc can change the value of the dropSiteStatus, operation, operations, and dropAction fields in this structure. When the drop procedure completes, the toolkit uses the resulting values to initialize the corresponding fields in the XmDropProcCallbackStruct callback structure passed to the initiating client's drop start callbacks.

The dropSiteStatus field in these structures specifies whether or not the drop site is valid. The toolkit initializes the value based on the XmNimportTargets resource of the DropSite and the XmNexportTargets resource of the DragContext. The possible values are XmDROP_SITE_VALID and XmDROP_SITE_INVALID.

The operations field in these structure specifies the set of operations supported for the data being dragged. The toolkit initializes the value based on the XmNdragOperations resource of the DragContext and the operation selected by the user. The operation field in these structures specifies the current operation. The toolkit initializes the value based on the value of operations and the XmNdropSiteOperations resource.

The animate field in the XmDragProcCallbackStruct specifies whether the toolkit or the receiving client handles the drag-under effects for the drop site. If the value is True, the toolkit handles the effects based on the XmNanimationStyle resource. Otherwise the receiver is responsible for providing drag-under effects.

The dropAction field in the XmDropProcCallbackStruct specifies the action associated with the drop, which is either a normal drop or a help action. The possible values are XmDROP and XmDROP_HELP.

See Also

`XmDropSiteConfigureStackingOrder`(1), `XmDropSiteEndUpdate`(1), `XmDropSiteQueryStackingOrder`(1), `XmDropSiteRegister`(1), `XmDropSiteRetrieve`(1), `XmDropSiteStartUpdate`(1), `XmDropSiteUnregister`(1), `XmDropSiteUpdate`(1), `XmDisplay`(1), `XmDragContext`(1), `XmDragIcon`(2), `XmDropTransfer`(2), `XmScreen`(2).

Name

XmDropTransfer widget class – an object used to store information about a drop transaction.

Synopsis

Public Header:
<Xm/DragDrop.h>

Class Name:
XmDropTransfer

Class Pointer:
xmDropTransferObjectClass

Class Hierarchy:
Object → DropTransfer

Instantiation:
widget = `XmDropTransferStart`(...)

Functions/Macros:
`XmDropTransferAdd`(), `XmDropTransferStart`()

Availability

Motif 1.2 and later.

Description

The XmDropTransfer object stores information that the toolkit needs to process a drop transaction. An application does not explicitly create a DropTransfer widget, but instead initiates a data transfer by calling `XmDropTransfer-Start`(), which initializes and returns a DropTransfer widget. If `XmDrop-TransferStart`() is called within an XmNdropProc, the data transfer starts after the callback returns. If no data needs to be transferred or the drop transaction is a failure, an application still needs to call `XmDropTransferStart`() with a failure status, so that the toolkit can complete the drag and drop operation.

The XmNtransferProc resource specifies a procedure of type XtSelectionCallbackProc that handles transferring the requested selection data. This procedure performs in conjunction with the underlying Xt selection mechanisms and is called for each type of data being transferred. Target types can be added after a transfer has started by calling the `XmDropTransferAdd`().

New Resources

DropTransfer defines the following resources:

Name	Class	Type	Default	Access
XmNdropTransfers	XmCDropTransfers	XmDropTransferEntryRec *	NULL	CG

Name	Class	Type	Default	Access
XmNincremental	XmCIncremental	Boolean	False	CSG
XmNnumDropTransfers	XmCNumDropTransfers	Cardinal	0	CSG
XmNtransferProc	XmCTransferProc	XtSelectionCallbackProc	NULL	CSG
XmNtransferStatus	XmCTransferStatus	unsigned char	XmTRANSFER_SUCCESS	CSG

XmNdropTransfers
Pointer to an array of XmDropTransferEntryRec structures, which specifies the requested target data types for the source data. A XmDropTransferEntryRec is defined as follows:

```
typedef struct {
    XtPointer    client_data;    /* any additional information necessary */
    Atom         target;         /* the selection target type            */
} XmDropTransferEntryRec, *XmDropTransferEntry;
```

The drop transfer is done when all of the entries have been processed.

XmNincremental
If True, the receiver uses the Xt incremental selection transfer mechanism. If False (default), the receiver uses atomic transfer.

XmNnumDropTransfers
The number of entries in XmNdropTransfers. The transfer is complete if the value is set to 0 at any time.

XmNtransferProc
A procedure of type XtSelectionCallbackProc that provides the requested selection values. The widget argument passed to this procedure is the DropTransfer widget and the selection atom is _MOTIF_DROP.

XmNtransferStatus
The current status of the drop transfer. The receiving client updates this value when the transfer ends and the value is communicated to the initiator. Possible values:

XmTRANSFER_SUCCESS XmTRANSFER_FAILURE

Inherited Resources

DropTransfer inherits the following resource:

Resource	Inherited From
XmNdestroyCallback	Object

See Also

XmDropTransferAdd(1), XmDropTransferStart(1), XmTargetsAreCompatible(1), Object(2), XmDisplay(2), XmDragContext(2), XmDragIcon(2), XmDropTransfer(2), XmScreen(2).

Name

XmErrorDialog – an unmanaged MessageBox as a child of a DialogShell.

Synopsis

Public Header:
<Xm/MessageB.h>

Instantiation:
widget = `XmCreateErrorDialog`(...)

Functions/Macros:
`XmCreateErrorDialog`(), `XmMessageBoxGetChild`()

Description

An XmErrorDialog is a compound object created by a call to XmCreateErrorDialog() that an application can use to inform the user about any type of error. An ErrorDialog consists of a DialogShell with an unmanaged MessageBox widget as its child. The MessageBox resource XmNdialogType is set to XmDIALOG_ERROR. An ErrorDialog includes four components: a symbol, a message, three buttons, and a separator between the message and the buttons. By default, the symbol is an octagon with a diagonal slash. In Motif 1.2, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Cancel**, and **Help** by default.

Default Resource Values

An ErrorDialog sets the following default values for MessageBox resources:

Name	Default
XmNdialogType	XmDIALOG_ERROR
XmNsymbolPixmap	xm_error

Widget Hierarchy

When an ErrorDialog is created with a specified name, the DialogShell is named name*_popup* and the MessageBox is called *name***.**

See Also

`XmCreateObject`(1), `XmMessageBoxGetChild`(1), `XmDialogShell`(2), `XmMessageBox`(2).

Name

XmFileSelectionBox widget class – a widget for selecting files.

Synopsis

Public Header:
<Xm/FileSB.h>

Class Name:
XmFileSelectionBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmBulletinBoard → XmSelectionBox → XmFileSelectionBox

Class Pointer:
xmFileSelectionBoxWidgetClass

Instantiation:
widget = `XmCreateFileSelectionBox` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmFileSelectionBoxWidgetClass,...)

Functions/Macros:
`XmCreateFileSelectionBox`(), `XmCreateFileSelectionDialog`(),
`XmFileSelectionBoxGetChild`(), `XmFileSelectionDoSearch`(),
`XmIsFileSelectionBox`()

Description

FileSelectionBox is a composite widget that is used to traverse a directory hierarchy and select files. FileSelectionBox provides a directory mask input field, a scrollable list of subdirectories, a scrollable list of filenames, a filename input field, and a group of four PushButtons. The names for the filter text, directory list, and directory list label are Text, DirList, and Dir respectively. The other components have the same names as the components in a SelectionBox.

In Motif 1.2, the button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Filter**, **Cancel**, and **Help** by default.

You can customize a FileSelectionBox by removing existing children or adding new children. Use `XmFileSelectionBoxGetChild()` to retrieve the widget ID of an existing child and then unmanage the child. With Motif 1.2, multiple widgets can be added as children of a FileSelectionBox. Additional children are added in the same way as for a SelectionBox. In Motif 1.1, only a single widget can be added as a child of a FileSelectionBox. This child is placed below the filename input field and acts as a work area.

In Motif 2.0 and later, the search pattern and base directory can be displayed in two separate text fields, depending on the value of the XmNpathMode resource. If the value is XmPATH_MODE_FULL, the behavior is consistent with that of Motif 1.2, and the filter text field (Text) contains the XmNdirMask resource. If the value is XmPATH_MODE_RELATIVE, the XmNdirectory resource is displayed in an additional text field which has the name **DirText**, with an accompanying label named **DirL**, and the filter text field **Text** contains the XmNpattern resource.

In some variants of CDE Motif, the directory pattern field may be replaced with an XmComboBox, providing a validset of directory locations. If the resource XmNenableFdbPickList is true, the FileSelectionBox creates an XmComboBox called **DirComboBox** in place of the **DirText** field.[1]

Traits

FileSelectionBox uses the XmQTactivatable trait.

New Resources

FileSelectionBox defines the following resources:

Name	Class	Type	Default	Access
XmNdirectory	XmCDirectory	XmString	dynamic	CSG
XmNdirectoryValid	XmCDirectoryValid	Boolean	dynamic	SG
XmNdirListItems	XmCDirListItems	XmStringTable	dynamic	SG
XmNdirListItemCount	XmCDirListItemCount	int	dynamic	SG
XmNdirListLabelString	XmCDirListLabelString	XmString	dynamic	CSG
XmNdirMask	XmCDirMask	XmString	dynamic	CSG
XmNdirSearchProc	XmCDirSearchProc	XmSearchProc	default procedure	CSG
XmNdirSpec	XmCDirSpec	XmString	dynamic	CSG
XmNdirTextLabelString	XmCDirTextLabelString	XmString	NULL	CSG
XmNfileFilterStyle	XmCFileFilterStyle	XtEnum	XmFILTER_NONE	SG

1.XmNenableFdbPickList is implemented on Solaris 2.7 and above.

Name	Class	Type	Default	Access
XmNfileListItems	XmCItems	XmStringTable	dynamic	SG
XmNfileListItemCount	XmCItemCount	int	dynamic	SG
XmNfileListLabelString	XmCFileListLabelString	XmString	dynamic	SG
XmNfileSearchProc	XmCFileSearchProc	XmSearchProc	default procedure	CSG
XmNfileTypeMask	XmCFileTypeMask	unsigned char	XmFILE_REGULAR	CSG
XmNfilterLabelString	XmCFilterLabelString	XmString	dynamic	CSG
XmNlistUpdated	XmCListUpdated	Boolean	dynamic	SG
XmNnoMatchString	XmCNoMatchString	XmString	XmUNSPECIFIED[a]	CSG
XmNpathMode	XmCPathMode	XtEnum	XmPATH_MODE_FULL	CSG
XmNpattern	XmCPattern	XmString	dynamic	CSG
XmNqualifySearchDataProc	XmCQualifySearchDataProc	XmQualifyProc	default procedure	CSG

a. Strictly speaking, more correct than the " [ ] " given in the 1st and 2nd editions. If the value is XmUNSPECIFIED, it defaults to this expression.

XmNdirectory
: The base directory that, in combination with XmNpattern, forms the directory mask (the XmNdirMask resource). The directory mask determines which files and directories to display.

XmNdirectoryValid
: A resource that can be set only by the directory search procedure (as specified by the XmNdirSearchProc resource). If the directory search procedure is unable to search the directory that was passed to it, then it will set XmNdirectoryValid to False, and as a result, the file search procedure won't be called.

XmNdirListItems
: The items in the directory list. This resource is set only by the directory search procedure. A call to XtGetValues() returns the actual list items (not a copy), so don't have your application free these items.

XmNdirListItemCount
: The number of items in XmNdirListItems. This resource is set only by the directory search procedure.

XmNdirListLabelString
: The string that labels the directory list. In Motif 1.2, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Directories".

XmNdirMask
: The directory mask that determines which files and directories to display. This value combines the values of the resources XmNdirectory and XmNpattern.

XmNdirSearchProc
The procedure that performs directory searches. For most applications, the default procedure works just fine. The call to this procedure contains two arguments: the widget ID of the FileSelectionBox and a pointer to an XmFileSelectionBoxCallbackStruct.

XmNdirSpec
The complete specification of the file path. Synonymous with the XmNtextString resource in SelectionBox. It is the initial directory and file search that determines the default value for this resource.

XmNdirTextLabelString
In Motif 2.0 and later, specifies the label for the directory text field when the XmNpathMode resource is XmPATH_MODE_RELATIVE. The value is otherwise ignored.

XmNfileFilterStyle
In Motif 2.0 and later, controls the behaviour of the default file and directory search procedures in the way in which hidden files are displayed. If enabled, any file or directory beginning with '.' is filtered out. The exception to the rule is ".." which is not filtered out in the directory search procedure. Possible values:

```
XmFILTER_NONE           /* do not filter out any files or directories */
XmFILTER_HIDDEN_FILES   /* filter out file beginning with '.'         */
```

XmNfileListItems
The items in the file list. Synonymous with the XmNlistItems resource in SelectionBox. This resource is set only by the file search procedure. A call to `XtGetValues`() returns the actual list items (not a copy), so don't have your application free these items.

XmNfileListItemCount
The number of items in XmNfileListItems. Synonymous with the XmNlistItemCount resource in SelectionBox. This resource is set only by the file search procedure.

XmNfileListLabelString
The string that labels the file list. Synonymous with the XmNlistLabelString resource in SelectionBox. In Motif 1.2, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Files".

XmNfileSearchProc
The procedure that performs file searches. For most applications, the default procedure works just fine. The call to this procedure contains two arguments: the widget ID of the FileSelectionBox and a pointer to an XmFileSelectionBoxCallbackStruct.

XmNfileTypeMask
Determines whether the file list will display only regular files, only directories, or any type of file. Possible values are XmFILE_DIRECTORY, XmFILE_REGULAR, and XmFILE_ANY_TYPE.

XmNfilterLabelString
The string that labels the field in which the directory mask is typed in by the user. In Motif 1.2, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Filter".

XmNlistUpdated
A resource that can be set only by the directory search procedure or by the file search procedure. This resource is set to True if the directory or file list was updated by a search procedure.

XmNnoMatchString
A string that displays in the file list when there are no filenames to display.

XmNpathMode
In Motif 2.0 and later, specifies the way in which the filter string is presented in the file selection box. The layout can either contain a single text field for the filter, as specified by the XmNdirMask resource, or two separate text fields containing the XmNpattern and XmNdirectory resources. Possible values:

```
XmPATH_MODE_FULL     /* single text field for XmNdirMask          */
XmPATH_MODE_RELATIVE /* 2 text fields for XmNpattern/XmNdirectory
*/
```

When XmNpathMode is XmPATH_MODE_RELATIVE, the text field associated with the XmNpattern resource is labelled using the value of the XmNfilterLabelString resource, and the text field associated with the XmNdirectory resource is labelled from the XmNdirTextLabelString value.

XmNpattern
The file search pattern that, in combination with XmNdirectory, forms the directory mask (the XmNdirMask resource). The directory mask determines which files and directories to display. If the XmNpattern resource defaults to NULL or is empty, a pattern for matching all files will be used.

XmNqualifySearchDataProc
The procedure that generates a valid directory mask, base directory, and search pattern to be used by XmNdirSearchProc and XmNfileSearchProc (the search procedures for directories and files). For most applications, the default procedure works just fine. The call to this procedure contains three arguments: the widget ID of the FileSelectionBox, a pointer to an XmFileSelectionBoxCallbackStruct containing the input data, and a pointer to an XmFileSelectionBoxCallbackStruct that will contain the output data.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* reason that the callback was called */
    XEvent      *event;         /* event that triggered callback       */
    XmString    value;          /* current value of XmNdirSpec resource */
    int         length;         /* number of bytes in value member     */
    XmString    mask;           /* current value of XmNdirMask resource */
    int         mask_length;    /* number of bytes in mask member      */
    XmString    dir;            /* current base directory              */
    int         dir_length;     /* number of bytes in dir member       */
    XmString    pattern;        /* current search pattern              */
    int         pattern_length; /* number of bytes in pattern member   */
} XmFileSelectionBoxCallbackStruct;
```

Inherited Resources

FileSelectionBox inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. FileSelectionBox sets the default values of XmNautoUnmanage to False and XmNdialogType to XmDIALOG_FILE_SELECTION. It also sets the default values of XmNlistItems and XmNlistItemCount dynamically. The default value of XmNborderWidth is reset to 0 by Manager. BulletinBoard sets the value of XmNinitialFocus to XmNdefaultButton and resets the default XmNshadowThickness from 0 to 1 if the FileSelectionBox is a child of a DialogShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNlistItemCount	XmSelectionBox
XmNallowOverlap	XmBulletinBoard	XmNlistItems	XmSelectionBox
XmNancestorSensitive	Core	XmNlistLabelString	XmSelectionBox
XmNapplyCallback	XmSelectionBox	XmNlistVisibleItemCount	XmSelectionBox
XmNapplyLabelString	XmSelectionBox	XmNmapCallback	XmBulletinBoard
XmNautoUnmanage	XmBulletinBoard	XmNmappedWhenManaged	Core
XmNbackground	Core	XmNmarginHeight	XmBulletinBoard
XmNbackgroundPixmap	Core	XmNmarginWidth	XmBulletinBoard
XmNborderColor	Core	XmNminimizeButtons	XmSelectionBox
XmNborderPixmap	Core	XmNmustMatch	XmSelectionBox
XmNborderWidth	Core	XmNnavigationType	XmManager
XmNbottomShadowColor	XmManager	XmNnoMatchCallback	XmSelectionBox
XmNbottomShadowPixmap	XmManager	XmNnoResize	XmBulletinBoard

Resource	Inherited From	Resource	Inherited From
XmNbuttonFontList	XmBulletinBoard	XmNnumChildren	Composite
XmNbuttonRenderTable	XmBulletinBoard	XmNokCallback	XmSelectionBox
XmNcancelButton	XmBulletinBoard	XmNokLabelString	XmSelectionBox
XmNcancelCallback	XmSelectionBox	XmNpopupHandlerCallback	XmManager
XmNcancelLabelString	XmSelectionBox	XmNresizePolicy	XmBulletinBoard
XmNchildren	Composite	XmNscreen	Core
XmNchildPlacement	XmSelectionBox	XmNselectionLabelString	XmSelectionBox
XmNcolormap	Core	XmNsensitive	Core
XmNdefaultButton	XmBulletinBoard	XmNshadowThickness	XmManager
XmNdefaultPosition	XmBulletinBoard	XmNshadowType	XmBulletinBoard
XmNdepth	Core	XmNstringDirection	XmManager
XmNdestroyCallback	Core	XmNtextAccelerators	XmSelectionBox
XmNdialogStyle	XmBulletinBoard	XmNtextColumns	XmSelectionBox
XmNdialogTitle	XmBulletinBoard	XmNtextFontList	XmBulletinBoard
XmNdialogType	XmSelectionBox	XmNtextRenderTable	XmBulletinBoard
XmNfocusCallback	XmBulletinBoard	XmNtextString	XmSelectionBox
XmNforeground	XmManager	XmNtextTranslations	XmBulletinBoard
XmNheight	Core	XmNtopShadowColor	XmManager
XmNhelpCallback	XmManager	XmNtopShadowPixmap	XmManager
XmNhelpLabelString	XmSelectionBox	XmNtranslations	Core
XmNhighlightColor	XmManager	XmNtraversalOn	XmManager
XmNhighlightPixmap	XmManager	XmNunitType	XmManager
XmNinitialFocus	XmManager	XmNunmapCallback	XmBulletinBoard
XmNinitialResourcesPersistent	Core	XmNuserData	XmManager
XmNinsertPosition	Composite	XmNwidth	Core
XmNlabelFontList	XmBulletinBoard	XmNx	Core
XmNlabelRenderTable	XmBulletinBoard	XmNy	Core
XmNlayoutDirection	XmManager		

Translations

The translations for FileSelectionBox are inherited from SelectionBox.

Action Routines

FileSelectionBox defines the following action routines:

SelectionBoxUpOrDown(flag)

Replaces the selection text or the filter text, depending on which one has the keyboard focus. That is, this action replaces either: the text string in the selection area with an item from the file list, or the text string in the directory mask (filter) area with an item from the directory list.

The value of flag determines which file list item or which directory list item is selected as the replacement string. A flag value of 0, 1, 2, or 3 selects the previous, next, first, or last item, respectively, of the appropriate list.

SelectionBoxRestore()

Replaces the selection text or the filter text, depending on which one has the keyboard focus. That is, this action replaces either: the text string in the selection area with the currently selected item in the file list (clearing the selection area if no list item is selected), or the text string in the filter area with a new directory mask (which is formed by combining the values of the XmNdirectory and XmN--pattern resources).

Additional Behavior

FileSelectionBox has the following additional behavior:

MAny KCancel

If the **Cancel** button is sensitive, invokes its XmNactivateCallback callbacks. If there is no **Cancel** button, the event is passed to the parent if it is a manager.

KActivate

In the filename text input area, first invokes the XmNactivateCallback callbacks for the text and then invokes either the XmNnoMatchCallback or the XmNokCallback callbacks based on the value of XmNmustMatch**.**

In the directory mask text input area, first invokes the XmNactivateCallback callbacks for the text and then starts a directory and file search and invokes the XmNapplyCallback callbacks.

In the directory list, invokes the XmNdefaultActionCallback callback, begins a directory and file search, and invokes the XmNapplyCallback callbacks.

In the file list, invokes XmNdefaultActionCallback and XmNokCallback callbacks.

When neither of these areas nor any button has the keyboard focus, it invokes the callbacks in either XmNnoMatchCallback or XmNokCallback depending on the value of XmNmustMatch and whether or not the selection text matches a file in the file list.

<DoubleClick>
In the directory or file list, has the same behavior as KActivate.

<Single Select> or <Browse Select>
In the directory list, composes a directory mask using the selected directory item and the current pattern. In the file list, uses the selected file item to replace the selection text.

BTransfer
In Motif 1.2, in the file or directory list, starts a drag and drop operation using the selected items in the list. If BTransfer is pressed over an unselected item, only that item is used in the drag and drop operation.

<Apply Button Activated>
Starts a directory and file search and invokes the XmNapplyCallback callbacks.

<Ok Button Activated>
Invokes either the XmNnoMatchCallback or XmNokCallback callbacks based on the value of XmNmustMatch and whether or not the selection text matches a file in the file list.

<Cancel Button Activated>
Invokes the XmNcancelCallback callbacks.

<Help Button Activated>
Invokes the XmNhelpCallback callbacks.

See Also

`XmCreateObject`(1), `XmFileSelectionBoxGetChild`(1), `XmFileSelectionDoSearch`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmBulletinBoard`(2), `XmFileSelectionDialog`(2), `XmManager`(2), `XmSelectionBox`(2).

Name

XmFileSelectionDialog – an unmanaged FileSelectionBox as a child of a Dialog Shell.

Synopsis

Public Header:
<Xm/FileSB.h>

Instantiation:
widget = XmCreateFileSelectionDialog(...)

Functions/Macros:
`XmCreateFileSelectionBox`(), `XmFileSelectionBoxGetChild`(), `XmCreateFileSelectionDialog`(), `XmFileSelectionDoSearch`(), `XmIsFileSelectionBox`()

Description

An XmFileSelectionDialog is a compound object created by a call to XmCreate-FileSelectionDialog() that an application can use to allow a user to select a file from a dialog box. A FileSelectionDialog consists of a DialogShell with an unmanaged FileSelectionBox widget as its child. The SelectionBox resource XmNdialogType is set to XmDIALOG_FILE_SELECTION.

A FileSelectionDialog provides a directory mask input field, a scrollable list of subdirectories, a scrollable list of filenames, a filename input field, and a group of four PushButtons. In Motif 1.2, the button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Filter**, **Cancel**, and **Help** by default.

Default Resource Values

A FileSelectionDialog sets the following default values for its resources:

Name	Default
XmNdialogType	XmDIALOG_FILE_SELECTION

Widget Hierarchy

When a FileSelectionDialog is created with a specified name, the DialogShell is named name*_popup* and the FileSelectionBox is called *name*.

See Also

`XmCreateObject`(1), `XmFileSelectionBoxGetChild`(1), `XmFileSelectionDoSearch`(1), `XmFileSelectionBox`(2), `XmDialogShell`(2).

Name

XmFontSelector – FontSelector widget

Synopsis

Public Header:
<Xm/FontS.h>

Class Name:
XmFontSelector

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmFontSelector

Class Pointer:
xmFontSelectorWidgetClass

Instantiation:

```
widget = XmCreateFontSelector(parent, name,...)
or
widget = XtCreateWidget(name, xmFontSelectorWidget-
Class,...)
```

Functions/Macros:

```
XmCreateFontSelector().
```

Availability

OpenMotif 2.2 and later. (Contributed Widget).

Description

The Font Selector widget allows users to easily choose a font by selecting the font family and size of the font. The bold and italic attributes may also be set for any font for which they are available. Any font may be passed to the font selector by the application as the initial value shown to the user. Advanced features greatly extend the widget's functionality.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using

XtSetValues (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmN100DPIString	XmC100DPIString	XmString	“100 dpi”	CSG
XmN75DPIstring	XmC75DPIString	XmString	“75 dpi”	CSG
XmNanyLowerString	XmCAnyLowerString	XmString	“any”	CSG
XmNanyString	XmCAnyString	XmString	“Any”	CSG
XmNboldString	XmCBoldString	XmString	“Bold”	CSG
XmNbothString	XmCBothString	XmString	“Both”	CSG
XmNcurrentFont	XmCString	String	NULL	CSG
XmNdefaultEncodingString	XmCDefaultEncodingString	String	“iso8859-1”	CSG
XmNencodingList	XmCEncodingList	StringTable	“iso8859-1”	CSG
XmNencodingString	XmCEncodingString	XmString	“Encoding”	CSG
XmNfamilyString	XmCBothString	XmString	“Family”	CSG
XmNitalicString	XmCItalicString	XmString	“Italic”	CSG
XmNmarginHeight	XmCMargin	Dimension	0	CSG
XmNmonoSpaceString	XmCMonoSpaceString	XmString	“Fixed Width Fonts”	CSG
XmNoptionString	XmCOptionString	XmString	“Options...”	CSG
XmNotherString	XmCOtherString	XmString	“Other Fonts”	CSG
XmNpropSpaceString	XmCPropSpaceString	XmString	“Proportional Fonts”	CSG
XmNsampleText	XmCSampleText	XmString	“abcdef...”	CSG
XmNScalingString	XmCScalingString	XmString	“Use Font Scaling”	CSG
XmNshowFontName	XmCShowFontName	Boolean	False	CSG
XmNshowNameString	XmCShowNameString	XmString	“Show Font Name”	CSG
XmNsizeString	XmCSizeString	XmString	“Size”	CSG
XmNspacing	XmCSpacing	Dimension	2	CSG
XmNtextRows	XmCTextRows	Dimension	8	CSG
XmNuseScaling	XmCBoolean	Boolean	True	CSG
XmNvalueChangedCallback	XmCCallback	XtCallbackList	NULL	CSG
XmNxlfdString	XmCXlfdString	XmString	“Xlfd Fonts”	CSG

XmN100DPSString
The label for the 100 DPI radio button.

XmN75DPSString
The label for the 75 DPI radio button.

XmNanyLowerString
: The label for the any button.

XmNanyString
: The label for the Any button.

XmNboldString
: The label for the Bold toggle button.

XmNbothString
: The labels for the Both radio buttons controlling both the dpi and width of the fonts displayed. The same resource is used to ensure consistent labels.

XmNcurrentFont
: This resource provides the main application input and output to the font selector. If the programmer sets the value at creation time or with XtSetValues then the currently displayed family, size, bold and italic will be changed to correspond to the values shown in the current font. Otherwise, the name of the font will be shown. The Font Selector's mode will be set to correspond to the type of font passed.
: Note: currentFont must contain 14 hyphens (-) to be considered an XLFD font. This resource is also used to retrieve the font the user has selected from the font selector. The value returned is only valid until the next time XtGetValues is called on this instance of the font selector widget.

XmNdefaultEncodingString
: This resource is the default selection from the Encoding options menu.

XmNencodingList
: This resource is the list of encodings available from the FontSelector Encoding options menu.

XmNencodingString
: This resource is the default selection from the Encoding options menu.

XmNfamilyString
: This resource is the default selection from the Family options menu.

XmNisoFontsOnly
: This resource controls and maintains the state of the iso8859-1 fonts only toggle button.

XmNitalicString
: This resource is the default selection from the Italic toggle button.

XmNmarginHeight
: The margin height for all subwidgets of the Font Selector.

XmNmonoSpaceString
: The label of the Fixed Width Fonts radio button.

XmNoptionString
: The label for the Options... push button.

XmNotherString
: The label for the Other Fonts radio button.

XmNpropSpaceString
The label for the Proportional Fonts radio button.
XmNsampleText
The string which appears in the sample text area.
XmNscalingString
The label for the Use Font Scaling toggle button.
XmNshowFontName
This boolean resource controls and maintains the state of Show Font Name toggle button.
XmNshowNameString
The label of the Show Font Name toggle button.
sizeString
The label for the Size option menu.
XmNspacing
The space between the toggle indicator and the toggle label.
XmNtextRows
This resource controls the number of rows that are shown in the text widget that displays sample text in the currently selected font. Since this is a scrolled text widget it will never dynamically change size, regardless of the font displayed. Unless the initial font is large this value should be at least 4 or the user interaction may be poor.
XmNuseScaling
This resource controls and maintains the state of the Use Font Scaling toggle button.
XmNvalueChangedCallback
The list of callbacks called when the XmNcurrentFont value is changed.
XmNxlfdString
The label for the Xlfd Fonts radio button.

Inherited Resources

Font Selector inherits behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass

Resource	Inherited from	Resource	Inherited from
XmNbottomShadowColor	XmManager	XmNaccelerators	Core
XmNbottomShadowPixmap	XmManager	XmNancestorSensitive	Core
XmNforeground	XmManager	XmNbackground	Core
XmNhelpCallback	XmManager	XmNbackgroundPixmap	Core

Resource	Inherited from	Resource	Inherited from
XmNhighlightColor	XmManager	XmNborderColor	Core
XmNhighlightPixmap	XmManager	XmNborderPixmap	Core
XmNinitialFocus	XmManager	XmNborderWidth	Core
XmNlayoutDirection	XmManager	XmNcolormap	Core
XmNnavigationType	XmManager	XmNdepth	Core
XmNpopupHandlerCallback	XmManager	XmNdestroyCallback	Core
XmNshadowThickness	XmManager	XmNheight	Core
XmNstringDirection	XmManager	XmNinitialResourcesPersistent	Core
XmNtopShadowColor	XmManager	XmNmappedWhenManaged	Core
XmNtopShadowPixmap	XmManager	XmNscreen	Core
XmNtraversalOn	XmManager	XmNsensitive	Core
XmNunitType	XmManager	XmNtranslations	Core
XmNuserData	XmManager	XmNwidth	Core
XmNchildren	Composite	XmNx	Core
XmNinsertPosition	Composite	XmNy	Core
XmNnumChildren	Composite		

Translations

XmFontSlector inherits translations from Xm Manager.

See Also

`XmCreateObject(1), Composite(2), Constraint(2), Core(2), XmManager(3).`

Name

XmForm widget class – a container widget that constrains its children.

Synopsis

Public Header:
<Xm/Form.h>

Class Name:
XmForm

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmBulletinBoard → XmForm

Class Pointer:
xmFormWidgetClass

Instantiation:
widget = `XmCreateForm` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmFormWidgetClass,...)

Functions/Macros:
`XmCreateForm`(), `XmCreateFormDialog`(), `XmIsForm`()

Description

Form is a container widget that constrains its children so as to define their layout when the Form is resized. Constraints on the children of a Form specify the attachments for each of the four sides of a child. Children may be attached to each other, to edges of the Form, or to relative positions within the Form.

New Resources

Form defines the following resources:

Name	Class	Type	Default	Access
XmNfractionBase	XmCMaxValue	int	100	CSG
XmNhorizontalSpacing	XmCSpacing	Dimension	0	CSG
XmNrubberPositioning	XmCRubberPositioning	Boolean	False	CSG
XmNverticalSpacing	XmCSpacing	Dimension	0	CSG

XmNfractionBase
The denominator part of the fraction that describes a child's relative position within a Form. The numerator of this fraction is one of the four positional constraint resources: XmNbottomPosition, XmNleftPosition, XmNrightPosition, or XmNtopPosition. For example, suppose you use the default XmNfractionBase of 100. Then, if you specify XmNtopPosition as 30, the top of the child will remain

invariably attached to a location that is 30/100 (or 30 percent) from the top of the Form. (In other words, resizing the Form's height might change the absolute position of the child's top, but not its position relative to the top of the Form.) Similarly, a value of 50 for XmNleftPosition ensures that the left side of the child is attached 50/100 from the left of the Form (or in this case, halfway between the left and right side). Note that these fractions are implemented only when the child's corresponding attachment constraint is set to XmATTACH_POSITION. (The attachment constraints are XmNbottomAttachment, XmNleftAttachment, XmNrightAttachment, and XmNtopAttachment.)

XmNhorizontalSpacing
The offset for right and left attachments.

XmNrubberPositioning
Defines the default behavior of a child's top and left side, in the absence of other settings. If this resource is False (default), the child's top and left sides are positioned using absolute values. If True, the child's top and left sides are positioned relative to the size of the Form.

XmNverticalSpacing
The offset for top and bottom attachments.

New Constraint Resources

Form defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNbottomAttachment	XmCAttachment	unsigned char	XmATTACH_NONE	CSG
XmNbottomOffset	XmCOffset	int	0	CSG
XmNbottomPosition	XmCAttachment	int	0	CSG
XmNbottomWidget	XmCWidget	Widget	NULL	CSG
XmNleftAttachment	XmCAttachment	unsigned char	XmATTACH_NONE	CSG
XmNleftOffset	XmCOffset	int	0	CSG
XmNleftPosition	XmCAttachment	int	0	CSG
XmNleftWidget	XmCWidget	Widget	NULL	CSG
XmNresizable	XmCBoolean	Boolean	True	CSG
XmNrightAttachment	XmCAttachment	unsigned char	XmATTACH_NONE	CSG
XmNrightOffset	XmCOffset	int	0	CSG
XmNrightPosition	XmCAttachment	int	0	CSG
XmNrightWidget	XmCWidget	Widget	NULL	CSG
XmNtopAttachment	XmCAttachment	unsigned char	XmATTACH_NONE	CSG
XmNtopOffset	XmCOffset	int	0	CSG

Name	Class	Type	Default	Access
XmNtopPosition	XmCAttachment	int	0	CSG
XmNtopWidget	XmCWidget	Widget	NULL	CSG

XmNbottomAttachment
The method of attachment for the child's bottom side. Each of the four attachment resources (XmNtopAttachment, XmNbottomAttachment, XmNleftAttachment, and XmNrightAttachment) has the following possible values. The comments below refer to a corresponding edge (top, bottom, left, or right) of the child widget within the Form.

```
XmATTACH_NONE              /* remains unattached                    */
XmATTACH_FORM              /* attached to same edge of Form         */
XmATTACH_OPPOSITE_FORM     /* attached to other edge of Form        */
XmATTACH_WIDGET            /* abuts an adjacent widget              */
XmATTACH_OPPOSITE_WIDGET   /* attached to other edge of adjacent    */
                           /*   widget                              */
XmATTACH_POSITION          /* relative to a dimension of Form       */
XmATTACH_SELF              /* relative to its current position and*/
                           /* to Form                               */
```

XmNbottomOffset
The distance between the child's bottom side and the object it's attached to. Offsets are absolute. Offsets are of type int and may not be resolution-independent.

XmNbottomPosition
Used in conjunction with XmNfractionBase to calculate the position of the bottom of a child, relative to the bottom of the Form. This resource has no effect unless the child's XmNbottomAttachment resource is set to XmATTACH_POSITION. (See XmNfractionBase for details.)

XmNbottomWidget
The name of the widget or gadget that serves as the attachment point for the bottom of the child. To use this resource, set the XmNbottomAttachment resource to either XmATTACH_WIDGET or XmATTACH_OPPOSITE_WIDGET.

XmNleftAttachment
The method of attachment for the child's left side.

XmNleftOffset
The distance between the child's left side and the object it's attached to. Offsets are absolute. Offsets are of type int and may not be resolution-independent.

XmNleftPosition
: Used in conjunction with XmNfractionBase to calculate the position of the left side of a child, relative to the left side of the Form. This resource has no effect unless the child's XmNleftAttachment resource is set to XmATTACH_POSITION. (See XmNfractionBase for details.)

XmNleftWidget
: The name of the widget or gadget that serves as the attachment point for the left side of the child. To use this resource, set the XmNleftAttachment resource to either XmATTACH_WIDGET or XmATTACH_OPPOSITE_WIDGET.

XmNresizable
: If True (default), a child's resize request is accepted by the Form, provided that the child isn't constrained by its attachments. That is, if both the left and right sides of a child are attached, or if both the top and bottom are attached, the resize request fails, whereas if the child has only one horizontal or one vertical attachment, the resize request is granted. If this resource is False, the child is never resized.

XmNrightAttachment
: The method of attachment for the child's right side.

XmNrightOffset
: The distance between the child's right side and the object it's attached to. Offsets are absolute. Offsets are of type int and may not be resolution-independent.

XmNrightPosition
: Used in conjunction with XmNfractionBase to calculate the position of the right side of a child, relative to the right side of the Form. This resource has no effect unless the child's XmNrightAttachment resource is set to XmATTACH_POSITION. (See XmNfractionBase for details.)

XmNrightWidget
: The name of the widget or gadget that serves as the attachment point for the right side of the child. To use this resource, set the XmNrightAttachment resource to either XmATTACH_WIDGET or XmATTACH_OPPOSITE_WIDGET.

XmNtopAttachment
: The method of attachment for the child's top side.

XmNtopOffset
: The distance between the child's top side and the object it's attached to. Offsets are absolute. Offsets are of type int and may not be resolution-independent.

XmNtopPosition
: Used in conjunction with XmNfractionBase to calculate the position of the top of a child, relative to the top of the Form. This resource has no effect unless the child's XmNtopAttachment resource is set to XmATTACH_POSITION. (See XmN-fraction-Base for details.)

XmNtopWidget

The name of the widget or gadget that serves as the attachment point for the top of the child. To use this resource, set the XmNtopAttachment resource to either XmATTACH_WIDGET or XmATTACH_OPPOSITE_WIDGET.

Inherited Resources

Form inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. Form sets the default values of XmNmarginWidth and XmN-margin-Height to 0. The default value of XmNborderWidth is reset to 0 by Manager. BulletinBoard sets the value of XmNinitialFocus to XmNdefaultButton and resets the default XmNshadowThickness from 0 to 1 if the Form widget is a child of DialogShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNlabelFontList	XmBulletinBoard
XmNallowOverlap	XmBulletinBoard	XmNlabelRenderTable	XmBulletinBoard
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNautoUnmanage	XmBulletinBoard	XmNmapCallback	XmBulletinBoard
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNmarginHeight	XmBulletinBoard
XmNborderColor	Core	XmNmarginWidth	XmBulletinBoard
XmNborderPixmap	Core	XmNnavigationType	XmManager
XmNborderWidth	Core	XmNnoResize	XmBulletinBoard
XmNbottomShadowColor	XmManager	XmNnumChildren	Composite
XmNbottomShadowPixmap	XmManager	XmNpopupHandlerCallback	XmManager
XmNbuttonFontList	XmBulletinBoard	XmNresizePolicy	XmBulletinBoard
XmNbuttonRenderTable	XmBulletinBoard	XmNscreen	Core
XmNcancelButton	XmBulletinBoard	XmNsensitive	Core
XmNchildren	Composite	XmNshadowThickness	XmManager
XmNcolormap	Core	XmNshadowType	XmBulletinBoard
XmNdefaultButton	XmBulletinBoard	XmNstringDirection	XmManager
XmNdefaultPosition	XmBulletinBoard	XmNtextFontList	XmBulletinBoard
XmNdepth	Core	XmNtextRenderTable	XmBulletinBoard
XmNdestroyCallback	Core	XmNtextTranslations	XmBulletinBoard
XmNdialogStyle	XmBulletinBoard	XmNtopShadowColor	XmManager
XmNdialogTitle	XmBulletinBoard	XmNtopShadowPixmap	XmManager
XmNfocusCallback	XmBulletinBoard	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager

Resource	Inherited From	Resource	Inherited From
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNunmapCallback	XmBulletinBoard
XmNhighlightColor	XmManager	XmNuserData	XmManager
XmNhighlightPixmap	XmManager	XmNwidth	Core
XmNinitialFocus	XmManager	XmNx	Core
XmNinitialResourcesPersistent	Core	XmNy	Core
XmNinsertPosition	Composite		

Translations

The translations for Form are inherited from XmBulletinBoard.

See Also

`XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmBulletinBoard`(2), `XmFormDialog`(2), `XmManager`(2).

Name

XmFormDialog – an unmanaged Form as a child of a DialogShell.

Synopsis

Public Header:
<Xm/Form.h>

Instantiation:
widget = XmCreateFormDialog(...)

Functions/Macros:
`XmCreateFormDialog`()

Description

An XmFormDialog is a compound object created by a call to `XmCreateFormDialog`() that is useful for creating custom dialogs. A FormDialog consists of a DialogShell with an unmanaged Form widget as its child. The FormDialog does not contain any labels, buttons, or other dialog components; these components are added by the application.

Widget Hierarchy

When a FormDialog is created with a specified name, the DialogShell is named name*_popup* and the Form is called *name*.

See Also

`XmCreateObject`(1), `XmDialogShell`(2), `XmForm`(2).

Name

XmFrame widget class –a manager widget that places a border around a single child.

Synopsis

Public Header:
<Xm/Frame.h>

Class Name:
XmFrame

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmFrame

Class Pointer:
xmFrameWidgetClass

Instantiation:
widget = `XmCreateFrame` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmFrameWidgetClass,...)

Functions/Macros:
`XmCreateFrame`(), `XmIsFrame`()

Description

Frame is a simple subclass of Manager that places a three-dimensional border around a single child. Frame is used to provide the typical Motif-style appearance for widget classes that do not have a visible frame, such as RowColumn.

As of Motif 1.2, a Frame can have two children: a work area child and a title child. The widget uses constraint resources to indicate the type of each child and to specify the alignment of the title child.

New Resources

Frame defines the following resources:

Name	Class	Type	Default	Access
XmNmarginHeight	XmCMarginHeight	Dimension	0	CSG
XmNmarginWidth	XmCMarginWidth	Dimension		CSG
XmNshadowType	XmCShadowType	unsigned char	dynamic	CSG

XmNmarginHeight
The spacing between the top or bottom of a Frame widget's child and the shadow of the Frame widget.

XmNmarginWidth
The spacing between the right or left side of a Frame widget's child and the shadow of the Frame widget.

XmNshadowType
The style in which Frame widgets are drawn. Possible values:

```
XmSHADOW_IN             /* widget appears inset           */
XmSHADOW_OUT            /* widget appears outset          */
XmSHADOW_ETCHED_IN      /* double line; widget appears inset */
XmSHADOW_ETCHED_OUT     /* double line; widget appears raised */
```

New Constraint Resources

As of Motif 1.2, Frame defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNchildType	XmCChildType	unsigned char	XmFRAME_WORKAREA_CHILD	CSG
XmNchildHorizontalAlignment	XmCChildHorizontalAlignment	unsigned char	XmALIGNMENT_BEGINNING	CSG
XmNchildHorizontalSpacing	XmCChildHorizontalSpacing	Dimension	dynamic	CSG
XmNchildVerticalAlignment	XmCChildVerticalAlignment	unsigned char	XmALIGNMENT_CENTER	CSG
XmNframeChildType	XmCFrameChildType	unsigned char	XmFRAME_WORKAREA_CHILD	CSG

XmNchildType
The type of the child. Frame supports one title and one work area child. Possible values:

```
XmFRAME_TITLE_CHILD         /* child is the title     */
XmFRAME_WORKAREA_CHILD      /* child is the work area */
XmFRAME_GENERIC_CHILD       /* child is ignored       */
```

From Motif 2.0 and later, the XmNchildType resource is deprecated, and the XmNframeChildType resource is the preferred method.

XmNchildHorizontalAlignment
The alignment (left to right) for a Frame's title. Possible values are:

XmALIGNMENT_BEGINNING
XmALIGNMENT_CENTER
XmALIGNMENT_END

XmNchildHorizontalSpacing
The minimum distance between the title text and the Frame shadow. The title is clipped to maintain this distance. The value of XmNmarginWidth is used as the default value.

XmNchildVerticalAlignment
The alignment of the Frame's title relative to the top shadow of the Frame. Possible values:

XmALIGNMENT_BASELINE_BOTTOM
XmALIGNMENT_BASELINE_TOP
XmALIGNMENT_WIDGET_TOP
XmALIGNMENT_CENTER
XmALIGNMENT_WIDGET_BOTTOM

XmNframeChildType
Introduced in Motif 2.0 as part of a rationalization in the naming of constraint resources. The behavior of the XmNframeChildType resource is identical in all respects to the XmNchildType resource, which is now deprecated.

Inherited Resources

Frame inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. Frame sets the default value of XmNshadowThickness to 1 if the Frame is a child of a Shell and 2 otherwise. The default value of XmNborderWidth is reset to 0 by Manager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager

Resource	Inherited From	Resource	Inherited From
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Translations

The translations for Frame are inherited from XmManager.

See Also

`XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmManager`(2).

Name

XmGadget widget class – the fundamental class for windowless widgets.

Synopsis

Public Header:
<Xm/Xm.h>

Class Name:
XmGadget

Class Hierarchy:
Object → RectObj → XmGadget

Class Pointer:
xmGadgetClass

Instantiation:
Gadget is a meta-class and is not normally instantiated.

Functions/Macros:
`XmIsGadget`()

Description

Gadget is a supporting superclass for other gadget classes. Gadget takes care of drawing and highlighting border shadows as well as managing traversal.

In versions of Motif prior to 2.0, a gadget is drawn using pixmap and color resources taken from the Manager parent. Changing such a resource in a Manager (for example, XmNforeground) also affects all gadget children. Gadgets sharing the same parent therefore also share the same general appearance.

In Motif 2.0 and later, the Gadget class supports independent appearance resources. For example, Gadgets sharing the same parent can have different XmNforeground values. Where a particular appearance resource is unspecified for a Gadget, the default value is taken from the Manager parent.

Traits

Gadget holds the XmQTspecifyLayoutDirection, XmQTaccessColors, and XmQTspecifyUnitType traits, which are inherited by any derived classes, and uses the XmQTspecifyUnhighlight trait.

New Resources

Gadget defines the following resources:

Name	Class	Type	Default	Access
XmNbackground	XmCBackground	Pixel	dynamic	CSG
XmNbackgroundPixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNbottomShadowColor	XmCBottomShadowColor	Pixel	dynamic	CSG
XmNbottomShadowPixmap	XmCBottomShadowPixmap	Pixmap	dynamic	CSG
XmNforeground	XmCForeground	Pixel	dynamic	CSG
XmNhighlightColor	XmCHighlightColor	Pixel	dynamic	CSG
XmNhighlightOnEnter	XmCHighlightOnEnter	Boolean	False	CSG
XmNhighlightPixmap	XmCHighlightPixmap	Pixmap	dynamic	CSG
XmNhighlightThickness	XmCHighlightThickness	Dimension	dynamic	CSG
XmNlayoutDirection	XmCLayoutDirection	unsigned char	dynamic	CG
XmNnavigationType	XmCNavigationType	unsigned char	XmNONE	CSG
XmNshadowThickness	XmNShadowThickness	Dimension	dynamic	CSG
XmNtoolTipString	XmCToolTipString	XmString	NULL	CSG
XmNtopShadowColor	XmCTopShadowColor	Pixel	dynamic	CSG
XmNtopShadowPixmap	XmCTopShadowPixmap	Pixmap	dynamic	CSG
XmNtraversalOn	XmCTraversalOn	Boolean	True	CSG
XmNunitType	XmCUnitType	unsigned char	dynamic	CSG
XmNuserData	XmCUserData	XtPointer	NULL	CSG

XmNbackground
In Motif 2.0 and later, specifies the background color for the Gadget.

XmNbackgroundPixmap
In Motif 2.0 and later, specifies the background pixmap for tiling the Gadget. The default is XmUNSPECIFIED_PIXMAP.

XmNbottomShadowColor
Specifies the color used in drawing the border shadow's bottom and right sides.

XmNbottomShadowPixmap
In Motif 2.0 and later, specifies the pixmap for drawing the bottom and right sides of the Gadget border shadow.

XmNforeground
In Motif 2.0 and later, specifies the foreground color for the Gadget.

XmNhighlightColor
Specifies the color used in drawing the highlighting rectangle.

XmNhighlightOnEnter
Determines whether to draw a gadget's highlighting rectangle whenever the cursor moves into the gadget. This resource applies only when the shell has a focus policy of XmPOINTER. If the XmNhighlightOnEnter resource is True, highlighting is drawn; if False (default), highlighting is not drawn.

XmNhighlightPixmap
In Motif 2.0 and later, specifies the pixmap used for drawing the highlighting rectangle.

XmNhighlightThickness
The thickness of the highlighting rectangle. In Motif 2.0 and earlier, the default is 2. In Motif 2.1 and later, the default depends upon the XmDisplay XmNenableThinThickness resource: if True, the default is 1, otherwise 2.

XmNlayoutDirection
In Motif 2.0 and later, specifies the direction in which components (for example, strings) of the Gadget are laid out. If unspecified, the value is inherited from the Manager parent, or from the nearest ancestor which has the XmQTspecifyLayoutDirection trait. Possible values:

XmLEFT_TO_RIGHT XmRIGHT_TO_LEFT

XmNnavigationType
Determines the way in which gadgets are to be traversed during keyboard navigation. Possible values:

```
XmNONE                  /* exclude from keyboard navigation */
                        /*    (default for non-shell parent)    */
XmTAB_GROUP             /* include in keyboard navigation     */
                        /*    (default when parent is a shell)  */
XmSTICKY_TAB_GROUP      /* include in keyboard navigation, even if */
                        /* XmAddTabGroup() was called          */
XmEXCLUSIVE_TAB_GROUP   /* application defines order of navigation */
```

XmNshadowThickness
The thickness of the shadow border. In Motif 2.0 and earlier, the default is 2. In Motif 2.1 and later, the default depends upon the XmDisplay XmNenableThinThickness resource: if True, the default is 1, otherwise 2.

XmNtoolTipString
The XmString to display as the tool Tip. If this resource is NULL, no tip will be displayed. ToolTips are described in VendorShell(3).

XmNtopShadowColor
Specifies the color used in drawing the border shadow's top and left sides.

XmNtopShadowPixmap
In Motif 2.0 and later, specifies the pixmap used in drawing the border shadow's top and left sides.

XmNtraversalOn
If True (default), traversal of this gadget is made possible.

XmNunitType
The measurement units to use in resources that specify a size or position--for example, any resources of data type Dimension (whose names generally include one of the words "Margin" or "Thickness"). For a gadget whose parent is a XmManager subclass, the default value is copied from this parent (provided the value hasn't been explicitly set by the application); otherwise, the default is XmPIXELS. Possible values:

XmPIXELS		Xm100TH_POINTS
Xm100TH_MILLIMETERS		Xm100TH_FONT_UNITS
Xm1000TH_INCHES		
XmINCHES	(2.0)	
XmPOINTS	(2.0)	
XmFONT_UNITS	(2.0)	

XmNuserData
A pointer to data that the application can attach to the gadget. This resource is unused internally.

Callback Resources

Gadget defines the following callback resources:

Callback	Reason Constant
XmNhelpCallback	XmCR_HELP

XmNhelpCallback
List of callbacks that are called when help is requested.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int      reason;     /* the reason that the callback was called */
    XEvent   *event;     /* event structure that triggered callback */
} XmAnyCallbackStruct;
```

Inherited Resources

Gadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. Gadget resets the default value of XmNborderWidth from 1 to 0.

Name	Inherited From
XmNancestorSensitive	RectObj
XmNborderWidth	RectObj
XmNdestroyCallback	Object
XmNheight	RectObj
XmNsensitive	RectObj
XmNwidth	RectObj
XmNx	RectObj
XmNy	RectObj

Behavior

Since Gadgets cannot have translations associated with them, a Gadget's behavior is controlled by the XmManager widget that contains the Gadget. If a Gadget has the keyboard focus, the XmManager handles passing events to the Gadget.

See Also

`Object`(2), `RectObj`(2), `XmManager`(2), `XmScreen`(2).

Name

XmGrabShell widget class – a popup shell that grabs the keyboard and pointer when mapped

Synopsis

Public Header:
<Xm/GrabShell.h>

Class Name:
XmGrabShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → XmGrabShell

Class Pointer:
xmGrabShellWidgetClass

Instantiation:
widget = `XmCreateGrabShell` (parent, name,...)
or
widget = `XtCreatePopupShell`[1] (name, xmGrabShellWidgetClass,...)

Functions/Macros:
`XmCreateGrabShell`(), `XmIsGrabShell`()

Availability

Motif 2.0 and later.

Description

GrabShell is a shell widget which grabs the pointer and keyboard when it is mapped. The purpose of this is to provide a popup which immediately directs focus to its child. The ComboBox widget utilizes this feature in implementing its popup List.

Although GrabShell is an internal widget used by the ComboBox, it has a public interface, and is therefore available for use.

New Resources

GrabShell defines the following resources:

Name	Class	Type	Default	Access
XmNbottomShadowColor	XmCBottomShadowColor	Pixel	dynamic	CSG
XmNbottomShadowPixmap	XmCBottomShadowPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNgrabStyle	XmCGrabStyle	int	GrabModeAsync	CSG

1.More precise than XtCreateWidget as given in 2nd edition.

Name	Class	Type	Default	Access
XmNownerEvents	XmCOwnerEvents	Boolean	False	CSG
XmNshadowThickness	XmCShadowThickness	Dimension	2	CSG
XmNtopShadowColor	XmCTopShadowColor	Pixel	dynamic	CSG
XmNtopShadowPixmap	XmCTopShadowPixmap	Pixmap	dynamic	CSG

XmNbottomShadowColor
The color used in drawing the border shadow's bottom and right sides, but only if XmNbottomShadowPixmap is NULL.

XmNbottomShadowPixmap
The pixmap used in drawing the border shadow's bottom and right sides.

XmNgrabStyle

Controls the further processing of pointer events once the grab has been initiated. Possible values:

GrabModeSync GrabModeAsync

Refer to Xlib documentation on `XGrabKeyboard`() and `XGrabPointer`() for more information.

XmNownerEvents
Specifies whether pointer or keyboard events are reported normally within the application, or only to the GrabShell window. Refer to Xlib documentation on `XGrabKeyboard`() and `XGrabPointer`() for more information.

XmNshadowThickness
The thickness of the shadow border.

XmNtopShadowColor
The color used in drawing the border shadow's top and left sides, but only if XmNtopShadowPixmap is NULL.

XmNtopShadowPixmap
The pixmap used in drawing the border shadow's top and left sides.

Inherited Resources

GrabShell inherits the resources shown below. The resources are listed alphabetically, along with the superclass that defines them. GrabShell resets XmNallowShellResize to True, XmNoverrideRedirect to True, XmNtransient to True, XmNwaitForWm to False, and XmNsaveUnder to False.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNmaxAspectX	WMShell
XmNallowShellResize	Shell	XmNmaxHeight	WMShell

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	Core	XmNmaxWIdth	WMShell
XmNaudibleWarning	VendorShell	XmNminAspectX	WMShell
XmNbackground	Core	XmNminAspectY	WMShell
XmNbackgroundPixmap	Core	XmNminHeight	WMShell
XmNbaseHeight	WMShell	XmNminWidth	WMShell
XmNbaseWidth	WMShell	XmNmwmDecorations	VendorShell
XmNborderColor	Core	XmNmwmFunctions	VendorShell
XmNborderPixmap	Core	XmNmwmInputMode	VendorShell
XmNborderWidth	Core	XmNmwmMenu	VendorShell
XmNbuttonFontList	VendorShell	XmNnumChildren	Composite
XmNbuttonRenderTable	VendorShell	XmNoverrideRedirect	Shell
XmNchildren	Composite	XmNpopdownCalback	Shell
XmNcolormap	Core	XmNpopupCallback	Shell
XmNcreatePopupChildProc	Shell	XmNpreeditType	VendorShell
XmNdefaultFontList	VendorShell	XmNsaveUnder	Shell
XmNdeleteResponse	VendorShell	XmNscreen	Core
XmNdepth	Core	XmNsensitive	Core
XmNdestroyCallback	Core	XmNshellUnitType	VendorShell
XmNgeometry	Shell	XmNtextFontList	VendorShell
XmNheight	Core	XmNtextRenderTable	VendorShell
XmNheightInc	WMShell	XmNtitle	WMShell
XmNiconMask	WMShell	XmNtitleEncoding	WMShell
XmNiconPixmap	WMShell	XmNtoolTipEnable	VendorShell
XmNiconWindow	WMShell	XmNtoolTipPostDuration	VendorShell
XmNinitialResourcesPersistent	Core	XmNtoolTipString	VendorShell
XmNinitialState	WMShell	XmNtransient	WMShell
XmNinput	WMShell	XmNtranslations	Core
XmNinputMethod	VendorShell	XmNvisual	Shell
XmNinputPolicy	VendorShell	XmNwaitForWm	WMShell
XmNinsertPosition	Composite	XmNwidth	Core
XmNkeyboardFocusPolicy	VendorShell	XmNwidthInc	WMShell
XmNlabelFontList	VendorShell	XmNwindowGroup	WMShell
XmNlabelRenderTable	VendorShell	XmNwinGravity	WMShell
XmNlayoutDirection	VendorShell	XmNwmTimeout	WMShell

Resource	Inherited From	Resource	Inherited From
XmNmappedWhenManaged	Core	XmNx	Core
XmNmaxAspectY	WMShell	XmNy	Core

Translations

The translations for GrabShell include those of WMShell.

Event	Action
BSelect Press	GrabShellBtnDown()
BSelect Release	GrabShellBtnUp()

Action Routines

GrabShell defines the following action routines:

GrabShellBtnDown()

If the event occurs outside the coordinates of the GrabShell, the widget is popped up, otherwise the event is ignored.

GrabShellBtnUp()

If the event occurs within the time specified by the multi-click interval of the display, the action ignores the event. Otherwise, the GrabShell is popped down.

GrabShellPopdown()

Grabs placed upon the pointer and keyboard are released, the GrabShell is unmapped, and the focus is reverted to the previous owner.

See Also

`XmCreateObject`(1), `Composite`(2), `Core`(2), `Shell`(2), `VendorShell`(2), `WMShell`(2)

Name

XmHierarchy – The Hierarchy widget class

Synopsis

Public Header:
<Xm/Hierarchy.h>

Class Name:
XmHierarchy

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmHierarchy

Class Pointer:
xmHierarchyWidgetClass

Instantiation:
The Hierarchy widget is not instantiated by itself.

Functions/Macros:
`XmHierarchyOpenAllAncestors(), XmHierarchygetChildNodes()`

Availability

OpenMotif 2.2 and later.

Description

The Hierarchy widget should be used as the base class for any widget that displays a hierarchy of information. (This refers to the information displayed, not to the hierarchy of objects in a compound widget.) This base class is used for the XmTree and XmOutline widgets, providing them with very similar APIs.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNautoClose	XmCAutoClose	Boolean	True	CSG
XmNcloseFolderPixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PISMAP	CSG
XmNhorizontalMargin	XmCDimension	Dimension	2	CSG
XmNnodeStateBeginEndCallback	XmCCallback	XtCallbackList	NULL	CSG
XmNnodeStateCallback	XmCNodeState-Callback	XtCallbackList	NULL	CSG
XmNNodeStateChangedCallback	XmCNodeState-ChangedCallback	XtCallbackList	NULL	CSG
XmNNodeStateChangedCallback	XmCNodeState-ChangedCallback	XtCallbackList	NULL	CSG
XmNopenFolderPixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNrefigureMode	XmCBoolean	Boolean	True	CSG
XmNverticalMargin	XmCDimension	Dimension	2	CSG

XmNautoClose

Specifies whether the Hierarchy should automatically restore a parent node's children when the parent node is reopened. If XmNautoClose is False, and the Hierarchy is fully expanded when the root node is closed, the entire Hierarchy is displayed when the root node is reopened. If XmNautoClose is True, the root node's children are closed when the root node is reopened.

XmNcloseFolderPixmap

XmNopenFolderPixmap

Specifies the pixmaps displayed in all folder button widgets that are associated with nodes with a state of XmClosed and XmOpen for nodes that specify no XmNnodeCloseFolderPixmap or XmNnodeOpenFolderPixmap, respectively. If the value of XmNopenFolderPixmap is set to XmUNSPECIFIED_PIXMAP (either at creation or via XtSetValues()), a default open-folder bitmap or color pixmap is displayed. Behavior is analogous for XmNcloseFolderPixmap.

XmNhorizontalMargin

XmNverticalMargin

The definitions of these resources are left to the subclass of the Hierarchy widget that does the geometry layout. They are intended to be used as the number of pixels between the object and the edges of the window in which it is contained. They are included here for consistency.

XmNnodeStateBeginEndCallback
: This callback is invoked at the beginning and end of a group of node changes (as for the closing of an entire tree, for instance). The call_date is a Boolean type. True indicates that this is the beginning of a state change. False indicates that the changes have finished.

XmNnodeStateCallback
: Specifies the list of callback routines called when a folder button is clicked. See Callback Routine for more details.

XmNnodeStateChangedCallback
: This callback list in invoked when a node's state (XmOpen or XmClosed) has changed. The callbacks are passed an XmHierarchy NodeStateData* as the call_data containing the information on the node whose state has changed.

XmNrefigureMode
: Specifies whether the Hierarchy should adjust the sizes of the children after a geometry or resize request, or simply ignore the request. This resource is very useful in improving the performance of an application that is making a large number of geometry changes all at once.

Constraint Resources

XmHierarchy Constraint Resource Set

Name	Class	Type	Default	Access
XmNinsertBefore	XmCInsertBefore	Widget	NULL	CSG
XmNnodeCLoseFolderPixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNnodeOpenFolderPixmap	XmCPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNnodeState	XmCNodeState	XmHierarchyNodeState	XmOpen	CSG
XmNparentNode	XmCParentNode	Widget	NULL	CSG

XmNinsertBefore
: Places the current node immediately before another node in the Hierarchy that has the same XmNparentNode value. If this value is NULL, the current node is inserted at the end of the list. This resource allows the Hierarchy's children to be reordered.

XmNnodeCLoseFolderPixmap
: Specifies the pixmap to be used when the node is in state XmClosed. The default is the parent widget's value for XmNcloseFolderPixmap.

XmNnodeOpenFolderPixmap
: Specifies the pixmap to be used when the node is in state XmOpen. The default is the parent widget's value for XmNopenFolderPixmap.

XmNnodeState
: Specifies the state of the current node. Acceptable values are: XmOpen, XmClose, XmAlwaysOpen, and XmHidden. A type converter has been registered that can convert the following strings: "open", "closed", "alwaysOpen", and "hidden".

XmNparentNode
: Specifies the parent of the current node. The parent node must be a widget sibling of the current node. If XmNparentNode is set to NULL, the node is placed on the screen as root. No node may have multiple parents.

Callback Structure

When a folder is clicked, the routines registered on the XmNnodeStateCallback list are passed a pointer to the following structure as client data:

```
typedef struct _XmHierarchyNodeStateData{
    widget                  widget;
    XmHierarchyNodeState    state
} XmHierarchyNodeStateData;
```

The widget element indicates the child node of Hierarchy being open or closed.

The state element indicates the current XmNnodeState (after the click) of this node. Legal values are XmOpen, XmClosed, XmAlwaysOpen, and XmHidden.

Inherited Resources

XmHierarchy behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager

Resource	Inherited From	Resource	Inherited From
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersist-ent	Core		

See Also

XmCreateObject(1), Composite(2), Constraint(2), XmCon-tainer, Core(2), XmHierarchy, XmManager(2), XmOutline, XmBulletinBoard(2).

Name

XmIconBox – The IconBox widget class

Synopsis

Public Header:
<Xm/IconBox.h>

Class Name:
XmIconBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmIconBox

Class Pointer:
xmIconBoxWidgetClass

Instantiation:

```
widget = XmCreateIconBox(parent, name,...)
or
widget = XtCreateWidget(name, xmIconBoxWidget-
Class,...)
```

Functions/Macros:

```
XmCreateIconBox()
```

Availability

OpenMotif 2.2 and later. (Contributed Widget).

Description

The Icon Box widget lays out its children on a grid with each child forced to be the same size and with the location of each child specified as an X and Y location on the grid. The size of the Icon Box, its children, and the number of cells displayed are calculated as described below. The general idea is that all children are always be shown and should be given their desired size whenever possible. The user may add or delete cells by resizing this window using the window manager widget. The preferred size is calculated by using the maximum desired child height or width and making sure that these are no smaller than the minimum sizes. This size is multiplied by the number of cells along the axis and properly padded to come up with a preferred size. The number of cells is the maximum of the largest cellX or cellY value and the minimum number of horizontal or vertical cells.

If the Icon box is forced larger than its preferred size more cells are added at the bottom-right of the widget while the children all remain at their preferred sizes.

If the Icon box is forced smaller than its preferred size each cell is forced to be smaller in order to allow all children to fit within the Icon Box. All children will be forced to the same smaller size.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNhorizontalMargin	XmCMargin	Dimension	4	CSG
XmNminimumHorizontalCells	XmCDefaultCells	int	8	CSG
XmNminimumVerticalCells	XmCDefaultCells	int	4	CSG
XmNminimumCellHeight	XmCMinimumCellSize	Dimension	10	CSG
XmNminimumCellWidth	XmCMinimumCellSize	Dimension	20	CSG
XmNverticalMargin	XmCMargin	Dimension	4	CSG

XmNhorizontalMargin
XmNverticalMargin
: The amount of space between each cell and its neighbor or the edge of the Icon Box.

XmNminimumHorizontalCells
XmNminimumVerticalCells
: The minimum number of cells to display in the horizontal and vertical directions. This number of cells will always be displayed.

XmNminimumCellWidth
XmNminimumCellHeight
: The smallest size the cells are allowed to be in the direction specified.

Inherited Resources

Icon Box inherits behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass.

Resource	Inherited from	Resource	Inherited from
XmNbottomShadowColor	Primitive	XmNancestorSensitive	Core
XmNbottomShadowPixmap	Primitive	XmNbackground	Core
XmNconvertCallback	Primitive	XmNbackgroundPixmap	Core
XmNforeground	Primitive	XmNborderColor	Core
XmNhelpCallback	Primitive	XmNborderPixmap	Core
XmNhighlightColor	Primitive	XmNborderWidth	Core
XmNhighlightOnEnter	Primitive	XmNcolormap	Core
XmNhighlightPixmap	Primitive	XmNdepth	Core
XmNhighlightThickness	Primitive	XmNdestroyCallback	Core
XmNlayoutDirection	Primitive	XmNheight	Core
XmNnavigationType	Primitive	XmNinitialResourcesPersistent	Core
XmNpopupHandlerCallback	Primitive	XmNmappedWhenManaged	Core
XmNshadowThickness	Primitive	XmNscreen	Core
XmNtopShadowColor	Primitive	XmNsensitive	Core
XmNtopShadowPixmap	Primitive	XmNtranslations	Core
XmNtraversalOn	Primitive	XmNwidth	Core
XmNunitType	Primitive	XmNx	Core
XmNuserData	Primitive	XmNy	Core
XmNaccelerators	Core		

Translations

XmIconBox inherits translations from XmManager.

See Also

`XmButtonBox, XmColumn, XmContainer, Core(2), XmCreate-Object(1), XmPrimitive(2), XmRowColumn.`

Name

XmIconButton – The IconButton widget class

Synopsis

Public Header:
<Xm/IconButton.h>

Class Name:
XmIconButton

Class Hierarchy:
Core → XmPrimitive → XmIconButton

Class Pointer:
xmIconButtonWidgetClass

Instantiation:

```
widget = XmCreateIconButton(parent, name,...)
or
widget = XtCreateWidget(name, xmIconButtonWidget-
Class,...)
```

Functions/Macros:

```
XmCreateIconButton()
```

Availability

OpenMotif 2.2 and later. (Contributed Widget).

Description

The Icon Button widget is a selectable area of the screen that contains both a label and a string. When the user selects this button with the OSF/Motif select button its activateCallback is called. This widget can also be used as a Toggle button, although it will have no indicator. The placement of the icon relative to the text can be modified by using the iconPlacement resource.
Note: This widget takes a String as its label, not an XmString, so there is no need to use XmStringCreate to get a properly formatted string. Internationalized text is not currently supported.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using

XtSetValues (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNactivateCallback	XmCCallback	Callback	NULL	CSG
XmNalignment	XmCAlignment	Alignment	XmALIGNMENT_BEGINNING	CSG
XmNarmColor	XmCArmColor	Pixel	<dynamic>	CSG
XmNdoubleClickCallback	XmCCallback	Callback	NULL	CSG
XmNfontList	XmCFontList	FontList	fixed	CSG
XmNhorizontalMargin	XmCSpace	HorizontalDimension	2	CSG
XmNiconTextPadding	XmCSpace	VerticalDimension	2	CSG
XmNiconPlacement	XmCIconPlacement	IconPlacement	XmIconTop	CSG
XmNlabel	XmCLabel	String	Widget name	CSG
XmNlabelString	XmCLabelString	XmString	Widget name	CSG
XmNpixmap	XmCPixmap	Pixmap	None	CSG
XmNrecomputeSize	XmCBoolean	Boolean	True	CSG
XmNset	XmCBoolean	Boolean	False	CSG
XmNstringDirection	XmCStringDirection	StringDirection	XmSTRING_DIRECTION_L_TO_R	CSG
XmNverticalMargin	XmCSpace	VerticalDimension	2	CSG

XmNactivateCallback
: This list of callback routines is called whenever the icon button is clicked on by the user. The format of the callback routines is specified below.

XmNarmColor
: This is the pixel index that describes the color to fill the widget with when it is set.

XmNdoubleClickCallback
: This list of callback routines is called whenever the user double clicks on this widget. The format of the callback routines is specified below.

XmNfontList
: The default font in this list is used to render the label string of the icon button. horizontalMargin

XmNverticalMargin
: The amount of space to be left between the edge of the shadow and the text or pixmap displayed. The vertical and horizontal spacing can be controlled independently.

XmNiconTextPadding
: The amount of space to be left between the pixmap and the label string.

XmNiconPlacement
: The location of the pixmap (icon) with respect to the displayed text. This resource can take one of the following values: XmIconTop, XmIconBottom, XmIconLeft, XmIconRight, XmIconNone, and XmIconOnly. A type converter has been registered that converts the following strings: "top", "bottom", "left", "right", "none", and "iconOnly".
: If only a string is displayed in the Icon Button, this resource can be used to change the justification of the label. XmIconTop=bottom, XmIconBottom=top, XmIconRight=Left, XmIconLeft=Right, XmIcon- None=Center. In order to use these options, pixmap must be set to None.

XmNlabel
: The string to display in this button. This string can only have one font, but can be any number of lines long. Use the NEW_LINE character (’0) to separate lines. This resource has been superseded by labelString, but is included for backwards compatibility. If XmNlabelString is set , XmNlabel is ignored

XmNlabelString
: Specifies the compound string to be displayed in the button. If this value is NULL, the value of XmNlabel is used. If both are NULL, labelString is initialized by converting the name of the widget to a compound string. Refer to XmString(3X) in the OSD/Motif Programmers’ Reference for more information on the creation and structure of compound strings.

XmNpixmap
: The pixmap to display. This pixmap may either be of depth one (1), or the same depth as the screen this widget is being displayed on. If the pixmap is of depth one then XCopyPlane is used to render the pixmap in the foreground and background colors. If the pixmap is not of depth one then XCopyArea is used and all the original colors of the pixmap are preserved. Unlike the Motif PushButton widget the pixmap is automatically stippled when the Icon Button becomes insensitive.

XmNrecomputeSize
: If this Boolean value is True then the icon button will ask its parent to resize it to be just large enough to contain the pixmap, label and shadows. If it is False then the icon button will not attempt a resize.

XmNset
: This Boolean value represents the current state of the icon button. If this value is True then the icon button is set and is rendered as depressed. Otherwise it is unset and is rendered normally.

XmNstringDirection
: Specifies the direction in which the string is to be drawn. XmNSTRING_DIRECTION_L_TO_R is drawn left to right, while

XmNSTRING_DIRECTION_R_TO_L is drawn right to left. The default for this resource is determined at creation time. If no value is specified for this resource and the widget's parent is a manager, the value is inherited from the parent; otherwise, it defaults to XmNSTRING_DIRECTION_ L_TO_R.

Inherited Resources

Resource	Inherited from	Resource	Inherited from
XmNbottomShadowColor	Primitive	XmNaccelerators	Core
XmNbottomShadowPixmap	Primitive	XmNancestorSensitive	Core
XmNconvertCallback	Primitive	XmNbackground	Core
XmNforeground	Primitive	XmNbackgroundPixmap	Core
XmNhelpCallback	Primitive	XmNborderColor	Core
XmNhighlightColor	Primitive	XmNborderPixmap	Core
XmNhighlightOnEnter	Primitive	XmNborderWidth	Core
XmNhighlightPixmap	Primitive	XmNcolormap	Core
XmNhighlightThickness	Primitive	XmNdepth	Core
XmNlayoutDirection	Primitive	XmNdestroyCallback	Core
XmNnavigationType	Primitive	XmNheight	Core
XmNpopupHandlerCallback	Primitive	XmNinitialResourcesPersistent	Core
XmNshadowThickness	Primitive	XmNmappedWhenManaged	Core
XmNtoolTipString	Primitive	XmNscreen	Core
XmNtopShadowColor	Primitive	XmNsensitive	Core
XmNtopShadowPixmap	Primitive	XmNtranslations	Core
XmNtraversalOn	Primitive	XmNwidth	Core
XmNunitType	Primitive	XmNx	Core
XmNuserData	Primitive	XmNy	Core

Translations

XmIconBox inherits translations from XmManager.

See Also

`XmCreateObject(1), Core(2), XmPrimitive(2), XmPushButton, XmPushButtonGadget.`

Name

XmIconGadget widget class –a gadget for displaying both text and a pixmap

Synopsis

Public Header:
<Xm/IconG.h>

Class Name:
XmIconGadget

Class Hierarchy:
Object → RectObj → XmGadget → XmIconGadget

Class Pointer:
xmIconGadgetClass

Instantiation:
widget = `XmCreateIconGadget` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmIconGadgetClass,...)

Functions/Macros:
`XmCreateIconGadget(), XmIsIconGadget`()

Availability

Motif 2.0 and later.

Description

IconGadget is a gadget which can display both textual and pixmap information simultaneously. The textual data can be either centered below the pixmap, or placed to the side, depending upon the value of the XmNviewType resource. The value XmLARGE_ICON centers below, and XmSMALL_ICON horizontally aligns, the pixmap and textual information.

IconGadget is intended for use with the Container, which lays out IconGadget children in various styles, in order to represent application objects of some kind. In addition to the textual labelling, an IconGadget can be associated with an array of detail information, which presumably represents attributes of the application object, and which the Container parent can lay out relative to the IconGadget.

By default, the IconGadget will use its widget name for the XmNlabelString resource. To display only an image, specify the appropriate pixmap resource, and set the XmNlabelString resource to XmUNSPECIFIED. Alternatively, apply an XmNlabelString resource which specifies a compound string with no text component.

Traits

IconGadget holds the XmQTcontainerItem, XmQTcareParentVisual, XmQTpointIn, and XmQTaccessColors traits, and uses the XmQTcontainer and XmQTspecifyRenderTable traits. The XmQTpointIn trait is undocumented.

New Resources

IconGadget defines the following resources:

Name	Class	Type	Default	Access
XmNalignment	XmCAlignment	unsigned char	XmALIGNMENT_CENTER	CSG
XmNdetail	XmCDetail	XmStringTable	NULL	CSG
XmNdetailCount	XmCDetailCount	Cardinal	0	CSG
XmNfontList	XmCFontList	XmFontList	NULL	CSG
XmNlabelString	XmCLabelString	XmString	dynamic	CSG
XmNlargeIconMask	XmCIconMask	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNlargeIconPixmap	XmCIconPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNmarginHeight	XmCMarginHeight	Dimension	2	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	2	CSG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNsmallIconMask	XmCIconMask	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNsmallIconPixmap	XmCIconPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNspacing	XmCSpacing	Dimension	4	CSG
XmNviewType	XmCViewType	unsigned char	XmLARGE_ICON	CSG
XmNvisualEmphasis	XmCVisualEmphasis	unsigned char	XmNOT_SELECTED	CSG

XmNalignment
: In Motif 2.1, specifies the horizontal alignment of the textual and pixmap data. Possible values:

  XmALIGNMENT_BEGINNING
  XmALIGNMENT_CENTER
  XmALIGNMENT_END

XmNdetail
: Specifies an array of compound strings, representing the detail information associated with the IconGadget.

XmNdetailCount
: Specifies the number of compound strings in XmNdetail.

XmNfontList
: Specifies the font list associated with the IconGadget. In Motif 2.0 and later, the XmFontList is an obsolete data type, and has been replaced by the XmRenderTable. The resource is maintained for backwards compatibility but implemented as a render table. Any specified XmNrenderTable resource takes priority.

XmNlabelString
: The compound string representing the textual data for labelling the IconGadget. If unspecified, the value is constructed out of the name of the gadget.

XmNlargeIconMask
: Specifies the icon mask used when XmNviewType is XmLARGE_ICON. This resource must be a bitmap (a pixmap of depth 1).

XmNlargeIconPixmap
: Specifies the pixmap used when XmNviewType is XmLARGE_ICON.

XmNmarginHeight
: In Motif 2.1, specifies the vertical distance in pixels between the highlight rectangle and the IconGadget contents.

XmNmarginWidth
: In Motif 2.1, specifies the horizontal distance in pixels between the highlight rectangle and the IconGadget contents.

XmNrenderTable
: Specifies the XmRenderTable used for displaying textual data associated with the IconGadget. If unspecified, the value is taken from the nearest ancestor which holds the XmQTspecifyRenderTable trait, using the XmLABEL_RENDER_TABLE value of any ancestor so found.

XmNsmallIconMask
: Specifies the icon mask used when XmNviewType is XmSMALL_ICON. This resource must be a bitmap (a pixmap of depth 1).

XmNsmallIconPixmap
: Specifies the pixmap used when XmNviewType is XmSMALL_ICON.

XmNspacing
: In Motif 2.1, specifies the distance in pixels between the textual and pixmap components of the IconGadget.

XmNviewType
: Specifies the IconGadget layout style. If the parent of the IconGadget is a Container, the view type is overridden by the value of the XmNentryViewType resource of the parent if the value is not XmANY_ICON. If XmNviewType is XmLARGE_ICON, the pixmap specified by XmNlargeIconPixmap is displayed above the textual label, with the text centered upon the pixmap. If XmNviewType

is XmSMALL_ICON, the label is displayed either to the left or the right of the pixmap, depending upon the XmNlayoutDirection resource.

XmNvisualEmphasis
Specifies whether the IconGadget is displayed in normal or selected state.

If the value is XmSELECTED, the gadget is rendered using the XmNselectColor resource of the Container parent. XmNOT_SELECTED displays in normal state.

Inherited Resources

IconGadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them.

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	RectObj	XmNlayoutDirection	XmGadget
XmNbackground	XmGadget	XmNnavigationType	XmGadget
XmNbackgroundPixmap	XmGadget	XmNsensitive	RectObj
XmNbottomShadowColor	XmGadget	XmNshadowThickness	XmGadget
XmNbottomShadowPixmap	XmGadget	XmNtoolTipString	XmGadget
XmNborderWidth	RectObj	XmNtopShadowColor	XmGadget
XmNdestroyCallback	Object	XmNtopShadowPixmap	XmGadget
XmNforeground	XmGadget	XmNtraversalOn	XmGadget
XmNheight	RectObj	XmNunitType	XmGadget
XmNhelpCallback	XmGadget	XmNuserData	XmGadget
XmNhighlightColor	XmGadget	XmNwidth	RectObj
XmNhighlightOnEnter	XmGadget	XmNx	RectObj
XmNhighlightPixmap	XmGadget	XmNy	RectObj
XmNhighlightThickness	XmGadget		

See Also

`XtCreateObject(1), Object`(2), `RectObj`(2), `XmGadget`(2).

Name

XmInformationDialog –an unmanaged MessageBox as a child of a DialogShell.

Synopsis

Public Header:
<Xm/MessageB.h>

Instantiation:
widget = `XmCreateInformationDialog`(...)

Functions/Macros:
`XmCreateInformationDialog`(), `XmMessageBoxGetChild`()

Description

An XmInformationDialog is a compound object created by a call to XmCreateInformationDialog() that an application can use to provide the user with information. An InformationDialog consists of a DialogShell with an unmanaged MessageBox widget as its child. The MessageBox resource XmNdialogType is set to XmDIALOG_INFORMATION. An InformationDialog includes four components: a symbol, a message, three buttons, and a separator between the message and the buttons. By default, the symbol is a lowercase *i*. In Motif 1.2, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Cancel**, and **Help** by default.

Default Resource Values

An InformationDialog sets the following default values for MessageBox resources:

Name	Default
XmNdialogType	XmDIALOG_INFORMATION
XmNsymbolPixmap	xm_information[a]

a. Erroneously given as Xm_information in 2nd edition.

Widget Hierarchy

When an InformationDialog is created with a specified name, the DialogShell is named name*_popup* and the MessageBox is called *name***.**

See Also

`XmCreateObject`(1), `XmMessageBoxGetChild`(1), `XmDialogShell`(2), `XmMessageBox`(2).

Name

XmLabel widget class –a simple widget that displays a non-editable label.

Synopsis

Public Header:
<Xm/Label.h>

Class Name:
XmLabel

Class Hierarchy:
Core → XmPrimitive → XmLabel

Class Pointer:
xmLabelWidgetClass

Instantiation:
widget = `XmCreateLabel` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmLabelWidgetClass,...)

Functions/Macros:
`XmCreateLabel`(), `XmIsLabel`()

Description

Label provides a text string or a pixmap for labelling other widgets in an application. Label is also a superclass for the various button widgets. Label does not accept any button or key events, but it does receive enter and leave events.

Traits

Label holds the XmQTmenuSavvy, XmQTtransfer, and XmQTaccessTextual traits, which are inherited by any derived classes, and uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

New Resources

Label defines the following resources, where the access for every resource is CSG:

Name	Class	Type	Default	Access
XmNaccelerator	XmCAccelerator	String	NULL	CSG
XmNacceleratorText	XmCAcceleratorText	XmString	NULL	CSG
XmNalignment	XmCAlignment	unsigned char	dynamic	CSG
XmNfontList	XmCFontList	XmFontList	dynamic	CSG

Name	Class	Type	Default	Access
XmNlabelInsensitivePixmap	XmCLabelInsensitivePixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNlabelPixmap	XmCLabelPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNlabelString	XmCLabelString	XmString	dynamic	CSG
XmNlabelType	XmCLabelType	unsigned char	XmSTRING	CSG
XmNmarginBottom	XmCMarginBottom	Dimension	0	CSG
XmNmarginHeight	XmCMarginHeight	Dimension	2	CSG
XmNmarginLeft	XmCMarginLeft	Dimension	0	CSG
XmNmarginRight	XmCMarginRight	Dimension	0	CSG
XmNmarginTop	XmCMarginTop	Dimension	0	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	2	CSG
XmNmnemonic	XmCMnemonic	KeySym	NULL	CSG
XmNmnemonicCharSet	XmCMnemonicCharSet	String	XmFONTLIST_DEFAULT_TAG	CSG
XmNrecomputeSize	XmCRecomputeSize	Boolean	True	CSG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNstringDirection	XmCStringDirection	XmStringDirection	dynamic	CSG

XmNaccelerator
: A string that describes a button widget's accelerator (the modifiers and key to use as a shortcut in selecting the button). The string's format is like that of a translation but allows only a single key press event to be specified.

XmNacceleratorText
: The text that is displayed for an accelerator.

XmNalignment
: The alignment (left to right) for a label's text or pixmap. Possible values are XmALIGNMENT_BEGINNING, XmALIGNMENT_CENTER, and XmALIGNMENT_END. In Motif 2.0 and later, the interpretation of alignment depends upon the value of any inherited XmNlayoutDirection resource.

XmNfontList
: The font list used for the widget's text. From Motif 2.0 and later, the XmfontList is an obsolete data type, and the Rendition Table is the preferred method of setting appearance. Although maintained for backwards compatibility, the resource is implemented through a render table. Any XmNrenderTable resource takes precedence.

XmNlabelInsensitivePixmap
The pixmap label for an insensitive button (when XmNlabelType is XmPIXMAP or XmPIXMAP_AND_STRING).

XmNlabelPixmap
The pixmap used when XmNlabelType is XmPIXMAP or XmPIXMAP_AND_STRING.

XmNlabelString
The compound string used when XmNlabelType is XmSTRING or XmPIXMAP_AND_STRING. If this resource is NULL, the application uses the widget's name (converted to compound string format).

XmNlabelType
The type of label (either string or pixmap). Possible values:
XmPIXMAP/* use XmNlabelPixmap or XmNlabelInsensitivePixmap */
XmPIXMAP_AND_STRING */
XmSTRING/* use XmNlabelString */

XmNmarginTop, XmNmarginBottom,

XmNmarginLeft, XmNmarginRight
The amount of space between one side of the label text and the nearest margin.

XmNmarginHeight, XmNmarginWidth
The spacing between one side of the label and the nearest edge of a shadow.

XmNmnemonic
A keysym that gives the user another way to select a button. In the label string, the first character matching this keysym will be underlined.

XmNmnemonicCharSet
The character set for the label's mnemonic.

XmNrecomputeSize
If True (default), the Label widget changes its size so that the string or pixmap fits exactly.

XmNrenderTable
In Motif 2.0 and later, specifies the render table for the Label. If NULL, this is inherited from the nearest ancestor that has the XmQTspecifyRenderTable trait, taking the XmLABEL_RENDER_TABLE value from the ancestor. The BulletinBoard, VendorShell, and MenuShell widgets and derived classes set this trait.

XmNstringDirection

In Motif 2.0 and later, XmNstringDirection is superseded by the inherited XmNlayoutDirection resource. The direction in which to draw the string. Possible values are:

XmSTRING_DIRECTION_L_TO_R
XmSTRING_DIRECTION_R_TO_L
XmDEFAULT_DIRECTION

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;     /* set to XmCR_HELP                    */
    XEvent      *event;     /* points to event that triggered callback */
} XmAnyCallbackStruct;
```

Inherited Resources

Label inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. Label sets the default values of XmNhighlightThickness and XmNshadowThickness to 0 and XmNtraversalOn to False. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNhighlightThickness	XmPrimitive
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNlayoutDirection	XmPrimitive
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnavigationType	XmPrimitive
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmPrimitive
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmPrimitive	XmNsensitive	Core
XmNbottomShadowPixmap	XmPrimitive	XmNshadowThickness	XmPrimitive
XmNcolormap	Core	XmNtoolTipString	XmPrimitive
XmNconvertCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNdepth	Core	XmNtopShadowPixmap	XmPrimitive
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNheight	Core	XmNunitType	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNuserData	XmPrimitive

Resource	Inherited From	Resource	Inherited From
XmNhighlightColor	XmPrimitive	XmNwidth	Core
XmNhighlightOnEnter	XmPrimitive	XmNx	Core
XmNhighlightPixmap	XmPrimitive	XmNy	Core

Translations

Event	Action
BTransfer Press	ProcessDrag()
KHelp	Help()

For subclasses of Label:

Event	Action
KLeft	MenuTraverseLeft()
KRight	MenuTraverseRight()
KUp	MenuTraverseUp()
KDown	MenuTraverseDown()
MAny KCancel	MenuEscape()

Action Routines

Label defines the following action routines:

Help()

Unposts menus, restores keyboard focus, and invokes the callbacks from XmNhelpCallback, if there are any.

MenuEscape()

Unposts the menu, disarms the associated CascadeButton, and restores keyboard focus.

MenuTraverseDown()

In a MenuBar, if the current menu item has a submenu, posts the submenu, disarms the current menu item, and arms the first item in the submenu. In a menu pane, disarms the current menu item and arms the item below it, wrapping around to the top if necessary.

MenuTraverseLeft()

In a MenuBar, disarms the current menu item and arms the next item to the left, wrapping if necessary. In a menu pane, disarms the current item and arms the item to the left if there is such an item. Otherwise, unposts the current submenu and, if that submenu is attached to a MenuBar item, traverses to the MenuBar item to the left (wrapping if necessary), posts the submenu, and arms the first item in the submenu. In a PopupMenu or a torn-off menu pane, traverses to the menu item to the left, wrapping to the right if necessary.

MenuTraverseRight()

In a MenuBar, disarms the current menu item and arms the next item to the right, wrapping if necessary. In a menu pane, if the current item is a CascadeButton, posts the associated submenu. Otherwise, disarms the current item and arms the item to the right if there is such an item or unposts all submenus, traverses to the MenuBar item to the right (wrapping if necessary), posts the submenu, and arms the first item in the submenu. In a PopupMenu or a torn-off menu pane, traverses to the menu item to the right, wrapping to the left if necessary.

MenuTraverseUp()

In a menu pane, disarms the current menu item and arms the item above it, wrapping around to the bottom if necessary.

ProcessDrag()

In Motif 1.2, initiates a drag and drop operation using the contents of the Label. In Motif 2.0 and later, this indirectly invokes any procedures specified by the inherited XmNconvertCallback resource.

See Also

`XmCreateObject`(1), `Core`(2), `XmPrimitive`(2).

Name

XmLabelGadget widget class – a simple gadget that displays a non-editable label.

Synopsis

Public Header:
<Xm/LabelG.h>

Class Name:
XmLabelGadget

Class Hierarchy:
Object → RectObj → XmGadget → XmLabelGadget

Class Pointer:
xmLabelGadgetClass

Instantiation:
widget = `XmCreateLabelGadget` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmLabelGadgetClass,...)

Functions/Macros:
`XmCreateLabelGadget`(), `XmOptionLabelGadget`(), `XmIsLabelGadget`()

Description

LabelGadget is the gadget variant of Label.

In Motif 2.0, the LabelGadget cached resource set is expanded to include XmNforeground, XmNbackground, XmNtopShadowColor, XmNtopShadowPixmap, XmNbottomShadowColor, XmNbottomShadowPixmap, XmNhighlightColor and XmNhighlightPixmap resources. LabelGadgets sharing the same Manager parent can therefore have independent appearance.

Traits

LabelGadget holds the XmQTmenuSavvy, XmQTtransfer, XmQTaccessTextual, XmQTcareParentVisual, and XmQTaccessColors traits, which are inherited by any derived classes, and uses the traits XmQTmenuSystem and XmQTspecifyRenderTable.

New Resources

LabelGadget's new resources, callback resources, and callback structure are the same as those for Label.

Inherited Resources

LabelGadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. LabelGadget sets the default values of XmNhighlightThickness and XmNshadowThickness to 0 (zero) and XmNtraversalOn to False. The default value of XmNborderWidth is reset to 0 by Gadget.

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	RectObj	XmNlayoutDirection	XmGadget
XmNbackground	XmGadget	XmNnavigationType	XmGadget
XmNbackgroundPixmap	XmGadget	XmNsensitive	RectObj
XmNbottomShadowColor	XmGadget	XmNshadowThickness	XmGadget
XmNbottomShadowPixmap	XmGadget	XmNtoolTipString	XmGadget
XmNborderWidth	RectObj	XmNtopShadowColor	XmGadget
XmNdestroyCallback	Object	XmNtopShadowPixmap	XmGadget
XmNforeground	XmGadget	XmNtraversalOn	XmGadget
XmNheight	RectObj	XmNunitType	XmGadget
XmNhelpCallback	XmGadget	XmNuserData	XmGadget
XmNhighlightColor	XmGadget	XmNwidth	RectObj
XmNhighlightOnEnter	XmGadget	XmNx	RectObj
XmNhighlightPixmap	XmGadget	XmNy	RectObj
XmNhighlightThickness	XmGadget		

Behavior

As a Gadget subclass, LabelGadget has no translations associated with it. However, LabelGadget behavior corresponds to the action routines of the Label widget. See the Label action routines for more information.

Behavior	Equivalent Label Action
BTransfer Press	ProcessDrag()
KHelp	Help()

For subclasses of LabelGadget:

Behavior	Equivalent Label Action
KLeft	MenuTraverseLeft()
KRight	MenuTraverseRight()
KUp	MenuTraverseUp()
KDown	MenuTraverseDown()
MAny KCancel	MenuEscape()

See Also

`XmCreateObject`(1), `XmOptionLabelGadget`(1), `Object`(2), `RectObj`(2), `XmGadget`(2), `XmLabel`(2).

Name

XmList widget class –a widget that allows a user to select from a list of choices.

Synopsis

Public Header:
<Xm/List.h>

Class Name:
XmList

Class Hierarchy:
Core → XmPrimitive → XmList

Class Pointer:
xmListWidgetClass

Instantiation:
widget = `XmCreateList` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmListWidgetClass,...)

Functions/Macros:
`XmCreateList`(), `XmCreateScrolledList`(), `XmList`... routines, `XmIsList`()

Description

List provides a list of choices from which a user can select one or more items, based on the selection policy. List supports four selection policies: single select, browse select, multiple select, and extended select.

In single select mode, only one item can be selected at a time; a button press on an item selects it and deselects the previously selected item. In browse select mode, only one item can be selected at a time; a button press works as in single select mode and, additionally, a button drag moves the selection with the pointer. In multiple select mode, any number of items can be selected at a time; a button press toggles the selection state of an item and does not change the selection state of any other items. In extended select mode, any number of items can be selected at a time; discontinuous ranges of items can be selected by combining button presses and button drags.

Selections can be made by using either the pointer or the keyboard. Keyboard selection has two modes: normal mode and add mode. In normal mode, keyboard navigation operations affect the selection; the item with the keyboard focus is

always selected. In add mode, keyboard navigation operations are distinct from selection operations; the item with the keyboard focus can be disjoint from the selection. Browse select operates in normal mode; single select and multiple select operate in add mode; extended select can be made to operate in either mode. Normal mode uses a solid location cursor while add mode uses a dashed location cursor.

In Motif 1.2 and later, List is a supported drag source for drag and drop operations. BTransfer Press starts a drag and drop operation using the selected items in the List. If BTransfer is pressed over an unselected item, that item is dragged instead of the selected items.

Traits

List holds the XmQTtransfer trait, which is inherited by any derived classes, and uses the XmQTnavigator, XmQTscrollFrame and XmQTspecifyRenderTable traits.

New Resources

List defines the following resources:

Name	Class	Type	Default	Access
XmNautomaticSelection	XmCAutomaticSelection	XtEnum	XmNO_AUTO_SELECT	CSG
XmNdoubleClickInterval	XmCDoubleClickInterval	int	dynamic	CSG
XmNfontList	XmCFontList	XmFontList	dynamic	CSG
XmNhorizontalScrollBar	XmCHorizontalScrollBar	Widget	NULL	G
XmNitemCount	XmCItemCount	int	0	CSG
XmNitems	XmCItems	XmStringTable	NULL	CSG
XmNlistMarginHeight	XmCListMarginHeight	Dimension	0	CSG
XmNlistMarginWidth	XmCListMarginWidth	Dimension	0	CSG
XmNlistSizePolicy	XmCListSizePolicy	unsigned char	XmVARIABLE	CSG
XmNlistSpacing	XmCListSpacing	Dimension	0	CSG
XmNmatchBehavior	XmCMatchBehavior	unsigned char	XmQUICK_NAVIGATE	CSG
XmNprimaryOwnership	XmCPrimaryOwnership	unsigned char	XmOWN_NEVER	CSG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNscrollBarDisplayPolicy	XmCScrollBarDisplayPolicy	unsigned char	XmAS_NEEDED	CSG
XmNselectColor	XmCSelectColor	Pixel	dynamic	CSG
XmNselectedItemCount	XmCSelectedItemCount	int	0	CSG
XmNselectedItems	XmCSelectedItems	XmStringTable	NULL	CSG

Name	Class	Type	Default	Access
XmNselectedPositionCount	XmCSelectedPositionCount	int	0	CSG
XmNselectedPositions	XmCSelectedPositions	int *	NULL	CSG
XmNselectionMode	XmCSelectionMode	unsigned char	XmNORMAL_MODE	CSG
XmNselectionPolicy	XmCSelectionPolicy	unsigned char	XmBROWSE_SELECT	CSG
XmNstringDirection	XmCStringDirection	XmStringDirection	dynamic	CSG
XmNtopItemPosition	XmCTopItemPosition	int	1	CSG
XmNverticalScrollBar	XmCVerticalScrollBar	Widget	NULL	G
XmNvisibleItemCount	XmCVisibleItemCount	int	dynamic	CSG

XmNautomaticSelection

If True (and the widget's XmNselectionPolicy is either XmBROWSE_SELECT or XmEXTENDED_SELECT), then this resource calls XmNsingleSelection-Callback whenever the user moves into a new item.

If False, then the user must release the mouse button before any selection callbacks are called.

From Motif 2.0, the resource has changed type from a Boolean to an enumeration. Possible values of the enumerated type:

```
XmAUTO_SELECT       /* enables automatic selection */
XmNO_AUTO_SELECT    /* disables automatic selection */
```

XmNdoubleClickInterval

The time span (in milliseconds) within which two button clicks must occur to be considered a double click rather than two single clicks. By default, this value is the multiclick time of the display.

XmNfontList

The font list used for the widget's items. From Motif 2.0 and later, the XmfontList is considered obsolete, and the Rendition Table is the preferred method of setting appearance. Any XmNrenderTable value will take precedence.

XmNhorizontalScrollBar

When the List is part of a ScrolledList, specifies the widget ID of the ScrollBar created by the List to perform horizontal scrolling.

XmNitemCount

The total number of items. The widget updates this resource every time a list item is added or removed.

XmNitems
A pointer to an array of compound strings. The compound strings are the list items to display. A call to XtGetValues() returns the actual list items (not a copy), so don't have your application free these items.

XmNlistMarginHeight

XmNlistMarginWidth
The height or width of the margin between the border of the List and the items in the list.

XmNlistSizePolicy
The method for resizing the widget when a list item exceeds the width of the work area. This resizing policy must be set at creation time. Possible values:

```
XmVARIABLE          /* grow to fit; don't add ScrollBar          */
XmCONSTANT          /* don't grow to fit; add ScrollBar          */
XmRESIZE_IF_POSSIBLE /* grow or shrink; add ScrollBar if too large */
```

XmNlistSpacing
The spacing between items.

XmNmatchBehavior
In Motif 2.0 and later, specifies whether the widget navigates to items within the List by matching keyboard input against the first character of each item. If the value is XmNONE, no matching is performed. If the value is XmQUICK_NAVIGATE, any characters typed when the List has the focus are compared against the first character of each item. If a match is found, the matched list item is automatically made the current item. Subsequently typing the same character progresses cyclically through the list to find any further matching item.

XmNprimaryOwnership
Specifies how the list interacts with the primary selection when a user selects an item from the list. Possible values:

```
XmOWN_NEVER          /* never take ownership of the primary selection */
XmOWN_ALWAYS         /* always take ownership of the selection        */
XmOWN_MULTIPLE       /* take ownership if more than one item selected */
XmOWN_POSSIBLE_MULTIPLE /* take ownership if selection policy         */
                     /* is multiple or extended                       */
```

XmNrenderTable
In Motif 2.0 and later, specifies the render table for the list. If unspecified, the value of the resource is inherited from the nearest ancestor which holds the XmQTspecifyRenderTable trait, using the XmTEXT_RENDER_TABLE value of the ancestor so found. This resource, together with the XmNvisibleItem-Count resource, is used to calculate the List widget's height.

XmNscrollBarDisplayPolicy
Determines when to display vertical scrollbars in a ScrolledList widget. Possible values:

```
XmSTATIC        /* vertical ScrollBar always displays   */
XmAS_NEEDED     /* add ScrollBar when list is too large */
```

XmNselectColor
In Motif 2.0 and later, specifies the color used to draw the background of selected items. In addition to allocated Pixel values, the constant XmDEFAULT_SELECT_COLOR specifies a color between the XmNbackground and XmNbottomShadowColor, XmHIGHLIGHT_COLOR makes the select color the same as the XmNhighlightColor value, and XmREVERSED_GROUND_COLORS makes the XmNselectColor the same as the XmNforeground, using the XmNbackground color to render any text.

XmNselectedItemCount
The number of items in the list of selected items.

XmNselectedItems
A pointer to an array of compound strings. The compound strings represent the currently selected list items. A call to `XtGetValues`() returns the actual list items (not a copy), so don't have your application free these items.

XmNselectedPositionCount
In Motif 2.0 and later, specifies the number of positions in the list of selected positions.

XmNselectedPositions
In Motif 2.0 and later, specifies a pointer to an array of integers, representing the currently selected list positions. A call to `XtGetValues`() returns the actual list position array, so don't free the array in application code. Compare with `XmListGetSelectedPos`() which returns a copy of the selected position array which should be freed after use.

XmNselectionMode
In Motif 2.0 and later, specifies the effect which keyboard navigation has upon selection. If the value is XmNORMAL_MODE, navigation operations can select the item under the location cursor, deselecting any other items. In XmADD_MODE, navigation operations have no effect on selection. For XmNORMAL_MODE, the selection policy must be browse or extended selection, and for XmADD_MODE, the policy must not be browse.

XmNselectionPolicy
Determines the effect of a selection action. Possible values:

XmSINGLE_SELECT
XmBROWSE_SELECT
XmMULTIPLE_SELECT
XmEXTENDED_SELECT

XmNstringDirection
The direction in which to draw the string. Possible values are:

XmSTRING_DIRECTION_L_TO_R
XmSTRING_DIRECTION_R_TO_L
XmDEFAULT_DIRECTION

In Motif 2.0 and later, the XmNstringDirection resource is obsolete, being subsumed into the XmNlayoutDirection resource. If the XmNlayoutDirection is NULL, and the XmNstringDirection is XmDEFAULT_DIRECTION, the value will be taken from the nearest ancestor which holds the XmQTspecifyLayoutDirection trait. Manager, MenuShell, and VendorShell support this trait.

XmNtopItemPosition
The position of the first item that will be visible in the list. Calling the `XmListSetPos`() routine is the same as setting this resource. In both cases, the first position is specified as 1 and the last position is specified as 0.

XmNverticalScrollBar
When the List is part of a ScrolledList, specifies the widget ID of the ScrollBar created by the List to perform vertical scrolling.

XmNvisibleItemCount
The number of items to display in the work area of the list. This value affects the widget's height. In Motif 1.2 and later, the default value of this resource is dynamic and based on the height of the List, while in Motif 1.1, the default value is 1.

Callback Resources

List defines the following callback resources:

Callback	Reason Constant
XmNbrowseSelectionCallback	XmCR_BROWSE_SELECT
XmNdefaultActionCallback	XmCR_DEFAULT_ACTION
XmNdestinationCallback	XmCR_OK
XmNextendedSelectionCallback	XmCR_EXTENDED_SELECT
XmNmultipleSelectionCallback	XmCR_MULTIPLE_SELECT

Callback	Reason Constant
XmNsingleSelectionCallback	XmCR_SINGLE_SELECT

XmNbrowseSelectionCallback
List of callbacks that are called when a list item is selected using the browse selection policy.

XmNdefaultActionCallback
List of callbacks that are called when a list item is double clicked or KActivate is pressed.

XmNdestinationCallback
List of callbacks that are called when the List is the destination of a transfer operation.

XmNextendedSelectionCallback
List of callbacks that are called when list items are selected using the extended selection policy.

XmNmultipleSelectionCallback
List of callbacks that are called when a list item is selected using the multiple selection policy.

XmNsingleSelectionCallback
List of callbacks that are called when a list item is selected using the single selection policy.

Callback Structure

Each selection callback function is passed the structure below; however, some structure members might be unused because they aren't meaningful for particular callback reasons.

```
typedef struct {
        int         reason;                 /* reason that callback was called   */
        XEvent      *event;                 /* event that triggered callback     */
        XmString    item;                   /* item most recently selected at    */
                                            /* the time event occurred           */
        int         item_length;            /* number of bytes in item member    */
        int         item_position;          /* item's position in XmNitems array */
        XmString    *selected_items;        /* list of items selected at time    */
                                            /* event occurred                    */
        int         selected_item_count;    /* number of items in selected_items */
        int         *selected_item_positions; /* array of integers that mark     */
                                            /* selected items                    */
        char        selection_type;         /* type of the most recent selection */
        char        auto_selection_type;    /* the type of automatic selection   */
```

```
} XmListCallbackStruct;
```

The structure members *event*, *item*, *item_length*, and *item_position* are valid for any value of *reason*.

The structure members *selected_items*, *selected_item_count*, and *selected_item_positions* are valid when the *reason* field has a value of XmCR_MULTIPLE_SELECT or XmCR_EXTENDED_SELECT.

The structure member *selection_type* is valid only when the *reason* field is XmCR_EXTENDED_SELECT.

The structure member *auto_selection_type* is valid only when the resource XmNautomaticSelection is XmAUTO_SELECT, and has the value XmAUTO_UNSET otherwise.

For the strings pointed to by *item* and *selected_items*, as well as for the integers pointed to by *selected_item_positions*, storage is overwritten each time the callback is invoked. Applications that need to save this data should make their own copies of it.

selected_item_positions is an integer array. The elements of the array indicate the positions of each selected item within the List widget's XmNitems array.

selection_type specifies what kind of extended selection was most recently made. One of three values is possible:

```
XmINITIAL       /* selection was the initial selection          */
XmMODIFICATION  /* selection changed an existing selection      */
XmADDITION      /* selection added non-adjacent items to        */
                /*          existing selection */
```

auto_selection_type specifies at what point within the selection the user is. Possible values:

```
XmAUTO_UNSET
XmAUTO_BEGIN
XmAUTO_MOTION
XmAUTO_CANCEL
XmAUTO_NO_CHANGE
XmAUTO_CHANGE
```

Destination callbacks are fully described within the sections covering the Uniform Transfer Model. See `XmTransfer`(s1) for more details. For quick reference, a pointer to the following structure is passed to callbacks on the XmNdestinationCallback list*:*

```
typedef struct {
    int         reason;          /* reason that the callback is invoked    */
    XEvent      *event;          /* event that triggered callback          */
    Atom        selection;       /* requested selection type, as an Atom   */
    XtEnum      operation;       /* type of transfer requested             */
    int         flags;           /* whether destination and source are same */
    XtPointer   transfer_id;     /* unique identifier for the request      */
    XtPointer   destination_data; /* information about the destination     */
    XtPointer   location_data;   /* information about the data             */
    Time        time;            /* time when transfer operation started   */
} XmDestinationCallbackStruct;
```

Inherited Resources

List inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. List sets the default value of XmNnavigationType to XmTAB_GROUP, and sets the default value of XmNlayoutDirection. XmDEFAULT_DIRECTION. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNhighlightThickness	XmPrimitive
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNlayoutDirection	XmPrimitive
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnavigationType	XmPrimitive
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmPrimitive
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmPrimitive	XmNsensitive	Core
XmNbottomShadowPixmap	XmPrimitive	XmNshadowThickness	XmPrimitive
XmNcolormap	Core	XmNtoolTipString	XmPrimitive
XmNconvertCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNdepth	Core	XmNtopShadowPixmap	XmPrimitive
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNheight	Core	XmNunitType	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNuserData	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNwidth	Core
XmNhighlightOnEnter	XmPrimitive	XmNx	Core
XmNhighlightPixmap	XmPrimitive	XmNy	Core

Translations

Event	Action	Event	Action
BSelect Press	ListBeginSelect()	KPageRight	ListRightPage()
BSelect Motion	ListButtonMotion()	KAny	ListQuickNavigate()
BSelect Release	ListEndSelect()	KBeginLine	ListBeginLine()
BExtend Press	ListBeginExtend()	KEndLine	ListEndLine()
BExtend Motion	ListButtonMotion()	KBeginData	ListBeginData()
BExtend Release	ListEndExtend()	MShift KBeginData	ListBeginDataExtend()
BToggle Press	ListBeginToggle()	KEndData	ListEndData()
BToggleMotion	ListButtonMotion()	MShift KEndData	ListEndDataExtend()
BToggle Release	ListEndToggle()	KAddMode	ListAddMode()
BTransfer Press	ListProcessDrag()	KActivate	ListKbdActivate()
KUp	ListPrevItem()	KCopy Press	ListCopyToClipboard()
MShift KUp	ListExtendPrevItem()	KSelectPress	ListKbdBeginSelect()
KDown	ListNextItem()	KSelect Release	ListKbdEndSelect()
MShift KDown	ListExtendNextItem()	KExtend Press	ListKbdBeginExtend()
KLeft	ListLeftChar()	KExtend Release	ListKbdEndExtend()
MCtrl KLeft	ListLeftPage()	Many KCancel	ListKbdCancel()
KRight	ListRightChar	KSelectAll	ListKbdSelectAll()
MCtrl KRight	ListRightPage()	KDeselectAll	ListKbdDeSelectAll()
KPageUp	ListPrevPage	KHelp	PrimitiveHelp()
KPageDown	ListNextPage()	KNextField	PrimitiveNextTabGroup()
KPageLeft	ListLeftPage()	KPrevField	PrimitivePrevTabGroup()

In addition, the translations of List are modified when the XmNenableBtn1Transfer() resource of the XmDisplay object is not XmOFF. The following translations apply under these circumstances:

Event	Action
BSelect Press	ListProcessBtn1(ListBeginSelect)
BSelect Motion	ListProcessBtn1(ListButtonMotion)
BSelect Release	ListProcessBtn1(ListEndSelect)
BExtend Press	ListProcessBtn1(ListBeginExtend)
BExtend Release	ListProcessBtn1(ListEndExtend)
BToggle Press	ListProcessBtn1(ListBeginToggle)

Event	Action
BToggle Release	ListProcessBtn1(ListEndToggle)
BTransfer Press	ListProcessBtn2(ListBeginExtend)
BTransfer Motion	ListProcessBtn2(ListButtonMotion)
BTransfer Release	ListProcessBtn2(ListEndExtend)

Action Routines

List defines the action routines below. The current selection always appears with its foreground and background colors reversed. Note that many List actions have different effects depending on the selection policy and also that some actions apply only for a particular selection policy.

ListAddMode()

Turns add mode on or off.

ListBeginData()

Moves the cursor to the first list item. If keyboard selection is in normal mode, this action also selects the first item after deselecting any earlier selection and invokes the callbacks specified either by XmNbrowseSelectionCallback or by XmNextendedSelectionCallback (as dictated by the selection policy).

ListBeginDataExtend()

Multiple selection: moves the cursor to the first list item.

Extended selection: moves the cursor to the first list item, cancels any current extended selection, selects (or deselects) all items from the first item to the current anchor, and invokes the callbacks specified by XmNextendedSelectionCallback.

ListBeginExtend()

Extended selection: cancels any current extended selection, selects (or deselects) all items from the pointer location to the current anchor, and invokes the callbacks specified by XmNextendedSelectionCallback *(if the XmNautomaticSelection* resource is True*).*

ListBeginLine()

Scrolls the List's viewing area horizontally to its beginning.

ListBeginSelect()

Single selection: selects or deselects the item under the pointer after deselecting any previous selection.

Browse selection: selects the item under the pointer after deselecting any previous selection and invokes the callbacks specified by XmNbrowseSelectionCallback *if the XmNautomaticSelection* resource is True.

Multiple selection: selects or deselects the item under the pointer, leaving previous selections unaffected.

Extended selection: selects the item under the pointer after deselecting any previous selection, marks this item as the current anchor, and invokes the callbacks specified by XmNextendedSelectionCallback *if the XmNautomaticSelection* resource is True.

ListBeginToggle()

Extended selection: keeps the current selection but shifts the anchor to the item under the pointer. This item's selection state is toggled, and if XmNautomaticSelection *is True*, the extended selection callbacks are invoked.

ListButtonMotion()

Browse selection: selects the item under the pointer after deselecting any previous selection and invokes the browse selection callbacks if XmNautomaticSelection *is True* and the pointer moved over a new item.

Extended selection: cancels any current extended selection, selects (or deselects) all items from the pointer location to the current anchor, and invokes the extended selection callbacks if XmNautomaticSelection *is True* and the pointer moved over a new item.

In addition, when the pointer moves outside a ScrolledList widget, the list scrolls in sync with the pointer motion.

ListCopyToClipboard()

In Motif 1.2 and later, this action copies the selected list items to the clipboard. The items are copied as a single compound string, with a new line between each item.

ListEndData()
: Moves the cursor to the last list item. If keyboard selection is in normal mode, this action also selects the last item after deselecting any earlier selection and invokes the appropriate callbacks (browse selection or extended selection).

ListEndDataExtend()
: *Multiple selection*: moves the cursor to the last list item.

  Extended selection: moves the cursor to the last list item, cancels any current extended selection, selects (or deselects) all items from the last item to the current anchor, and invokes the extended selection callbacks.

ListEndExtend()
: *Extended selection*: moves the cursor to the last item whose selection state was switched, and invokes the extended selection callbacks if XmNautomaticSelection is *False*.

ListEndLine()
: Scrolls the List's viewing area horizontally to its beginning.

ListEndSelect()
: *Single selection or multiple selection*: moves the cursor to the last item whose selection state was switched, and invokes the appropriate selection callbacks.

  Browse selection or extended selection: same as above, except that the appropriate callbacks are called only if XmNautomaticSelection *is False*.

ListEndToggle()
: *Extended selection*: moves the cursor to the last item whose selection state was switched, and invokes the extended selection callbacks if XmNautomaticSelection *is False*.

ListExtendNextItem()
ListExtendPrevItem()
: *Extended selection*: adds the next/previous item to an extended selection and invokes the extended selection callbacks.

ListKbdActivate()
: Invokes the default action callbacks.

ListKbdBeginExtend()

This action is the keyboard's complement to the mouse-activated ListBeginExtend() action.

Extended selection: cancels any current extended selection and selects (or deselects) all items from the cursor to the current anchor.

ListKbdBeginSelect()

This action is the keyboard's complement to the mouse-activated ListBeginSelect*()* action.

Single selection: selects or deselects the item at the cursor after deselecting any previous selection.

Browse selection: selects the item at the cursor after deselecting any previous selection and invokes the browse selection callbacks if XmNautomaticSelection *is True*.

Multiple selection: selects or deselects the item at the cursor, leaving previous selections unaffected.

Extended selection: shifts the anchor to the item at the cursor. In normal mode, this item is selected after any previous selection is deselected; in add mode, this item's state is toggled, and the current selection remains unaffected. This action calls the extended selection callbacks if XmNautomaticSelection *is True*.

ListKbdCancel()

Extended selection: cancels an extended selection and restores the items to their previous selection state.

ListKbdDeSelectAll()

Deselects all list items and calls the appropriate selection callbacks. This action applies to all selection modes except browse selection because this mode requires one item to remain selected at all times. In extended selection with keyboard Normal Mode and an XmNkeyboardFocusPolicy of XmEXPLICIT, the item at the cursor remains selected after this action is applied.

ListKbdEndExtend()

Extended selection: calls the extended selection callbacks if XmNautomaticSelection *is False*.

ListKbdEndSelect()

Single selection or multiple selection: calls the appropriate selection callbacks. If XmNautomaticSelection *is False*, this action applies under any of the four selection policies.

ListKbdSelectAll()

Single selection or browse selection: selects the item at the cursor and calls the appropriate selection callbacks.

Multiple selection or extended selection: selects all list items and calls the appropriate selection callbacks.

ListLeftChar()
ListLeftPage()

Scrolls the list either one character or one page to the left.

ListNextItem()

Moves the cursor to the next list item and has the following additional operations:

Browse selection: selects this item, deselects any previously selected item(s), and calls the browse selection callbacks.

Extended selection: in normal mode, selects this item and moves the anchor there, deselects any previously selected item(s), and calls the extended selection callbacks. In add mode, neither the selection nor the anchor is affected.

ListNextPage()

Moves the cursor by scrolling the list to the list item at the top of the next page and has the same additional operations as ListNextItem().

ListPrevItem()

Same as *ListNextItem()*, going back one item instead.

ListPrevPage() *Same as ListNextPage()*, going back one page instead.

ListProcessDrag()

In Motif 1.2 and later, this action initiates a drag and drop operation using the selected items, where each item is separated by a newline. If BTransfer is pressed over an unselected item, only that item is used in the drag and drop operation.

ListProcessBtn1(string)

In Motif 2.0 and later, the XmDisplay resource XmNenableBtn1Transfer configures the integration of selection and transfer operations on Button 1.

If XmNenableBtn1Transfer is XmOFF, if no data transfer has been initialised, the action specified by string is invoked to initiate selection. Possible values for string are:

ListBeginExtend
ListEndExtend
ListButtonMotion
ListBeginSelect
ListEndSelect
ListBeginToggle
ListEndToggle

ListProcessBtn2(string)
In Motif 2.0 and later, if the XmDisplay resource XmNenableBtn1Transfer has the value XmBUTTON2_TRANSFER, the actions for extending List selection are bound to Button 2, and data transfer is initiated. If the resource has the value XmBUTTON2_ADJUST, the action specified by string is invoked to extend selection. Possible values for string are:

ListBeginExtend
ListEndExtend
ListButtonMotion

ListQuickNavigate()
In Motif 2.0 and later, navigates to an item if XmNmatchBehavior is XmQUICK_NAVIGATE.

ListRightChar()
ListRightPage() Scrolls the list either one character or one page to the right.

ListScrollCursorVertically()
Makes the item with the keyboard focus visible.

PrimitiveHelp()
Calls the help callbacks for this widget.

PrimitiveNextTabGroup()
PrimitivePrevTabGroup()
Moves the keyboard focus to the beginning of the next or previous tab group, wrapping around if necessary.

Additional Behavior

List has the following additional behavior:

<Double Click>
: Calls the XmNdefaultActionCallback callbacks.

<FocusIn>
: If the keyboard focus policy is explicit, sets the focus and draws the location cursor.

<FocusOut>
: If the keyboard focus policy is explicit, removes the focus and erases the location cursor.

See Also

XmCreateObject(s1), XmListAddItem(s1), XmListAddItemUnselected(s1), XmListDeleteAllItems(s1), XmListDeleteItem(s1), XmListDeleteItemsPos(s1), XmListDeletePos(s1), XmListDeletePositions(s1), XmListDeselectAllItems(s1), XmListDeselectItem(s1), XmListDeselectPos(s1), XmListGetKbdItemPos(s1), XmListGetMatchPos(s1), XmListGetSelectedPos(s1), XmListItemExists(s1), XmListItemPos(s1), XmListPosSelected(s1), XmListPosToBounds(s1), XmListReplaceItems(s1), XmListReplaceItemsPos(s1), XmListReplaceItemsPosUnselected(s1), XmListReplaceItemsUnselected(s1), XmListReplacePositions(1), XmListSelectItem(1), XmListSelectPos(1), XmListSetAddMode(1), XmListSetBottomItem(1), XmListSetBottomPos(1), XmListSetHorizPos(1), XmListSetItem(1), XmListSetKbdItemPos(1), XmListSetPos(1), XmListUpdateSelectedList(1), XmListYToPos(1), Core(2), XmDisplay(2). XmPrimitive(2), XmTransfer(1).

Name

XmMainWindow widget class – the standard layout widget for an application's primary window.

Synopsis

Public Header:
<Xm/MainW.h>

Class Name:
XmMainWindow

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmScrolledWindow → XmMainWindow

Class Pointer:
xmMainWindowWidgetClass

Instantiation:
widget = `XmCreateMainWindow` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmMainWindowWidgetClass,...)

Functions/Macros:
`XmCreateMainWindow`(), `XmMainWindowSep1`(), `XmMainWindowSep2`(), `XmMainWindowSep3`(), `XmMainWindowSetAreas`(), `XmIsMainWindow`()

Description

MainWindow provides the standard appearance for the primary window of an application. MainWindow supports five standard areas: a MenuBar, a command window, a work region, a message window, and two ScrollBars (one horizontal and one vertical). An application can use as many or as few of these areas as necessary; they are all optional. A MainWindow can also display three Separator widgets for dividing one area from another.

Each of the MainWindow regions is associated with a MainWindow resource; `XmMainWindowSetAreas`() sets the associated resources. If an application does not call `XmMainWindowSetAreas`(), the widget may still set some of the standard regions. When a MenuBar child is added to a MainWindow, if XmNmenuBar has not been set, it is set to the MenuBar child. When a Command child is added to a MainWindow, if XmNcommand has not been set, it is set to the

Command child. If ScrollBars are added as children, the XmNhorizontalScrollBar and XmNverticalScrollBar resources may be set if they have not already been specified. Any child that is not one of these types is used for the XmNworkWindow. If you want to be certain about which widgets are used for the different regions, it is wise to call `XmMainWindowSetAreas`() explicitly.

Traits

MainWindow uses the XmQTmenuSystem trait.

New Resources

MainWindow defines the following resources:

Name	Class	Type	Default	Access
XmNcommandWindow	XmCCommandWindow	Widget	NULL	CSG
XmNcommandWindowLocation	XmCCommandWindowLocation	unsigned char	XmCOMMAND_ABOVE_WORKSPACE	CG
XmNmainWindowMarginHeight	XmCMainWindowMarginHeight	Dimension	0	CSG
XmNmainWindowMarginWidth	XmCMainWindowMarginWidth	Dimension	0	CSG
XmNmenuBar	XmCMenuBar	Widget	NULL	CSG
XmNmessageWindow	XmCMessageWindow	Widget	NULL	CSG
XmNshowSeparator	XmCShowSeparator	Boolean	False	CSG

XmNcommandWindow
The widget ID of the command window child.

XmNcommandWindowLocation
One of two positions for the command window. Possible values:

```
XmCOMMAND_ABOVE_WORKSPACE   /* default; appears below menu bar */
XmCOMMAND_BELOW_WORKSPACE   /* appears between work and      */
                            /*     message windows           */
```

XmNmainWindowMarginHeight
The margin on the top and bottom of the MainWindow widget. This resource overrides the corresponding margin resource in the ScrolledWindow widget.

XmNmainWindowMarginWidth
The margin on the right and left of the MainWindow widget. This resource overrides the corresponding margin resource in the ScrolledWindow widget.

XmNmenuBar
The widget ID of the menu bar child.

XmNmessageWindow
The widget ID of the message window child.

XmNshowSeparator
If True, separators are displayed between components of the MainWindow widget. If False (default), separators are not displayed.

Inherited Resources

MainWindow inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of XmNborderWidth is reset to 0 by XmManager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNnavigationType	XmManager
XmNancestorSensitive	Core	XmNnumChildren	Composite
XmNautoDragModel	XmScrolledWindow	XmNpopupHandlerCallback	XmManager
XmNbackground	Core	XmNscreen	Core
XmNbackgroundPixmap	Core	XmNscrollBarDisplayPolicy	XmScrolledWindow
XmNborderColor	Core	XmNscrollBarPlacement	XmScrolledWindow
XmNborderPixmap	Core	XmNscrolledWindowMarginHeight	XmScrolledWindow
XmNborderWidth	Core	XmNscrolledWindowMarginWidth	XmScrolledWindow
XmNbottomShadowColor	XmManager	XmNscrollingPolicy	XmScrolledWindow
XmNbottomShadowPixmap	XmManager	XmNsensitive	Core
XmNchildren	Composite	XmNshadowThickness	XmManager
XmNclipWindow	XmScrolledWindow	XmNspacing	XmScrolledWindow
XmNcolormap	Core	XmNstringDirection	XmManager
XmNdepth	Core	XmNtopShadowColor	XmManager
XmNdestroyCallback	Core	XmNtopShadowPixmap	XmManager
XmNforeground	XmManager	XmNtranslations	Core
XmNheight	Core	XmNtraversalOn	XmManager
XmNhelpCallback	XmManager	XmNtraverseObscuredCallback	XmScrolledWindow
XmNhighlightColor	XmManager	XmNunitType	XmManager
XmNhighlightPixmap	XmManager	XmNuserData	XmManager
XmNhorizontalScrollBar	XmScrolledWindow	XmNverticalScrollBar	XmScrolledWindow
XmNinitialFocus	XmManager	XmNvisualPolicy	XmScrolledWindow
XmNinitialResourcesPersistent	Core	XmNwidth	Core
XmNinsertPosition	Composite	XmNworkWindow	XmScrolledWindow
XmNlayoutDirection	XmManager	XmNx	Core

Resource	Inherited From	Resource	Inherited From
XmNmappedWhenManaged	Core	XmNy	Core

Translations

The translations for MainWindow are inherited from ScrolledWindow.

See Also

XmCreateObject(1), XmMainWindowSep(1), XmMainWindowSetAreas(1), Composite(2), Constraint(2), Core(2), XmManager(2), XmScrolledWindow(2).

Name

XmManager widget class – the fundamental class for Motif widgets that manage children.

Synopsis

Public Header:
<Xm/Xm.h>

Class Name:
XmManager

Class Hierarchy:
Core → Composite → Constraint → XmManager

Class Pointer:
xmManagerWidgetClass

Instantiation:
Manager is a meta-class and is not normally instantiated.

Functions/Macros:
`XmIsManager`()

Description

Manager is a superclass for Motif widget classes that contain children. Manager supports geometry management by providing resources for visual shadows and highlights and for keyboard traversal mechanisms.

The default values of the color resources for the foreground, background, top and bottom shadows, and highlighting are set dynamically. If no colors are specified, they are generated automatically. On a monochrome system, black and white colors are selected. On a color system, four colors are selected that provide the appropriate shading for the 3-D visuals. When the background color is specified, the shadow colors are selected to provide the appropriate 3-D appearance and foreground and highlight colors are selected to provide the necessary contrast. The colors are generated when the widget is created; using `XtSetValues`() to change the background does not change the other colors. With Motif 1.2, use `XmChangeColor`() to change the associated colors when the background color is changed.

In Motif 2.0 and later, the way in which popup menus are posted for particular widgets is rationalized by including within Manager and Primitive popup handler callback resources. When a popup menu is created, an event handler may be automatically installed upon the parent Manager to catch posting events. Once in receipt of the posting event, the Manager's XmNpopupHandlerCallback list is invoked in order to determine which of the various popup menus available is

currently required. Whether the handlers are automatically installed depends upon the value of the XmNpopupEnabled resource of the menus concerned.

Traits

Manager holds the XmNspecifyLayoutDirection, XmNspecifyUnitType, and XmNaccessColors traits, which are inherited by any derived classes.

New Resources

Manager defines the following resources:

Name	Class	Type	Default	Access
XmNbottomShadowColor	XmCBottomShadowColor	Pixel	dynamic	CSG
XmNbottomShadowPixmap	XmCBottomShadowPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNforeground	XmCForeground	Pixel	dynamic	CSG
XmNhighlightColor	XmCHighlightColor	Pixel	dynamic	CSG
XmNhighlightPixmap	XmCHighlightPixmap	Pixmap	dynamic	CSG
XmNinitialFocus	XmCInitialFocus	Widget	NULL	CSG
XmNlayoutDirection	XmCLayoutDirection	XmDirection	dynamic	CG[a]
XmNnavigationType	XmCNavigationType	XmNavigationType	XmTAB_GROUP	CSG
XmNshadowThickness	XmCShadowThickness	Dimension	0	CSG
XmNstringDirection	XmCStringDirection	XmStringDirection	dynamic	CG
XmNtopShadowColor	XmCTopShadowColor	Pixel	dynamic	CSG
XmNtopShadowPixmap	XmCTopShadowPixmap	Pixmap	dynamic	CSG
XmNtraversalOn	XmCTraversalOn	Boolean	True	CSG
XmNunitType	XmCUnitType	unsigned char	dynamic	CSG
XmNuserData	XmCUserData	XtPointer	NULL	CSG

a. Erroneously given as CSG in 2nd edition.

XmNbottomShadowColor
: The color used in drawing the border shadow's bottom and right sides, but only if XmNbottomShadowPixmap is NULL.

XmNbottomShadowPixmap
: The pixmap used in drawing the border shadow's bottom and right sides.

XmNforeground
: The foreground color used by Manager widgets.

XmNhighlightColor
: The color used in drawing the highlighting rectangle, but only if XmNhighlight-Pixmap is XmUNSPECIFIED_PIXMAP.

XmNhighlightPixmap
The pixmap used in drawing the highlighting rectangle.

XmNinitialFocus
In Motif 1.2, the widget ID of the widget that receives the keyboard focus when the manager is a child of a shell and the shell receives the keyboard focus for the first time.

XmNlayoutDirection
In Motif 2.0 and later, specifies the direction in which components of the Manager are laid out. If unspecified, the value is inherited from the nearest ancestor holding the XmQTspecifyLayoutDirection trait. Manager, MenuShell, and VendorShell hold this trait. Possible values:

XmLEFT_TO_RIGHT
XmRIGHT_TO_LEFT
XmBOTTOM_TO_TOP
XmTOP_TO_BOTTOM
XmBOTTOM_TO_TOP_LEFT_TO_RIGHT
XmBOTTOM_TO_TOP_RIGHT_TO_LEFT
XmTOP_TO_BOTTOM_LEFT_TO_RIGHT
XmTOP_TO_BOTTOM_RIGHT_TO_LEFT
XmLEFT_TO_RIGHT_BOTTOM_TO_TOP
XmRIGHT_TO_LEFT_BOTTOM_TO_TOP
XmLEFT_TO_RIGHT_TOP_TO_BOTTOM
XmRIGHT_TO_LEFT_TOP_TO_BOTTOM

XmNnavigationType
Determines the way in which a Manager widget is traversed during keyboard navigation. Possible values:

```
XmNONE                  /* exclude from keyboard navigation      */
XmTAB_GROUP             /* include in keyboard navigation        */
XmSTICKY_TAB_GROUP      /* include in keyboard navigation,       */
                        /* even if XmAddTabGroup() was called    */
XmEXCLUSIVE_TAB_GROUP   /* application defines order of navigation */
```

XmNshadowThickness
The thickness of the shadow border. This resource is dynamically set to 1 in a top-level window and 0 otherwise.

XmNstringDirection
The direction in which to draw the string. Possible values are:

XmSTRING_DIRECTION_L_TO_R
XmSTRING_DIRECTION_R_TO_L
XmSTRING_DIRECTION_DEFAULT

In Motif 2.0 and later, the XmNstringDirection resource is obsolete, and is subsumed into the XmNlayoutDirection resource. If the XmNlayoutDirection is NULL, and the XmNstringDirection is XmSTRING_DIRECTION_DEFAULT, the value will be taken from the nearest ancestor which holds the XmQTspecify-LayoutDirection trait. Manager itself, MenuShell, and VendorShell hold this trait.

XmNtopShadowColor
The color used in drawing the border shadow's top and left sides. (Used only if XmNtopShadowPixmap is NULL.)

XmNtopShadowPixmap
The pixmap used in drawing the border shadow's top and left sides.

XmNtraversalOn
If True (default), traversal of this widget is made possible.

XmNunitType
The measurement units to use in resources that specify a size or position--for example, any resources of type Dimension (whose names generally include one of the words "Margin", "Height", "Width", "Thickness", or "Spacing"), as well as the offset resources defined by Form. For a widget whose parent is a manager, the default value is copied from this parent (provided the value hasn't been explicitly set by the application); otherwise, the default is XmPIXELS. Possible values:

XmPIXELS
Xm100TH_POINTS
Xm100TH_MILLIMETERS
Xm100TH_FONT_UNITS
Xm1000TH_INCHES
XmINCHES (2.0)
XmPOINTS (2.0)
XmFONT_UNITS (2.0)

XmNuserData
A pointer to data that the application can attach to the widget. This resource is unused internally.

Callback Resources

Manager defines the following callback resources:

Callback	Reason Constant
XmNhelpCallback	XmCR_HELP
XmNpopupHandlerCallback	XmCR_POST XmCR_REPOST

XmNhelpCallback
List of callbacks that are called when help is requested.

XmNpopupHandlerCallback
In Motif 2.0 and later, the list of callbacks invoked in order to determine which popup menu to display.

Callback Structure

With the exception of a popup handler callback, each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* set to XmCR_HELP                 */
    XEvent      *event;         /* event structure that triggered callback */
} XmAnyCallbackStruct;
```

A popup handler callback is passed a pointer to the following structure:

```
typedef struct {
    int         reason;         /* the reason the callback is invoked   */
    XEvent      *event;         /* event structure that triggered callback */
    Widget      menuToPost;     /* the menu to post                 */
    Boolean     postIt;         /* whether to continue posting      */
    Widget      target;         /* the manager descendant issuing request */
} XmPopupHandlerCallbackStruct;
```

reason is either XmCR_POST[1] or XmCR_REPOST[2]. XmCR_POST is the normal menu post request. XmCR_REPOST is called when the menu is unposted because of event replay.

menuToPost is the suggested menu to be posted. Alter the element if a different menu is required.

postIt is a flag indicating whether the posting operation is to continue once the callback has finished. The default value is True if reason is XmPOST, otherwise False.

target is the widget or gadget which the Manager believes best describes the source of the posting event. The algorithm performs a recursive descent, matching the received event against the location of managed children.

1.Erroneously given as XmPOST in 2nd edition.

2.Erroneously given as XmREPOST in 2nd edition.

Inherited Resources

Manager inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. Manager resets the default value of XmNborderWidth from 1 to 0.

Name	Inherited From	Name	Inherited From
XmNaccelerators	Core	XmNheight	Core
XmNancestorSensitive	Core	XmNinsertPosition	Composite
XmNbackground	Core	XmNinitialResourcesPersistent	Core
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNscreen	Core
XmNborderWidth	Core	XmNsensitive	Core
XmNchildren	Composite	XmNtranslations	Core
XmNcolormap	Core	XmNwidth	Core
XmNdepth	Core	XmNx	Core
XmNdestroyCallback	Core	XmNy	Core

Translations

For Manager widgets that have gadget children:

Event	Action
BAny Motion	ManagerGadgetButtonMotion()
BSelect Press	ManagerGadgetArm()
BSelect Click	ManagerGadgetActivate()
BSelect Release	ManagerGadgetActivate()
BSelect Press 2+	ManagerGadgetMultiArm()
BSelect Release 2+	ManagerGadgetMultiActivate
BTransfer Press	ManagerGadgetDrag()
KActivate	ManagerParentActivate() (1.2)
KCancel	ManagerParentCancel()
KPrevField	ManagerGadgetPrevTabGroup()
KNextField	ManagerGadgetNextTabGroup()
KUp	ManagerGadgetTraverseUp()
KDown	ManagerGadgetTraverseDown()
KLeft	ManagerGadgetTraverseLeft()

Event	Action
KRight	ManagerGadgetTraverseRight()
KSelect	ManagerGadgetSelect()
KBeginLine	ManagerGadgetTraverseHome()
KHelp	ManagerGadgetHelp(), ManagerGadgetSelect() (1.2)
KAny	ManagerGadgetKeyInput()

Action Routines

The action routines for a Manager widget affect the gadget child that has the keyboard focus. The descriptions below refer to the gadget that has the focus.

ManagerGadgetActivate()
: Activates the gadget.

ManagerGadgetArm()
: Arms the gadget.

ManagerGadgetButtonMotion()
: Triggers the mouse motion event that the gadget received.

ManagerGadgetDrag()
: In Motif 1.2, initiates a drag and drop operation using the contents of a gadget's label.

ManagerGadgetHelp()
: Invokes the list of callbacks specified by the gadget's XmNhelpCallback resource. If the gadget doesn't have any help callbacks, the ManagerGadgetHelp() routine invokes those associated with the nearest ancestor that has them.

ManagerGadgetKeyInput()
: Triggers the keyboard event that the gadget received.

ManagerGadgetMultiActivate()
: Processes a multiple click of the mouse.

ManagerGadgetMultiArm()
: Processes a multiple press of the mouse button.

ManagerGadgetNextTabGroup()
ManagerGadgetPrevTabGroup()
: Traverses to the beginning of the next/previous tab group, wrapping if necessary.

ManagerGadgetSelect()
: Arms and activates the gadget.

ManagerGadgetTraverseDown()
ManagerGadgetTraverseUp()
: Within the same tab group, descends/ascends to the item below/above the gadget, wrapping if necessary.

ManagerGadgetTraverseHome()
: Changes the focus to the first item in the tab group.

ManagerGadgetTraverseLeft()
ManagerGadgetTraverseRight()
: Within the same tab group, traverses to the item on the left/right of the gadget, wrapping if necessary.

ManagerGadgetTraverseNext()
ManagerGadgetTraversePrev()
: Within the same tab group, traverses to the next/previous item, wrapping if necessary.

ManagerParentActivate()
: In Motif 1.2, passes the KActivate event to the parent if it is a manager.

ManagerParentCancel()
: In Motif 1.2, passes the KCancel event to the parent if it is a manager.

Additional Behavior

Manager has the following additional behavior:

<FocusIn>
: If the event occurs in a gadget, highlights the gadget and gives it the focus under the explicit keyboard focus policy.

<FocusOut>
: If the event occurs in a gadget, unhighlights the gadget and removes the focus under the explicit keyboard focus policy.

See Also

`Composite`(2), `Constraint`(2), *Core*(2), `XmGadget`(2).

Name

XmMenuBar – a type of RowColumn widget used as a menu bar.

Synopsis

Public Header:
<Xm/RowColumn.h>

Class Name:
XmRowColumn

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmRowColumn

Class Pointer:
xmRowColumnWidgetClass

Instantiation:
widget = `XmCreateMenuBar` (parent, name,...)

Functions/Macros:
`XmCreateMenuBar`(), `XmCreateSimpleMenuBar`(), `XmVaCreateSimpleMenuBar`(), `XmIsRowColumn`()

Description

An XmMenuBar is an instance of a RowColumn widget that is normally used for constructing a pulldown menu system. An application typically places a MenuBar across the top of the main application window. CascadeButtons are added to the MenuBar and pulldown menus are associated with each of the CascadeButtons.

MenuBar is a RowColumn widget whose XmNrowColumnType resource is set to XmMENU_BAR. The XmNentryClass resource is set to xmCascadeButtonWidgetClass and XmNisHomogeneous is set to True, so that only CascadeButtons can be added to the widget. The XmNmenuAccelerator resource is set to KMenuBar and XmNmenuPost is set to BSelect Press. The XmNmenuHelpWidget resource can be set to specify the CascadeButton for the Help menu. The XmNorientation resource is set to XmHORIZONTAL.

A MenuBar can be created using `XmCreateMenuBar`(). In this case, the MenuBar does not automatically contain any CascadeButtons; they are added by the application.

A MenuBar can also be created by `XmCreateSimpleMenuBar`(), which automatically creates the MenuBar with the specified CascadeButtonGadgets as children. This routine uses the RowColumn resources associated with the creation of simple menus. For a MenuBar, the only type allowed in the XmNbuttonType resource is XmCASCADEBUTTON. The name of each CascadeButtonGadget is

button_*n*, where *n* is the number of the button, ranging from 0 to 1 less than the number of buttons in the MenuBar.

Default Resource Values

A MenuBar sets the following default values for RowColumn resources:

Name	Default
XmNentryClass	xmCascadeButtonWidgetClass
XmNisHomogenous	True
XmNmenuAccelerator	KMenuBar
XmNmenuPost	BSelect Press
XmNorientation	XmHORIZONTAL
XmNrowColumnType	XmMENU_BAR

See Also

`XmCreateObject`(1), `XmVaCreateSimpleMenuBar`(1), `XmCascadeButton`(2), `XmRowColumn`(2).

Name

XmMenuShell widget class – a shell widget meant to contain popup and pull-down menu panes.

Synopsis

Public Header:
<Xm/MenuShell.h>

Class Name:
XmMenuShell

Class Hierarchy:
Core → Composite → Shell → OverrideShell → XmMenuShell

Class Pointer:
xmMenuShellWidgetClass

Instantiation:
widget = `XmCreateMenuShell` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmMenuShellWidgetClass,...)

Functions/Macros:
`XmCreateMenuShell`(), `XmCreatePopupMenu`(), `XmCreatePulldownMenu`(),
`XmIsMenuShell`()

Description

MenuShell is a subclass of OverrideShell that is meant to contain only popup or pulldown menu panes. Most application writers do not need to create MenuShell widgets explicitly because they are created automatically by the convenience routines XmCreatePopupMenu() and XmCreatePulldownMenu().

If you do not use the convenience functions and create your own MenuShell widgets, the type of menu system being built determines the parent to specify for the MenuShell. For a top-level popup menu, specify the widget from which it will pop up. For a pulldown menu pane from the menu bar, specify the menu bar. For a pulldown menu pane from another pulldown menu or a popup menu, specify the menu pane from which it is pulled down. For pulldown menu in an option menu, specify the option menu's parent.

Traits

MenuShell holds the XmQTspecifyLayoutDirection and XmQTspecifyRenderTable traits, which are inherited by any derived classes, and uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

New Resources

MenuShell defines the following resource:

Name	Class	Type	Default	Access
XmNbuttonFontList	XmCButtonFontList	XmFontList	dynamic	CSG
XmNbuttonRenderTable	XmCButtonRenderTable	XmRenderTable	dynamic	CSG
XmNdefaultFontList	XmCDefaultFontList	XmFontList	dynamic	CG
XmNlabelFontList	XmCLabelFontList	XmFontList	dynamic	CSG
XmNlabelRenderTable	XmCLabelRenderTable	XmRenderTable	dynamic	CSG
XmNlayoutDirection	XmCLayoutDirection	XmDirection	dynamic	CG[a]

a. Erroneously given as CSG in 2nd edition.

XmNbuttonFontList
: The font list used for the button children of the MenuShell widget. In Motif 2.0 and later, the XmfontList is considered obsolete, and the Rendition Table is the preferred method of setting appearance. Any XmNbuttonRenderTable value will take precedence.

XmNbuttonRenderTable
: Specifies the render table used for button children of the MenuShell widget. If the value is NULL, this will be inherited from the nearest ancestor with the XmQTspecifyRenderTable trait, taking the XmBUTTON_RENDER_TABLE value from the ancestor so found. The BulletinBoard, VendorShell, and MenuShell widgets and derived classes set this trait.

XmNdefaultFontList
: The default font list for the children of the MenuShell widget. This resource is obsolete in Motif 1.2.

XmNlabelFontList
: The font list used for the label children of the MenuShell widget. In Motif 2.0 and later, the XmFontList is considered obsolete, and the Rendition Table is the preferred method of setting appearance. Any XmNlabelRenderTable value will take precedence.

XmNlabelRenderTable
: Specifies the render table used for label children of the MenuShell widget. If the value is NULL, this will be inherited from the nearest ancestor with the XmQTspecifyRenderTable trait, taking the XmLABEL_RENDER_TABLE value from the ancestor so found. The BulletinBoard, VendorShell, and MenuShell widgets and derived classes set this trait.

XmNlayoutDirection
In Motif 2.0 and later, specifies the default direction in which descendants of the MenuShell are laid out.

XmLEFT_TO_RIGHT
XmRIGHT_TO_LEFT
XmBOTTOM_TO_TOP
XmTOP_TO_BOTTOM
XmBOTTOM_TO_TOP_LEFT_TO_RIGHT
XmBOTTOM_TO_TOP_RIGHT_TO_LEFT
XmTOP_TO_BOTTOM_LEFT_TO_RIGHT
XmTOP_TO_BOTTOM_RIGHT_TO_LEFT
XmLEFT_TO_RIGHT_BOTTOM_TO_TOP
XmRIGHT_TO_LEFT_BOTTOM_TO_TOP
XmLEFT_TO_RIGHT_TOP_TO_BOTTOM
XmRIGHT_TO_LEFT_TOP_TO_BOTTOM

Inherited Resources

MenuShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. MenuShell sets the default value of XmNallowShellResize to True and XmNborderWidth to 0. The default values of XmNoverrideRedirect and XmNsaveUnder are set to True by OverrideShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinitialResourcesPersistent	Core
XmNallowShellResize	Shell	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNmappedWhenManaged	Core
XmNbackground	Core	XmNnumChildren	Composite
XmNbackgroundPixmap	Core	XmNoverrideRedirect	Shell
XmNborderColor	Core	XmNpopdownCalback	Shell
XmNborderPixmap	Core	XmNpopupCallback	Shell
XmNborderWidth	Core	XmNsaveUnder	Shell
XmNchildren	Composite	XmNscreen	Core
XmNcolormap	Core	XmNsensitive	Core
XmNcreatePopupChildProc	Shell	XmNtranslations	Core
XmNdepth	Core	XmNvisual	Shell
XmNdestroyCallback	Core	XmNwidth	Core
XmNgeometry	Shell	XmNx	Core
XmNheight	Core	XmNy	Core

Translations

Event	Action
BSelect Press	ClearTraversal()
BSelect Release	MenuShellPopdownDone()

Action Routines

MenuShell defines the following action routines:

ClearTraversal()
: Shuts off keyboard traversal within this menu, turns on mouse traversal, and unposts any submenus that this menu posted.

MenuShellPopdownDone()
: Unposts the menu tree and restores the previous focus.

MenuShellPopdownOne()
: Like MenuShellPopdownDone() except that it unposts only one level of the menu tree. In a top-level pulldown menu pane attached to a menu bar, this action routine disarms the cascade button and the menu bar.

See Also

`XmCreateObject`(1), `Composite`(2), `Core`(2), `OverrideShell`(2), `Shell`(2), `XmRowColumn`(2).

Name

XmMessageBox widget class –a composite widget used for creating message dialogs.

Synopsis

Public Header:
<Xm/MessageB.h>

Class Name:
XmMessageBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmBulletinBoard → XmMessageBox

Class Pointer:
xmMessageBoxWidgetClass

Instantiation:
widget = `XmCreateMessageBox` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmMessageBoxWidgetClass,...)

Functions/Macros:
`XmCreateErrorDialog`(), `XmCreateInformationDialog`(), `XmCreateMessageBox`(), `XmCreateMessageDialog`(), `XmCreateQuestionDialog`(), `XmCreateTemplateDialog`(), `XmCreateWarningDialog`(), `XmCreateWorkingDialog`(), `XmIsMessageBox`(), `XmMessageBoxGetChild`()

Description

MessageBox is composite widget that is used for creating simple message dialog boxes, which normally present transient messages. A MessageBox usually contains a message symbol, a message, three PushButtons, and a separator between the message and the buttons. The names of the symbol and the separator gadgets are Symbol and Separator. In Motif 1.2, the default symbols and button labels can be localized. The XmNdialogType resource controls the type of message symbol that is displayed. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Cancel**, and **Help** by default.

You can customize a MessageBox by removing existing children or adding new children. Use `XmMessageBoxGetChild`() to retrieve the widget ID of an existing child and then unmanage the child[1]. With Motif 1.2, multiple widgets can be added as children of a MessageBox. If a menu bar is added, it is placed at the top of the window. Any buttons are placed after the **OK** button. Any additional children are placed below the message. In Motif 1.1, only a single widget can be added as a child of a MessageBox. This child is placed below the message and acts as a work area.

In Motif 1.2 and later, a XmNdialogType of XmDIALOG_TEMPLATE create a TemplateDialog which contains nothing but a separator by default. Specifying callback, label string, or pixmap symbol resources causes the appropriate children of the MessageBox to be created.

Traits

MessageBox uses the XmQTactivatable trait.

New Resources

MessageBox defines the following resources:

Name	Class	Type	Default	Access
XmNcancelLabelString	XmCCancelLabelString	XmString	dynamic	CSG
XmNdefaultButtonType	XmCDefaultButtonType	unsigned char	XmDIALOG_OK_BUTTON	CSG
XmNdialogType	XmCDialogType	unsigned char	XmDIALOG_MESSAGE	CSG
XmNhelpLabelString	XmCHelpLabelString	XmString	dynamic	CSG
XmNmessageAlignment	XmCAlignment	unsigned char	XmALIGNMENT_BEGINNING	CSG
XmNmessageString	XmCMessageString	XmString	""[a]	CSG
XmNminimizeButtons	XmCMinimizeButtons	Boolean	False	CSG
XmNokLabelString	XmCOkLabelString	XmString	dynamic	CSG
XmNsymbolPixmap	XmCPixmap	Pixmap	dynamic	CSG

a. Strictly speaking, NULL, which is internally mapped through the empty string.

XmNcancelLabelString
: The string that labels the **Cancel** button. In Motif 1.2, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Cancel".

1. From Motif 2.0 onwards, use `XtNameToWidget`(): the various toolkit GetChild() routines are considered deprecated.

XmNdefaultButtonType
Specifies which PushButton provides the default action. Possible values:

XmDIALOG_CANCEL_BUTTON
XmDIALOG_OK_BUTTON
XmDIALOG_HELP_BUTTON

XmNdialogType
The type of MessageBox dialog, which also indicates the message symbol that displays by default. Possible values:

XmDIALOG_ERROR	XmDIALOG_TEMPLATE (1.2)
XmDIALOG_INFORMATION	XmDIALOG_WARNING
XmDIALOG_MESSAGE	XmDIALOG_WORKING
XmDIALOG_QUESTION	

XmNhelpLabelString
The string that labels the **Help** button. In Motif 1.2, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Help".

XmNmessageAlignment
The type of alignment for the message label. Possible values:

XmALIGNMENT_BEGINNING
XmALIGNMENT_CENTER
XmALIGNMENT_END

XmNmessageString
The string to use as the message label.

XmNminimizeButtons
If False (default), all buttons are standardized to be as wide as the widest button and as high as the highest button. If True, buttons will keep their preferred size.

XmNokLabelString
The string that labels the **OK** button. In Motif 1.2, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "OK".

XmNsymbolPixmap
The pixmap label to use as the message symbol.

Callback Resources

MessageBox defines the following callback resources:

Callback	Reason Constant
XmNcancelCallback	XmCR_CANCEL
XmNokCallback	XmCR_OK

XmNcancelCallback
List of callbacks that are called when the user selects the **Cancel** button.

XmNokCallback
List of callbacks that are called when the user selects the **OK** button.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
    XEvent      *event;         /* event structure that triggered callback */
} XmAnyCallbackStruct;
```

Inherited Resources

MessageBox inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of XmNborderWidth is reset to 0 by XmManager. BulletinBoard sets the value of XmNinitialFocus to XmNdefaultButton and resets the default XmNshadowThickness from 0 to 1 if the MessageBox is a child of a DialogShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNlabelFontList	XmBulletinBoard
XmNallowOverlap	XmBulletinBoard	XmNlabelRenderTable	XmBulletinBoard
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNautoUnmanage	XmBulletinBoard	XmNmapCallback	XmBulletinBoard
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNmarginHeight	XmBulletinBoard
XmNborderColor	Core	XmNmarginWidth	XmBulletinBoard
XmNborderPixmap	Core	XmNnavigationType	XmManager
XmNborderWidth	Core	XmNnoResize	XmBulletinBoard
XmNbottomShadowColor	XmManager	XmNnumChildren	Composite
XmNbottomShadowPixmap	XmManager	XmNpopupHandlerCallback	XmManager
XmNbuttonFontList	XmBulletinBoard	XmNresizePolicy	XmBulletinBoard
XmNbuttonRenderTable	XmBulletinBoard	XmNscreen	Core
XmNcancelButton	XmBulletinBoard	XmNsensitive	Core
XmNchildren	Composite	XmNshadowThickness	XmManager
XmNcolormap	Core	XmNshadowType	XmBulletinBoard
XmNdefaultButton	XmBulletinBoard	XmNstringDirection	XmManager
XmNdefaultPosition	XmBulletinBoard	XmNtextFontList	XmBulletinBoard

Resource	Inherited From	Resource	Inherited From
XmNdepth	Core	XmNtextRenderTable	XmBulletinBoard
XmNdestroyCallback	Core	XmNtextTranslations	XmBulletinBoard
XmNdialogStyle	XmBulletinBoard	XmNtopShadowColor	XmManager
XmNdialogTitle	XmBulletinBoard	XmNtopShadowPixmap	XmManager
XmNfocusCallback	XmBulletinBoard	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNunmapCallback	XmBulletinBoard
XmNhighlightColor	XmManager	XmNuserData	XmManager
XmNhighlightPixmap	XmManager	XmNwidth	Core
XmNinitialFocus	XmManager	XmNx	Core
XmNinitialResourcesPersistent	Core	XmNy	Core
XmNinsertPosition	Composite		

Translations

The translations for MessageBox include those from XmManager.

Additional Behavior

MessageBox has the following additional behavior:

MAny KCancel
: For a sensitive **Cancel** button, invokes the callbacks in XmNactivateCallback**.**

KActivate
: For the button that has keyboard focus, or the default button, invokes the callbacks in XmNactivateCallback.

<OK Button Activated>
: Invokes the callbacks for XmNokCallback.

<Cancel Button Activated>
: Invokes the callbacks for XmNcancelCallback.

<Help Button Activated>
: Invokes the callbacks for XmNhelpCallback.

<FocusIn>
: Invokes the callbacks for XmNfocusCallback.

<Map>

Invokes the callbacks for XmNmapCallback if the parent is a DialogShell.

<Unmap>

Invokes the callbacks for XmNunmapCallback if the parent is a DialogShell.

See Also

`XmCreateObject`(1), `XmMessageBoxGetChild`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmBulletinBoard`(2), `XmErrorDialog`(2), `XmInformationDialog`(2), `XmManager`(2), `XmQuestionDialog`(2), `XmTemplateDialog`(2), `XmWarningDialog`(2), `XmWorkingDialog`(2).

Name

XmMessageDialog –an unmanaged MessageBox as a child of DialogShell.

Synopsis

Public Header:
<Xm/MessageB.h>

Instantiation:
widget = XmCreateMessageDialog (parent, name,...)

Functions/Macros:
`XmCreateMessageDialog`(), `XmMessageBoxGetChild`()

Description

An XmMessageDialog is a compound object created by a call to `XmCreate-MessageDialog`() that an application can use to present a message to the user. A MessageDialog consists of a DialogShell with an unmanaged MessageBox widget as its child. The MessageBox resource XmNdialogType is set to XmDIALOG_MESSAGE. A MessageDialog includes four components: a symbol, a message, three buttons, and a separator between the message and the buttons. By default, there is no symbol. In Motif 1.2, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Cancel**, and **Help** by default.

Default Resource Values

A MessageDialog sets the following default values for MessageBox resources:

Name	Default
XmNdialogType	XmDIALOG_MESSAGE

Widget Hierarchy

When a MessageDialog is created with a specified name, the DialogShell is named name*_popup* and the MessageBox is called *name***.**

See Also

`XmCreateObject`(1), `XmMessageBoxGetChild`(1), `XmDialogShell`(2), `XmMessageBox`(2).

Name

XmMultiList – The Internationalized Extended List widget

Synopsis

Public Header:
<Xm/MultiList.h>

Class Name:
XmMultiList

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmMultiList

Class Pointer:
xmMultiListWidgetClass

Instantiation:

```
widget = XmCreateMultiList(parent, name,...)
or
widget = XtCreateWidget(name, xmMultiListWidget-
Class,...)
```

Functions/Macros:

```
XmCreateMultiList(), XmMultiListGetSelectedRows(),
XmMultiListUnselectAllItems(), XmMultiListUnselec-
tItem(), XmMultiListToggleTow(), XmMultiListSelec-
tItems(), XmMultiListDeselectItems(),
XmMultiListSelectAllItems(), XmMultiListSelectRow(),
XmMultiListDeselectRow(), XmMultiListGetSelectedRowAr-
ray(), XmMultiListMakeRowVisible().
```

Availability

OpenMotif 2.2 and later. Known as the Ext18List until OpenMotif 2.3. (Provisional Widget).

Description

This widget contains a multi-column list with headers along the top and a search area along the bottom. The list has scrollbars along the right and bottom edges that allow vertical and horizontal scrolling both by column and by pixel. The portion of the list data that is currently visible can be altered by scrollbar actions, widget resource setting and the redisplay of the list data after a string search has been successful. The sorting of elements within a particular column is also supported. To sort the list by the elements in a given column, select the column's title.

To search for a particular string in the list, type the string value to be searched for in the list's associated text field and then press the "Find" pushbutton. The search for the string begins in the currently selected row, after the location of the

previously searched for string, or at the first column and first row if there is no column selected. If the string is not found in that row, then the search continues through all rows after and then before, the currently selected row. If the string is found, the display of the list is adjusted to make the string visible. If the string was not found, or if the string is visible, the application will issue a warning beep. Pointer button one allows the user to select a row or a column for sorting. The callbacks on the doubleClickCallback list are called when the user double clicks pointer button one. If the list data can contain a row pixmap to display at the extreme left of the row.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a .Xdefaults file, remove the XmN or XmC prefix and use the remaining letters. To specify one of the defined values for a resource in a .Xdefaults file, remove the Xm prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using XtSetValues (S), retrieved by using XtGetValues (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNcolumnTitles	XmCColumnTitles	XmString*	NULL	CSG
XmNdoubleClickCallback	XmCCallback	XtCallbackList	NULL	C
XmNentryData	XmCEntryData	XtPointer	NULL	CSG
XmNfindLabel	XmCFindLabel	XmString	Find	CSG
XmNfirstColumn	XmCFirstLocation	short	O	CSG
XmNfirstColumnPixmaps	FirstColumnPixmaps	Boolean	False	CSG
XmNfirstRow	XmCFirstLocation	short	0	CSG
XmNfontList	XmCFontList	XmFontList	dynamic	CSG
XmNheight	XmCHeight	Dimension	300	CSG
XmNnumColumns	XmCNumColumns	short	0	CSG
XmNnumRows	XmCNumRows	short	0	CSG
XmNselectedColumn	XmCSelectedColumn	short	0	CSG
XmNselectionPolicy	XmCSelectionPolicy	unsigned char	XmEXTENDED_SELECT	CSG
XmNshowFind	XmCShowFind	boolean	True	CSG
XmNsingleSelectionCallback	XmCCallback	XtCallbackList	NULL	CSG
XmNsortFunctions	XmCFunction	Xm18SortFunction **	NULL	CSG

Name	Class	Type	Default	Access
XmNtitle	XmCTitle	XmString	NULL	CSG
Xmwidth	XmCWidth	Dimension	300	CSG

XmNcolumnTitles
: This is an array of length numColumns of strings displayed at the top of each column. The data is allocated and maintained by the client.

XmNdoubleClickCallback
: All routines in this list will be called whenever the user double clicks on a row in the list.

XmNentryData
: This resource is the data associated with each row in the list. The data is an array of Xm18RowInfo structures of length numRows allocated by the client. The data is allocated and maintained by the client. The Xm18RowInfo structure is defined below.

XmNfindLabel
: The label to be shown on the find button.

XmNfirstColumn
: This resource allows the client to adjust the current view of the list data to have a new top left column location. When setting this resource, firstRow should also be updated.

XmNfirstColumnPixmaps
: This resource specifies that the pixmap stored in the row info structure should be used instead of Xm18RowInfo values[0]. If pixmaps are present, the rows may be dragged by pressing on the pixmap with pointer button three. If this resource is True, then values[0] is never referenced. If False, then the Xm18RowInfo data pixmap is never referenced.

XmNfirstRow
: This resource allows the client to adjust the current view of the list data to have a new top left row location. When setting this resource, firstColumn should also be updated.

XmNfontList
: This is an OSF/Motif style font list. The first font in this list will be used to display all text in the Extended List widget. The Extended List widget currently supports only one font.

XmNheight
: This is the overall height value assigned to the Extended List widget. Modifying this resource will affect scrollbar size and location.

XmNnumColumns
XmNnumRows
: These resources specify the number of columns and rows the widget expects to display. These resources are used as the maximum indices for many of the other resources in this widget. Care should be taken when modifying these resources to ensure that the other values have also been modified.

XmNselectedColumn
: This is the index of the currently selected column. This also the column by which the list is being sorted.

XmNselectionPolicy
: Defines the interpretation of the select action. This resource can have the values XmSINGLE_SELECT or XmEXTENDED_SELECT. Other values result in undefined behavior.

XmNshowFind
: This boolean manages and unmanages the find button

XmNsingleSelectionCallback
: All routines in this list will be called whenever the user clicks on a line in the list. A pointer to the Xm18RowInfo structure corresponding to the line selected is passed as call_data. If in extended select mode the value of call_data is undefined.

XmNsortFunctions
: This is an array of functions, one for each column, called to determine the ordering of the rows in the column, similar to qsort.

XmNtitle
: This is the title that is displayed at the top of the Extended List widget. If this value is NULL, the title area will not be shown.

XmNwidth
: This is the overall width value assigned to the Extended List widget. Modifying this resource will affect scrollbar size and location.

Inherited Resources

Resource	Inherited from	Resource	Inherited from
XmNbottomShadowColor	XmManager	XmNaccelerators	Core
XmNbottomShadowPixmap	XmManager	XmNancestorSensitive	Core
XmNforeground	XmManager	XmNbackground	Core
XmNhelpCallback	XmManager	XmNbackgroundPixmap	Core
XmNhighlightColor	XmManager	XmNborderColor	Core
XmNhighlightPixmap	XmManager	XmNborderPixmap	Core

Resource	Inherited from	Resource	Inherited from
XmNinitialFocus	XmManager	XmNborderWidth	Core
XmNlayoutDirection	XmManager	XmNcolormap	Core
XmNnavigationType	XmManager	XmNdepth	Core
XmNpopupHandlerCallback	XmManager	XmNdestroyCallback	Core
XmNshadowThickness	XmManager	XmNheight	Core
XmNstringDirection	XmManager	XmNinitialResourcesPersistent	Core
XmNtopShadowColor	XmManager	XmNmappedWhenManaged	Core
XmNtopShadowPixmap	XmManager	XmNscreen	Core
XmNtraversalOn	XmManager	XmNsensitive	Core
XmNunitType	XmManager	XmNtranslations	Core
XmNuserData	XmManager	XmNwidth	Core
XmNchildren	Composite	XmNx	Core
XmNinsertPosition	Composite	XmNy	Core
XmNnumChildren	Composite		

Translations

The following are the default translation bindings used by the icon button:

~Ctrl ~Shift <Btn1Down>: ButtonDown()
Ctrl ~Shift <Btn1Down>: ButtonDown(Toggle)
~Ctrl Shift <Btn1Down>: ButtonDown(Extend)
Button1 <Motion>: Motion()
<Btn1Up>: ButtonUpOrLeave()

The following actions are supported by the Extended List:

ButtonDown(type)
: Processes a button press action that may begin with either a select or a double click. The type argument can be either Toggle or Extend. These values determine which mode of an extended select will be initiated on this button event. Consult the OSF/Motif Style Guide for details.

Motion() Processes motion events to allow the selection region to be modified when in extended selection mode. It is assumed that this action is called between a ButtonDown() and ButtonUpOrLeave() action.

ButtonUpOrLeave()
: Cleans up after ButtonDown() and Motion().

Callback Routines

All procedures on the Extended List's singleSelectionCallback and doubleClick-Callback lists will have a pointer to a Xm18RowInfo structure passed to them in the call_data field. This structure is defined above.
Note: if a single SelectionCallback is registered on an extended list in extended_select_mode, the value of call_data is undefined.

void (callback)(**Widget** *w*, **XtPointer** *client_data*, **XtPointer** *call_data*)

w	the Extended List widget
client_data	the client data specified by the application
call_data	a pointer to an Xm18RowInfo structure corresponding to the row selected

Sort Function

typedef **int** (Xm18SortFunction) (**short** *column*, **Xm18RowInfo *** *row1*, **Xm18RowInfo *** *row2*);

column	the column currently being sorted
row1, *row2*	the two rows being compared. The return value must be an integer less than, equal to, or greater than 0, depending on whether the first argument is less than, equal to, or greater than the second.

See Also

`XmCreateObject(1), Composite(2), Constraint(2), Core(2), XmList, XmManager(2).`

Name

XmNotebook widget class – a constraint widget which lays out its children like pages in a book

Synopsis

Public Header:
<Xm/Notebook.h>

Class Name:
XmNotebook

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmNotebook

Class Pointer:
xmNotebookWidgetClass

Instantiation:
widget = `XmCreateNotebook` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmNotebookWidgetClass,...)

Functions/Macros:
`XmNotebookGetPageInfo`(), `XmCreateNotebook`(), `XmIsNotebook`()

Availability

Motif 2.0 and later.

Description

Notebook is a constraint widget that organises its children into logical pages. It lays itself out so that the stack of logical pages look like the pages of an open notebook. The Notebook has visuals to simulate book binding and overlapping (back) page edges. Only one page is visible at any time.

The Notebook can be divided in sections and sub-sections by creating tabs which are displayed along the edge of the Notebook pages. A major tab divides the Notebook into sections, within each section there may be further minor tabs. A tab is associated with a page, so that activating a tab causes the relevant page to be displayed within the Notebook. The Notebook automatically creates a tab scroller, consisting of a set of four ArrowButtonGadgets. These are used for moving backwards and forwards between the major and minor tabs, when it is not possible to display all the tabs.

A status area can be associated with a page, and this is used for providing additional information, typically the page number.

In addition to associating tabs with pages, a page scroller can also be created for moving between pages of the Notebook. This can be any preferred method of navigating between the pages. If, however, there is no page scroller associated with the Notebook when it is realized, the Notebook creates a default page scroller, consisting of a SpinBox. The Notebook only requires one page scroller.

A fully programmed Notebook therefore consists of pages, tabs, status areas, a page scroller, and a tab scroller. The programmer does not create the tab scroller, and only optionally provides an alternative page scroller. Tabs and status areas are also optional.

An application adds a page to the Notebook by creating a child with the constraint resource XmNnotebookChildType set to XmPAGE. Any widget derived from RectObj may form a page of the Notebook. The default behavior is that if XmNnotebookChildType is unspecified (XmNONE), the child forms a page, with the following exceptions: a child with the XmQTactivatable trait (ArrowButton, DrawnButton, PushButton, and derived classes) is set up as a tab; a child with the XmQTaccessTextual trait (Label, Text, TextField, and derived classes) becomes a status area; a child with the XmQTnavigator trait (ScrollBar, SpinBox, and derivatives) is made into a page scroller.

Pages are ordered by setting the constraint resource XmNpageNumber appropriately for each child of type XmPAGE. A tab is attached to a page by adding an appropriate widget to the Notebook with XmNnotebookChildType set to either XmMAJOR_TAB or XmMINOR_TAB, and with the XmNpageNumber constraint resource set to that of the relevant page. Similarly, a status area can be attached to a page by adding the child with XmNnotebookChildType set to XmSTATUS_AREA, and again appropriately setting XmNpageNumber to the relevant page.

The resources XmNcurrentPageNumber, XmNfirstPageNumber, and XmNlastPageNumber control the range of pages which may be displayed. If a child has a XmNpageNumber constraint value which falls outside of the bounds set by XmNfirstPageNumber and XmNlastPageNumber, either the bounds must be altered to encompass the page constraint, or the page number itself must be altered, before the Notebook will display the child. The XmNlastPageNumber resource is dynamically maintained as the highest XmNpageNumber supplied, unless the application itself changes the value of XmNlastPageNumber. Once set by the application, the XmNlastPageNumber resource is no longer maintained by the Notebook, even if higher XmNpageNumber values are set.

When a child page is managed and no XmNpageNumber has been explicitly assigned, the Notebook automatically assigns a number. The assigned number is the smallest unused number which is not less than either the XmNfirstPageNumber or the previous automatically allocated number. This may exceed XmNlastPageNumber. If XmNfirstPageNumber exceeds XmNlastPageNumber, the behavior of the Notebook is undefined. The default value of the constraint XmNpageNumber is XmUNSPECIFIED_PAGE_NUMBER, which causes the Notebook to automatically assign a number.

Pages may be assigned duplicate XmNpageNumber values, in which case the Notebook displays the child which is most recently managed. There may be gaps in the assigned page numbering scheme, in which case the Notebook displays a blank background, unless the application dynamically provides feedback into the background whilst processing a XmNpageChangedCallback. It is possible to create a tab for an empty slot by assigning an XmNpageNumber constraint to a tab child, for which there is no corresponding XmPAGE child.

The ConstraintSetValues method of the Notebook sorts children into ascending logical page number order. Logical page numbers may be assigned in any order, provided that the programmer does not rely anywhere upon the particular order in which children are added.

Traits

Notebook holds the XmQTscrollFrame, XmQTtraversalControl, and XmQTspecifyUnhighlight traits, which are inherited by any derived classes, and uses the XmQTscrollFrame, XmQTactivatable, XmQTnavigator, XmQTjoinSide, and XmQTaccessTextual traits.

New Resources

Notebook defines the following resources:

Name	Class	Type	Default	Access
XmNbackPageBackground	XmCBackPageBackground	Pixel	dynamic	CSG
XmNbackPageForeground	XmCBackPageForeground	Pixel	dynamic	CSG
XmNbackPageNumber	XmCBackPageNumber	Cardinal	2	CSG
XmNbackPagePlacement	XmCBackPagePlacement	unsigned char	XmBOTTOM_RIGHT	CSG
XmNbackPageSize	XmCBackPageSize	Dimension	8	CSG
XmNbindingPixmap	XmCBindingPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNbindingType	XmCBindingType	unsigned char	XmSPIRAL	CSG
XmNbindingWidth	XmCBindingWidth	Dimension	25	CSG

Name	Class	Type	Default	Access
XmNcurrentPageNumber	XmCCurrentPageNumber	int	XmUNSPECIFIED_PAGE_NUMBER	CSG
XmNfirstPageNumber	XmCFirstPageNumber	int	1	CSG
XmNframeBackground	XmCFrameBackground	Pixel	dynamic	CSG
XmNframeShadowThickness	XmCShadowThickness	Dimension	dynamic	CSG
XmNinnerMarginHeight	XmCInnerMarginHeight	Dimension	0	CSG
XmNinnerMarginWidth	XmCInnerMarginWidth	Dimension	0	CSG
XmNlastPageNumber	XmCLastPageNumber	int	XmUNSPECIFIED_PAGE_NUMBER	CSG
XmNmajorTabSpacing	XmCMajorTabSpacing	Dimension	3	CSG
XmNminorTabSpacing	XmCMinorTabSpacing	Dimension	3	CSG
XmNorientation	XmCOrientation	unsigned char	XmHORIZONTAL	CSG

XmNbackPageBackground
Specifies the background color for drawing the back (overlapped) pages.

XmNbackPageForeground
Specifies the foreground color for drawing the back (overlapped) pages.

XmNbackPageNumber
Specifies the number of lines to draw to represent the back (overlapped) pages. The minimum value is 1, the maximum is half the XmNbackPageSize value.

XmNbackPagePlacement
Specifies where to draw the back (overlapped) pages. The default depends upon the XmNlayoutDirection of the Notebook parent widget, and the Notebook XmNorientation. Possible values:

```
XmBOTTOM_RIGHT    /* lines drawn to bottom and right */
XmBOTTOM_LEFT     /* lines drawn to bottom and left */
XmTOP_RIGHT       /* lines drawn to top and right   */
XmTOP_LEFT        /* lines drawn to top and left    */
```

XmNbackPageSize
Specifies the thickness, in pixels, of the back page rendering.

XmNbindingPixmap
Specifies the pixmap for drawing the binding. The value is only used if the XmNbindingType is XmPIXMAP or XmPIXMAP_OVERLAP_ONLY.

XmNbindingType
Specifies the type of binding. Possible values:

XmNONE	XmSOLID
XmSPIRAL	XmPIXMAP

XmPIXMAP_OVERLAP_ONLY

A value of XmNONE displays no binding.

The value XmSOLID draws a solid binding using the foreground color of the widget. The binding is contained within the area of a frame containing the pages of the Notebook, and curtailed by the value of XmNbindingWidth.

The value XmSPIRAL draws a spiral binding using the foreground color of the widget. The binding is contained within the area of a frame containing the pages of the Notebook, and curtailed by the value of XmNbindingWidth, and by an area outside the frame of the pages also bounded by the value of XmNbindingWidth.

The value XmPIXMAP draws a binding using the value of XmNbindingPixmap as a tile or stipple, depending upon the depth of the supplied pixmap. A pixmap of depth 1 is used as a stipple, and tiled otherwise. The foreground color of the Notebook is used when stippling. The size of the binding drawn is the larger of XmNbindingWidth and the width of the pixmap.

The value XmPIXMAP_OVERLAP_ONLY is similar to XmPIXMAP, except that the size of the binding is bounded only by the value of XmNbindingWidth, and not the width of the pixmap.

XmNbindingWidth
Specifies the width, in pixels, of the Notebook binding area.

XmNcurrentPageNumber
Specifies the page number of the currently visible page. Initially set to the value of XmNfirstPageNumber, the XmNcurrentPageNumber is constrained to be not be less than the XmNfirstPageNumber and not to exceed the XmNlastPageNumber.

XmNfirstPageNumber
Specifies the page number of the first logical page that may be displayed in the Notebook. Any child with an XmNpageNumber less than this value cannot be displayed until such time as either the XmNfirstPageNumber, or the XmNpageNumber of the given child itself, is suitably altered.

XmNframeBackground
Specifies the background color for the Notebook frame.

XmNframeShadowThickness
Specifies the shadow thickness around the Notebook frame. In Motif 2.0 the default is 2. In Motif 2.1 and later, the default depends upon the XmDisplay XmNenableThinThickness resource: if True, the default is 1, otherwise 2.

XmNinnerMarginHeight
Specifies the margin on the top and bottom sides of page, status area, and page scroller children.

XmNinnerMarginWidth
Specifies the margin on the left and right sides of page, status area, and page scroller children.

XmNlastPageNumber
Specifies the page number of the last logical page that may be displayed in the Notebook. Any child with an XmNpageNumber in excess of this value may not be displayed until such time as either the XmNlastPageNumber, or the XmNpageNumber of the given child itself, is suitably altered. The XmNlastPageNumber is automatically set by the Notebook as pages are added, unless the programmer directly changes the value: once modified, the Notebook no longer maintains the last page reference, and the programmer must continue to set the value as required.

XmNmajorTabSpacing
Specifies the spacing between major tabs. If XmNframeShadowThickness exceeds the value of the spacing, then the shadow thickness is used.

XmNminorTabSpacing
Specifies the spacing between minor tabs. If XmNframeShadowThickness exceeds the value of the spacing, then the shadow thickness is used.

XmNorientation
Specifies the orientation of the Notebook. Possible values:

```
XmHORIZONTAL        /* binding on left or right side */
XmVERTICAL          /* binding on top or bottom side */
```

New Constraint Resources

Notebook defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNnotebookChildType	XmCNotebookChildType	unsigned char	XmNONE	CG[a]
XmNpageNumber	XmCPageNumber	int	XmUNSPECIFIED_PAGE_NUMBER	CSG
XmNresizable	XmCResizable	Boolean	True	CSG

a.Erroneously listed as CSG in 2nd edition.

XmNnotebookChildType
Specifies the child type of the Notebook. Possible values:

```
XmPAGE              /* the child is a page         */
XmMAJOR_TAB         /* the child is a major tab    */
XmMINOR_TAB         /* the child is a minor page   */
XmSTATUS_AREA       /* the child is a status area  */
XmPAGE_SCROLLER     /* the child is a page scroller */
```

XmNpageNumber
Specifies a logical page number associated with a child of the Notebook. If unspecified, the Notebook assigns the next unallocated number when the child is managed. The assigned number is calculated to be not less than the Notebook XmNfirstPageNumber.

XmNresizable
Specifies whether any child resize request is processed by the Notebook.

Callback Resources

Notebook defines the following callback resources:

Callback	Reason Constant
XmNpageChangedCallback	XmCR_NONE XmCR_PAGE_SCROLLER_INCREMENT XmCR_PAGE_SCROLLER_DECREMENT XmCR_MAJOR_TAB XmCR_MINOR_TAB

XmNpageChangedCallback
List of callbacks called when the current logical page number is initialized or changed.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;              /* the reason that the callback was called    */
    XEvent      *event;              /* points to event structure                  */
                                     /* that triggered callback                    */
    int         page_number;         /* current page number                        */
    Widget      page_widget;         /* widget associated with current page number */
    int         prev_page_number;    /* previous current logical page number       */
    Widget      prev_page_widget;    /* widget associated with previous            */
                                     /* current logical page number                */
} XmNotebookCallbackStruct;
```

The structure members page_number, page_widget, prev_page_number, prev_page_widget are valid for any value of reason.

reason specifies the cause of callback invocation. At Notebook initialization, the page changed callback list is called in order to set up the first current page, and the *reason* structure member in this instance will have the value XmCR_NONE. Thereafter, if a tab child is activated, the *reason* member has the value XmCR_MAJOR_TAB or XmCR_MINOR_TAB, depending upon the XmNnotebookChildType resource of the selected tab. If the page scroller is activated, the

reason field is either XmCR_PAGE_SCROLLER_INCREMENT or XmCR_PAGE_SCROLLER_DECREMENT, depending upon the scroller action. The *reason* member is also XmCR_NONE if the XmNcurrentPageNumber resource is changed through `XtSetValues`().

page_number specifies the new logical page number.

page_widget specifies the widget which has the new logical page number. This is NULL if no page widget with an XmNpageNumber value equal to *page_number* is found.

prev_page_number specifies the current logical page number. At Notebook initialization, the value is XmUNSPECIFIED_PAGE_NUMBER.

prev_page_widget specifies the currently displayed page child of the Notebook. This is NULL at Notebook initialization, and where there is no page widget with an XmNpageNumber value equal to *prev_page_number*.

Inherited Resources

Notebook inherits the resources shown below. The resources are listed alphabetically, along with the superclass that defines them.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core

Resource	Inherited From	Resource	Inherited From
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Translations

The translations for Notebook include those of XmManager. In addition, Notebook places the following accelerators upon Tab children:

Event	Action
KBeginLine	TraverseTab(Home)
KEndLine	TraverseTab(End)
KLeft	TraverseTab(Previous)
KRight	TraverseTab(Next)
KUp	TraverseTab(Previous)
KDown	TraverseTab(Next)

Action Routines

Notebook defines the following action routines:

TraverseTab(type)

A generic action to move the focus between major and minor tabs in the Notebook. The action type may be one of Home, End, Previous, or Next.

See Also

`XmNotebookGetPageInfo`(1), `XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `RectObj`(2), `XmManager`(2), `XmTabStack`(2).

Name

XmOptionMenu – a type of RowColumn widget used as an option menu.

Synopsis

Public Header:
<Xm/RowColumn.h>

Class Name:
XmRowColumn

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmRowColumn

Class Pointer:
xmRowColumnWidgetClass

Instantiation:
widget = XmCreateOptionMenu (parent, name,...)

Functions/Macros:
`XmCreateOptionMenu`(), `XmCreateSimpleOptionMenu`(), `XmVaCreateSimpleOptionMenu`(), `XmIsRowColumn`(), `XmOption-ButtonGadget`(), `XmOptionLabelGadget`()

Description

An XmOptionMenu is an instance of a RowColumn widget that is used as a menu that allows a user to select one of several choices. An OptionMenu consists of a label, a selection area, and pulldown menu pane. When you create an OptionMenu, you must supply the pulldown menu pane via the XmNsubMenuId resource. The menu pane must already exist and it must be a child of the OptionMenu's parent. The label (a LabelGadget) and the selection area (a CascadeButtonGadget) are created by the OptionMenu. You can specify the label string with the XmNlabelString resource.

OptionMenu is a RowColumn widget whose XmNrowColumnType resource is set to XmMENU_OPTION. The XmNorientation resource defaults to XmHORIZONTAL, which means that the label is displayed to the left of the selection area. If the resource is set to XmVERTICAL, the label is placed above the selection area. The selection area posts the menu pane, as well as displays the label of the current selection. The XmNmenuPost resource is set to BSelect Press. The XmNmenuHistory resource can be used to specify which item in the pulldown menu is the current choice. The XmNmnemonic and XmNmnemonicCharSet resources can be set to specify a mnemonic for the OptionMenu.

An OptionMenu can be created using `XmCreateOptionMenu`(). In this case, the OptionMenu does not automatically create its submenu; it must be added by the application.

An OptionMenu can also be created by XmCreateSimpleOptionMenu(), which automatically creates the OptionMenu and its submenu with the specified children. This routine uses the RowColumn resources associated with the creation of simple menus. For an OptionMenu, the only types allowed in the XmNbuttonType resource are XmCASCADEBUTTON, XmPUSHBUTTON, XmSEPARATOR, and XmDOUBLE_SEPARATOR. The name of each button is button_*n*, where *n* is the number of the button, ranging from 0 to 1 less than the number of buttons in the submenu. The name of each separator is separator_*n*, where *n* is the number of the separator, ranging from 0 to 1 less than the number of separators in the submenu.

Default Resource Values

An OptionMenu sets the following default values for RowColumn resources:

Name	Default
XmNmenuPost	BSelect Press
XmNorientation	XmHORIZONTAL
XmNrowColumnType	XmMENU_OPTION

Widget Hierarchy

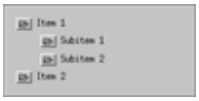
When an OptionMenu is created, the LabelGadget is named OptionLabel and the CascadeButtonGadget is named OptionButton.

See Also

`XmCreateObject`(1), `XmOptionButtonGadget`(1), `XmOptionLabelGadget`(1), `XmVaCreateSimpleOptionMenu`(1), `XmCascadeButtonGadget`(2), `XmLabelGadget`(2), `XmRowColumn`(2).

Name

XmOutline – The Outline widget class



Synopsis

Public Header:
<Xm/Outline.h>

Class Name:
XmOutline

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmHierarchy → XmOutline

Class Pointer:
xmOutlineWidgetClass

Instantiation:

```
widget = XmCreateOutline(parent, name,...)
or
widget = XtCreateWidget(name, xmOutlineWidget-
Class,...)
```

Functions/Macros:

```
XmCreateOutline()
```

Availability

OpenMotif 2.2 and later.

Description

The Outline widget is a container that shows the relationship of its children in an Outline format. Each child of the Outline widget is a node in the Outline. The hierarchy of nodes is created by specifying the “parent” of each node as a constraint resource. If a node’s parent is NULL, it is assumed to be a root of the Outline. Although each widget can only have one parent, the Outline widget supports adding more than one “root” node to a single Outline. Clicking on the associated control button hides the node’s child nodes.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining

letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNconnectNodes	XmCBoolean	Boolean	False	CSG
XmNconstrainWidth	XmCConstrain-Width	Boolean	False	CSG
XmNindentSpace	XmCIndentSpace	Dimension	30	CSG

XmNconnectNodes
: Specifies whether or not the child nodes of a node should be visually connected to it by lines drawn on the XmOutline's background.

XmNconstrainWidth
: Specifies that the Outline widget should negotiate with its children to offer them the best layout when the Outline itself cannot grow to display them at their preferred sizes.

XmNverticalindentSpace
: Specifies the number of pixels by which each new level of the Outline is indented.

Inherited Resources

Outline behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass.

Resource	Inherited from	Resource	Inherited from
XmNautoClose	XmHierarchy	XmNchildren	Composite
XmNcloseFolderPixmap	XmHierarchy	XmNinsertPosition	Composite
XmNhorizontalMargin	XmHierarchy	XmNnumChildren	Composite
XmNopenFolderPixmap	XmHierarchy	XmNaccelerators	Core
XmNfigureMode	XmHierarchy	XmNancestorSensitive	Core
XmNverticalMargin	XmHierarchy	XmNbackground	Core
XmNbottomShadowColor	XmManager	XmNbackgroundPixmap	Core
XmNbottomShadowPixmap	XmManager	XmNborderColor	Core

Resource	Inherited from	Resource	Inherited from
XmNforeground	XmManager	XmNborderPixmap	Core
XmNhelpCallback	XmManager	XmNborderWidth	Core
XmNhighlightColor	XmManager	XmNcolormap	Core
XmNhighlightPixmap	XmManager	XmNdepth	Core
XmNinitialFocus	XmManager	XmNdestroyCallback	Core
XmNlayoutDirection	XmManager	XmNheight	Core
XmNnavigationType	XmManager	XmNinitialResourcesPersistent	Core
XmNpopupHandlerCallback	XmManager	XmNmappedWhenManaged	Core
XmNshadowThickness	XmManager	XmNscreen	Core
XmNstringDirection	XmManager	XmNsensitive	Core
XmNtopShadowColor	XmManager	XmNtranslations	Core
XmNtopShadowPixmap	XmManager	XmNwidth	Core
XmNtraversalOn	XmManager	XmNx	Core
XmNunitType	XmManager	XmNy	Core
XmNuserData	XmManager		

See Also

`XmCreateObject(1), Composite(2), Constraint(2), XmContainer, Core(2), XmHierarchy, XmManager(2), XmBulletinBoard(2).`

Name

XmPanedWindow widget class – a constraint widget that tiles its children.

Synopsis

Public Header:
<Xm/PanedW.h>

Class Name:
XmPanedWindow

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmPanedWindow

Class Pointer:
xmPanedWindowWidgetClass

Instantiation:
widget = `XmCreatePanedWindow` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmPanedWindowWidgetClass,...)

Functions/Macros:
`XmCreatePanedWindow`(), `XmIsPanedWindow`()

Description

PanedWindow is a constraint widget that tiles its children. In Motif 1.1, the children are laid out vertically from top to bottom, in the order that they are added to the PanedWindow. In Motif 1.2, the position of each child is controlled by the XmNpositionIndex resource. A PanedWindow is as wide as its widest child and all children are made that width. Users can adjust the height of a pane using a sash that appears below the corresponding pane.

In Motif 2.0 and later, the PanedWindow may be oriented either vertically or horizontally. When the XmNorientation resource is set to XmHORIZONTAL, the PanedWindow is as tall as its tallest child, and all children are made that height. The sash in this orientation is used to control the width of the pane.

New Resources

PanedWindow defines the following resources:

Name	Class	Type	Default	Access
XmNmarginHeight	XmCMarginHeight	Dimension	3	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	3	CSG

Name	Class	Type	Default	Access
XmNorientation	XmCOrientation	unsigned char	XmVERTICAL	CSG
XmNrefigureMode	XmCBoolean	Boolean	True	CSG
XmNsashHeight	XmCSashHeight	Dimension	10	CSG
XmNsashIndent	XmCSashIndent	Position	-10	CSG
XmNsashShadowThickness	XmCShadowThickness	Dimension	dynamic	CSG
XmNsashWidth	XmCSashWidth	Dimension	10	CSG
XmNseparatorOn	XmCSeparatorOn	Boolean	True	CSG
XmNspacing	XmCSpacing	Dimension	8	CSG

XmNmarginHeight
: The spacing between a PanedWindow widget's top or bottom edge and any child widget.

XmNmarginWidth
: The spacing between a PanedWindow widget's right or left edge and any child widget.

XmNorientation
: In Motif 2.0 and later, the orientation of the PanedWindow. Possible values:

  XmHORIZONTAL
  XmVERTICAL

XmNrefigureMode
: If True (default), children are reset to their appropriate positions following a change in the PanedWindow widget.

XmNsashHeight

XmNsashWidth
: The height and width of the sash.

XmNsashIndent
: If the PanedWindow orientation is XmVERTICAL, the resource specifies the horizontal position of the sash along each pane. Positive values specify the indent from the left edge; negative values, from the right edge (assuming the default values of XmNstringDirection and XmNlayoutDirection). Similarly, in an XmHORIZONTAL PanedWindow, it specifies the vertical position of the sash, positive values being calculated from the top edge, negative values from the bottom. If the value is too large, the sash is placed flush with the edge of the PanedWindow.

XmNsashShadowThickness
: The thickness of shadows drawn on each sash. In Motif 2.0 and earlier, the default is 2. In Motif 2.1 and later, the default depends upon the XmDisplay XmNenableThinThickness resource: if True, the default is 1, otherwise 2.

XmNseparatorOn
If True, the widget places a Separator or SeparatorGadget between each pane.

XmNspacing
The distance between each child pane.

New Constraint Resources

PanedWindow defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNallowResize	XmCBoolean	Boolean	False	CSG
XmNpaneMaximum	XmCPaneMaximum	Dimension	1000	CSG
XmNpaneMinimum	XmCPaneMinimum	Dimension	1	CSG
XmNpositionIndex	XmCPositionIndex	short	XmLAST_POSITION	CSG
XmNskipAdjust	XmCBoolean	Boolean	False	CSG

XmNallowResize
If False (default), the PanedWindow widget always refuses resize requests from its children. If True, the PanedWindow widget tries to grant requests to change a child's height.

XmNpaneMaximum

XmNpaneMinimum
The values of a pane's maximum and minimum dimensions for resizing. You can prevent a sash from being drawn by setting these values to be equal.

XmNpositionIndex
In Motif 1.2, the position of the widget in the PanedWindow's list of children, not including sashes. A value of 0 indicates the beginning of the list, while XmLAST_POSITION places the child at the end of the list.

XmNskipAdjust
If False (default), the PanedWindow widget automatically resizes this pane child. If True, resizing is not automatic, and the PanedWindow may choose to skip the adjustment of this pane.

Inherited Resources

PanedWindow inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. PanedWindow sets the default value of XmNshadowThickness to 2. The default value of XmNborderWidth is reset to 0 by Manager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Widget Hierarchy

The sash children of a PanedWindow are created using a private widget class called XmSash, derived from XmPrimitive. Each instance of a sash has the name Sash. The appearance of the sash can be configured using these names in external resource files.

Translations

The translations for PanedWindow are inherited from Manager. Additional translations are defined for sashes within a PanedWindow widget:

Event	Action
BSelect Press	SashAction(Start)
BSelect Motion	SashAction(Move)
BSelect Release	SashAction(Commit)
BTransfer Press	SashAction(Start)
BTransfer Motion	SashAction(Move)

Event	Action
BTransfer Release	SashAction(Commit)
KHelp	Help()
KUp	SashAction(Key,DefaultIncr,Up)
MCtrl KUp	SashAction(Key,LargeIncr,Up)
KDown	SashAction(Key,DefaultIncr,Down)
MCtrl KDown	SashAction(Key,LargeIncr,Down)
KLeft	SashAction(Key,DefaultIncr,Left)
MCtrl KLeft	SashAction(Key,LargeIncr,Left)
KRight	SashAction(Key,DefaultIncr,Right)
MCtrl KRight	SashAction(Key,LargeIncr,Right)
KNextField	NextTabGroup()
KPrevField	PrevTabGroup()

Action Routines

PanedWindow defines the following action routines:

Help()

Invokes the list of callbacks specified by XmNhelpCallback. If the PanedWindow doesn't have any help callbacks, the Help() routine invokes those associated with the nearest ancestor that has them.

NextTabGroup()

Traverses to the next tab group. Normally a tab group consists of a pane and its sash.

PrevTabGroup()

Traverses to the previous tab group. Normally a tab group consists of a pane and its sash.

SashAction(action)

Controls the interactive placement of the sash using the mouse. action can have one of three values:

StartBegins the placement operation.
MoveCauses the sash to move as the mouse moves.
CommitEnds the placement operation.

SashAction(Key,increment,direction)

Controls the placement of the sash when it is moved using the keyboard. Increment is either DefaultIncr, which moves the sash's

position by one line or LargeIncr, which moves the sash's position by one viewing region. direction is either Up, Down, Left, or Right.

Additional Behavior

PanedWindow has the following additional behavior:

<FocusIn>
: Highlights the sash and gives it keyboard focus.

<FocusOut>
: Unhighlights the sash and removes its keyboard focus.

See Also

`XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmManager`(2), `XmPaned`.

Name

XmParseMapping data type –an opaque type representing an entry in a parse table

Synopsis

Public Header:
<Xm/Xm.h>

Functions/Macros:
`XmParseMappingCreate`(), `XmParseMappingFree`(), `XmParseMappingGetValues`(), `XmParseMappingSetValues`(),

Availability

Motif 2.0 and later.

Description

XmParseMapping is an opaque data type representing an entry in a parse table, which is used for table-driven parsing of strings and compound strings.

A parse mapping consists of a match pattern, and either a substitution pattern or parse procedure, which can be used by string manipulation functions in order to compare against and subsequently transform text.

An XmParseTable is simply an array of parse mappings. `XmParseMappingCreate`() creates a parse mapping, for subsequent use in a parse table, using a resource style parameter list. The parse table can be passed to `XmStringParseText`(), for example, in order to filter or modify an input string. In the parse process, a pointer is initialized to the head of some input text. The parse table is inspected from top to bottom, comparing the match pattern of each parse mapping entry with the value at the input pointer. Where a correspondence is found, the parse mapping is used to supply transformed output at that point in the parsing process. The input pointer is subsequently advanced, and the process is reiterated.

The implementation of XmParseMapping is that of a pseudo widget: although not a real widget, the object has resources and a resource style interface for setting and fetching values of the mapping, principally the match pattern, substitution pattern, and parse procedure. Resources of the object are set and fetches through the procedures `XmParseMappingGetValues`() and `XmParseMappingSetValues`() respectively.

New Resources

ParseMapping defines the following resources:

Name	Type	Default	Access
XmNclientData	XtPointer	NULL	CSG
XmNincludeStatus	XmIncludeStatus	XmINSERT[a]	CSG
XmNinvokeParseProc	XmParseProc	NULL	CSG
XmNpattern	XtPointer	NULL	CSG
XmNpatternType	XmTextType	XmCHARSET_TEXT	CSG
XmNsubstitute	XmString	NULL	CSG

a. Erroneously given as XmInsert in 2nd edition.

XmNclientData
: Specifies application data passed to the parse procedure associated with the XmNinvokeParseProc resource.

XmNincludeStatus
: Specifies the way in which the result of the mapping is constructed. Possible values:

```
XmINSERT      /* concatenate XmNsubstitute value to output      */
              /*     parsing is continued                       */
XmINVOKE      /* result determined by XmNinvokeParseProc        */
XmTERMINATE   /* concatenate XmNsubstitute value to output      */
              /*     parsing is terminated                      */
```

XmNinvokeParseProc
: Specifies a procedure for determining the result of the mapping. The procedure is only used if XmNincludeStatus is XmINVOKE. An XmParseProc routine places the result of the mapping into the address specified by its *str_include* parameter, and returns either XmINSERT or XmTERMINATE, depending upon whether parsing is to continue. A full description of the format of an XmParseProc is given below.

XmNpattern
: Specifies a pattern to be matched against the input being parsed. The pattern is a maximum of one character.

XmNpatternType
The type of the value specified as value for the resource XmNpattern. Possible values:

XmMULTIBYTE_TEXT
XmWIDECHAR_TEXT
XmCHARSET_TEXT

XmNsubstitute
Specifies a compound string to be added to the result of the parse process.

Procedures

The XmParseProc has the following format:

```
typedef XmIncludeStatus (*XmParseProc) (    XtPointer *,
                                            XtPointer,
                                            XmTextType,
                                            XmStringTag,
                                            XmParseMapping,
                                            int,
                                            XmString *,
                                            XtPointer)

    XtPointer       *in_out;        /* text being parsed                  */
    XtPointer       text_end;       /* pointer to end of the text         */
    XmTextType      type;           /* type of text                       */
    XmStringTag     locale_tag;     /* type to be used for the result     */
    XmParseMapping  entry;          /* parse mapping triggering callback  */
    int             pattern_length; /* number of bytes in input text      */
    XmString        *str_include;   /* returned result of the parse       */
    XtPointer       call_data;      /* application data                   */
```

in_out initially points to the current location within the text being parsed. The pointer can be changed in order to reset the location from which to continue parsing after the callback finishes.

text_end points to the end of the in_out string. A parse procedure can set the value of the element to indicate where the parse is to continue from after the mapping has been applied to the input.

type is the type of the text *in_out*, and the type of the *locale_tag* to be used in creating the return compound string. *type* is one of XmCHARSET_TEXT, XmMULTIBYTE_TEXT, or XmWIDECHAR_TEXT.

locale_tag specifies the tag to be used in creating the result. If NULL, the tag created depends upon the value of *type*. If *type* is XmCHARSET_TEXT, a charset string tag is created from the value XmSTRING_DEFAULT_CHARSET.

Otherwise, a locale string tag is created from the value _MOTIF_DEFAULT_LOCALE.

entry points to the XmParseMapping object which triggered the callback.

pattern_length is the number of bytes in the input text remaining at the address specified by *in_out*.

str_include is an address where the parse procedure supplies a compound string which is to be inserted into the result of the parse process.

call_data is a pointer to application data passed through by the string parsing functions which invoke the callback.

See Also

`XmParseMappingCreate`(1), `XmParseMappingFree`(1), `XmParseMappingGetValues`(1), `XmParseMappingSetValues`(1), `XmParseTableFree`(1), `XmStringParseText`(1), `XmStringUnparse`(1).

Name

XmPopupMenu –a type of RowColumn widget used as a popup menu pane.

Synopsis

Public Header:
<Xm/RowColumn.h>

Instantiation:
widget = XmCreatePopupMenu (parent, name,...)

Functions/Macros:
`XmCreatePopupMenu`(), `XmCreateSimplePopupMenu`(), `XmMenuPosition`(), `XmVaCreateSimplePopupMenu`()

Description

An XmPopupMenu is the first menu pane in a popup menu system. All other menu panes in the menu system are pulldown panes. A PopupMenu can contain Labels, Separators, PushButtons, ToggleButtons, CascadeButtons, or the corresponding Gadget equivalents.

A PopupMenu is a RowColumn widget whose XmNrowColumnType resource is set to XmMENU_POPUP. The XmNmenuAccelerator resource is set to KMenu and XmNmenuPost is set to BMenu Press. The XmNpopupEnabled resource controls whether or not keyboard accelerators and mnemonics are enabled for a PopupMenu. A PopupMenu needs to be the child of a MenuShell widget to function properly. Use `XmMenuPosition`() to place a PopupMenu.

A PopupMenu can be created using `XmCreatePopupMenu`(). In this case, the PopupMenu does not automatically contain any components; they are added by the application. The PopupMenu created by this routine is a compound object consisting of a MenuShell widget and a RowColumn child.

A PopupMenu can also be created by `XmCreateSimplePopupMenu`(), which automatically creates the PopupMenu with the specified children and makes it the child of a MenuShell. This routine uses the RowColumn resources associated with the creation of simple menus. For a PopupMenu, any type is allowed in the XmNbuttonType resource. The name of each button is button_*n*, where *n* is the number of the button, ranging from 0 to 1 less than the number of buttons in the menu. The name of each separator is separator_*n*, where *n* is the number of the separator, ranging from 0 to 1 less than the number of separators in the menu. The name of each title is label_*n*, where *n* is the number of the title, ranging from 0 to 1 less than the number of titles in the menu.

In Motif 2.0 and later, the support for automatic popup menus is extended. The Manager and Primitive classes contain XmNpopupHandlerCallback resources, and the RowColumn installs event handlers which simplifies the posting and

choice of a popup menu to display. These are installed if the XmNpopupEnabled resource is set to XmPOPUP_AUTOMATIC or XmPOPUP_AUTOMATIC_RECURSIVE.

Default Resource Values

A PopupMenu sets the following default values for RowColumn resources:

Name	Default
XmNmenuAccelerator	KMenu
XmNmenuPost	BMenu Press
XmNpopupEnabled	XmPOPUP_KEYBOARD
XmNrowColumnType	XmMENU_POPUP

Widget Hierarchy

When a PopupMenu is created with a specified name, the MenuShell is named popup_*name* and the RowColumn is called *name*.

See Also

`XmCreateObject`(1), `XmMenuPosition`(1), `XmVaCreateSimplePopupMenu`(1), `XmMenuShell`(2), `XmRowColumn`(2).

Name

XmPrintShell widget class – a Shell interfacing onto the Xp printing facilities

Synopsis

Public Headers:
<Xm/Print.h>

Class Name:
XmPrintShell

Class Hierarchy:
Core → Composite → Shell → WMShell → VendorShell → TopLevelShell → ApplicationShell → XmPrintShell

Class Pointer:
xmPrintShellWidgetClass

Instantiation:
widget = `XmPrintSetup` (...)
or
widget = `XtCreatePopupShell` (name, xmPrintShellWidgetClass,...)

Functions/Macros:
`XmPrintSetup`(), `XmPrintToFile`(), `XmPrintPopupPDM`(), `XmRedisplayWidget`(), `XmIsPrintShell`()

Availability

Motif 2.1 and later.

Description

PrintShell is a Shell widget which interfaces to the X11R6 X Print (Xp) extensions in order to print out a widget hierarchy. The X Printing Architecture reuses the code which renders the contents and visuals of a widget on the video screen in order to print the widget hierarchy. The technique involves creating instances of requisite widgets within the X Print Server by adding the hierarchy below a PrintShell, and configuring the widgets with suitable resources: the expose methods of the widgets perform the printing. The PrintShell itself is created through `XmPrintSetup`(), which returns an XmPrintShell widget after establishing a connection to an X Print Server. The PrintShell provides resources for configuring an XPrint connection and for specifying printer attributes. In addition, callback resources are available for handling any pagination. If no XPrint connection can be established, PrintShell behaves like an ApplicationShell, from which it is derived.

The Print model allows for either synchronous or asynchronous printing. `XmRedisplayWidget`() provides synchronous printing by forcing a widget to print itself directly. Asynchronous printing is performed through widget exposure as

a result of events generated by the X Print Server and dispatched to the PrintShell: the programmer calls `XpStartJob`() to initialize the printing process; XmNstartJobCallback, XmNpageSetupCallback, and XmNendJobCallback resources of the PrintShell specify callbacks to provide asynchronous notification at critical points in the printing task.

Whether printing synchronously or asynchronously, an application creates a widget hierarchy on the PrintShell suitable for the required output. What is meant by suitable in this context is application and widget specific: typically, a Text widget is created under the PrintShell, and resources are set to contain the data to print. It may be appropriate to turn off highlighting or blinking cursors in the Text, or to set the background white. This depends upon whether the intention is to print out just the content of the widget, or whether the programmer intends a screen shot which involves the widget visuals.

As a related topic, the procedure `XmPrintSetupPDM`() requests that a print dialog manager (PDM) is started. The status of the connection to the PDM is monitored by specifying a procedure for the XmNpdmNotificationCallback resource of the PrintShell. Lastly, an additional printing interface is available through the function `XmPrintToFile`(), which fetches the data on the Print Server, and sends this to a file.

New Resources

PrintShell defines the following resources:

Name	Class	Type	Default	Access
XmNdefaultPixmapResolution	XmCDefaultPixmapResolution	unsigned short	100	CSG
XmNmaxX	XmCMaxX	Dimension	dynamic	G
XmNmaxY	XmCMaxY	Dimension	dynamic	G
XmNminX	XmCMinX	Dimension	dynamic	G
XmNminY	XmCMinY	Dimension	dynamic	G

XmNdefaultPixmapResolution
: Indicates the resolution in dots per inch used for scaling image files read by descendants of the widget. In general, the resource is not used when reading tile pixmaps (XmNbackgroundPixmap, (XmNbottomShadowPixmap, or similar).

XmNmaxX

XmNmaxY

XmNminX

XmNminY

Specifies the image area of the page in the current print context. The PrintShell maintains the values to reflect change in resolution or other attributes.

Callback Resources

PrintShell defines the following callback resources:

Callback	Reason Constant
XmNendJobCallback	XmCR_END_JOB
XmNpageSetupCallback	XmCR_PAGE_SETUP
XmNpdmNotificationCallback	XmCR_PDM_NONE XmCR_PDM_UP XmCR_PDM_START_ERROR XmCR_PDM_START_VXAUTH XmCR_PDM_START_PXAUTH XmCR_PDM_OK XmCR_PDM_CANCEL XmCR_PDM_EXIT_ERROR
XmNstartJobCallback	XmCR_START_JOB

XmNendJobCallback
Specifies the list of callbacks called to control the end of rendering.

XmNpageSetupCallback
Specifies the list of callbacks called to control page layout.

XmNpdmNotificationCallback
Specifies the list of callbacks called in notification of the Print Dialog Manager (PDM) status.

XmNstartJobCallback
Specifies the list of callbacks called to control the start of rendering.

Callback Structure

Each callback is passed a pointer to the following structure:

```
typedef struct {
    int         reason;     /* reason that the callback is invoked      */
    XEvent      *event;     /* event structure that triggered callback */
    XPContext   context;    /* X Print Context                          */
    Boolean     last_page;  /* whether this is the last page            */
    XtPointer   detail;     /* PDM selection                            */
} XmPrintShellCallbackStruct;
```

reason specifies the cause of the callback invocation. The value is XmCR_START_JOB for any callback on the XmNstartJobCallback list, XmCR_END_JOB for XmNendJobCallback procedures, and XmCR_PAGE_SETUP for XmNpageSetupCallback routines. When a

XmNpdmNotificationCallback procedure is invoked, *reason* indicates the status of the PDM connection. Possible values:

```
XmCR_PDM_CANCEL          /* PDM exited with CANCEL status          */
XmCR_PDM_EXIT_ERROR      /* PDM exited with ERROR status           */
XmCR_PDM_NONE            /* No PDM available on the display        */
XmCR_PDM_OK              /* PDM exited with OK status              */
XmCR_PDM_START_ERROR     /* PDM cannot start                       */
XmCR_PDM_START_VXAUTH    /* PDM cannot connect to video display */
XmCR_PDM_START_PXAUTH    /* PDM cannot connect to print display */
XmCR_PDM_UP              /* PDM is running                         */
```

context is an opaque handle representing the connection to the Print Server.

last_page is only of relevance within an XmNpageSetupCallback procedure, and it is a flag whereby the application indicates that this is the last page of the printing task. The value is initially set to False by the toolkit, and the application callback should set the value True as required. Thereafter, the XmNpageSetupCallback procedures are invoked once more with the value initially True, by way of return notification, and subsequently the XmNendJobCallback procedures are called.

detail is only used by XmNpdmNotificationCallback procedures, and it contains an Atom representing the selection used in connecting to the PDM. The *reason* element is XmCR_PDM_NONE[1] when detail is set.

Inherited Resources

PrintShell inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of XmNborderWidth is reset to 0 by VendorShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNlayoutDirection	VendorShell
XmNallowShellResize	Shell	XmNmappedWhenManaged	Core
XmNancestorSensitive	Core	XmNmaxAspectX	WMShell
XmNargc	ApplicationShell	XmNmaxAspectY	WMShell
XmNargv	ApplicationShell	XmNmaxHeight	WMShell
XmNaudibleWarning	VendorShell	XmNmaxWIdth	WMShell
XmNbackground	Core	XmNminAspectX	WMShell
XmNbackgroundPixmap	Core	XmNminAspectY	WMShell

1.Erroneously given as XmPDM_NONE in 2nd edition.

Resource	Inherited From	Resource	Inherited From
XmNbaseHeight	WMShell	XmNmwmDecorations	VendorShell
XmNbaseWidth	WMShell	XmNmwmFunctions	VendorShell
XmNborderColor	Core	XmNmwmInputMode	VendorShell
XmNborderPixmap	Core	XmNmwmMenu	VendorShell
XmNborderWidth	Core	XmNnumChildren	Composite
XmNbuttonFontList	VendorShell	XmNoverrideRedirect	Shell
XmNbuttonRenderTable	VendorShell	XmNpopdownCalback	Shell
XmNchildren	Composite	XmNpopupCallback	Shell
XmNcolormap	Core	XmNpreeditType	VendorShell
XmNcreatePopupChildProc	Shell	XmNsaveUnder	Shell
XmNdefaultFontList	VendorShell	XmNscreen	Core
XmNdeleteResponse	VendorShell	XmNsensitive	Core
XmNdepth	Core	XmNshellUnitType	VendorShell
XmNdestroyCallback	Core	XmNtextFontList	VendorShell
XmNgeometry	Shell	XmNtextRenderTable	VendorShell
XmNheight	Core	XmNtitle	WMShell
XmNheightInc	WMShell	XmNtitleEncoding	WMShell
XmNiconic	TopLevelShell	XmNtoolTipEnable	VendorShell
XmNiconMask	WMShell	XmNtoolTipPostDuration	VendorShell
XmNiconName	TopLevelShell	XmNtoolTipString	VendorShell
XmNiconNameEncoding	TopLevelShell	XmNtransient	WMShell
XmNiconPixmap	WMShell	XmNtranslations	Core
XmNiconWindow	WMShell	XmNuseAsyncGeometry	VendorShell
XmNinitialResourcesPersistent	Core	XmNunitType	VendorShell
XmNinitialState	WMShell	XmNvisual	Shell
XmNinput	WMShell	XmNwaitForWm	WMShell
XmNinputMethod	VendorShell	XmNwidth	Core
XmNinputPolicy	VendorShell	XmNwidthInc	WMShell
XmNinsertPosition	Composite	XmNwindowGroup	WMShell
XmNkeyboardFocusPolicy	VendorShell	XmNwinGravity	WMShell
XmNlabelFontList	VendorShell	XmNwmTimeout	WMShell
XmNlabelRenderTable	VendorShell	XmNx	Core
XmNminHeight	WMShell	XmNy	Core
XmNminWidth	WMShell		

See Also

XmGetScaledPixmap(1), XmPrintPopupPDM(1), XmPrintToFile(1), XmPrintSetup(1), XmRedisplayWidget(1), ApplicationShell(2), Composite(2), Core(2), Shell(2), TopLevelShell(2), VendorShell(2), WMShell(2).

Name

XmPromptDialog –an unmanaged SelectionBox as a child of a Dialog Shell.

Synopsis

Public Header:
<Xm/SelectioB.h>

Instantiation:
widget = XmCreatePromptDialog(...)

Functions/Macros:
`XmCreatePromptDialog`(), `XmSelectionBoxGetChild`(), `XmIsSelectionBox`()

Description

An XmPromptDialog is a compound object created by a call to XmCreatePromptDialog() that an application can use to prompt the user for textual input. A PromptDialog consists of a DialogShell with an unmanaged SelectionBox widget as its child. The SelectionBox resource XmNdialogType is set to XmDIALOG_PROMPT.

A PromptDialog contains a message, a region for text input, and three managed buttons. A fourth button is created but not managed; you can manage it explicitly if necessary. In Motif 1.2 and later, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Apply**, **Cancel**, and **Help** by default. The **Apply** button is the unmanaged button.

Default Resource Values

A PromptDialog sets the following default values for SelectionBox resources:

Name	Default
XmNdialogType	XmDIALOG_PROMPT
XmNlistLabelString	NULL
XmNlistVisibleItemCount	0

Widget Hierarchy

When a PromptDialog is created with a specified name, the DialogShell is named name*_popup* and the SelectionBox is called *name***.**

See Also

`XmCreateObject`(1), `XmSelectionBoxGetChild`(1), `XmDialogShell`(2), `XmSelectionBox`(2).

Name

XmPulldownMenu – a type of RowColumn used as a pulldown menu pane.

Synopsis

Public Header:
<Xm/RowColumn.h>

Instantiation:
widget = XmCreatePulldownMenu (parent, name,...)

Functions/Macros:
`XmCreatePulldownMenu`(), `XmCreateSimplePulldownMenu`(), `XmVaCreateSimplePulldownMenu`()

Description

An XmPulldownMenu is a menu pane for all types of pulldown menu systems, including menus off of a menu bar, cascading submenus, and the menu associated with an option menu. A PulldownMenu is associated with a CascadeButton. A PulldownMenu can contain Separators, PushButtons, ToggleButtons, and CascadeButtons.

A PulldownMenu is a RowColumn widget whose XmNrowColumnType resource is set to XmMENU_PULLDOWN. A PulldownMenu needs to be the child of a MenuShell widget to function properly.

A PulldownMenu can be created using `XmCreatePulldownMenu`(). In this case, the PulldownMenu does not automatically contain any components; they are added by the application. The PulldownMenu created by this routine is a compound object consisting of a MenuShell widget and a RowColumn child.

A PulldownMenu can also be created by `XmCreateSimplePulldown-Menu`(), which automatically creates the PulldownMenu with the specified children and makes it the child of a MenuShell. This routine uses the RowColumn resources associated with the creation of simple menus. For a PulldownMenu, any type is allowed in the XmNbuttonType resource. The name of each button is button_*n*, where *n* is the number of the button, ranging from 0 to 1 less than the number of buttons in the menu. The name of each separator is separator_*n*, where *n* is the number of the separator, ranging from 0 to 1 less than the number of separators in the menu. The name of each title is label_*n*, where *n* is the number of the title, ranging from 0 to 1 less than the number of titles in the menu.

Default Resource Values

A PulldownMenu sets the following default values for RowColumn resources:

Name	Default
XmNrowColumnType	XmMENU_PULLDOWN[a]

a.Erroneously given as XmMENU_POPUP in 2nd edition.

Widget Hierarchy

When a PulldownMenu is created with a specified name, the MenuShell is named popup_*name* and the RowColumn is called *name*.

See Also

`XmCreateObject`(1), `XmVaCreateSimplePulldownMenu`(1), `XmCascadeButton`(2), `XmMenuShell`(2), `XmRowColumn`(2).

Name

XmPushButton widget class – a widget that starts an operation when it is pressed.

Synopsis

Public Header:
<Xm/PushB.h>

Class Name:
XmPushButton

Class Hierarchy:
Core → XmPrimitive → XmLabel → XmPushButton

Class Pointer:
xmPushButtonWidgetClass

Instantiation:
widget = `XmCreatePushButton` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmPushButtonWidgetClass,...)

Functions/Macros:
`XmCreatePushButton`(), `XmIsPushButton`()

Description

A PushButton is a widget which, when pressed by the user, issues a logical event to the application. A PushButton displays a text or pixmap label. It invokes an application callback when it is clicked on with the mouse. The shading of the PushButton changes to make it appear either pressed in when selected or raised when unselected.

Traits

PushButton holds the XmQTactivatable, XmQTtakesDefault, XmQT-careParentVisual and XmQTmenuSavvy traits, which are inherited by any derived classes, and uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

New Resources

PushButton defines the following resources:

Name	Class	Type	Default	Access
XmNarmColor	XmCArmColor	Pixel	dynamic	CSG
XmNarmPixmap	XmCArmPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG

Name	Class	Type	Default	Access
XmNdefaultButtonShadowThickness	XmCDefaultButtonShadowThickness	Dimension	dynamic	CSG
XmNfillOnArm	XmCFillOnArm	Boolean	True	CSG
XmNmultiClick	XmCMultiClick	unsigned char	dynamic	CSG
XmNshowAsDefault	XmCShowAsDefault	Dimension	0	CSG

XmNarmColor
The color with which the armed button is filled. For a color display, the default color is a shade between the bottom shadow color and the background color. For a monochrome display, the default is the foreground color, and label text is switched to the background color. This resource is in effect only when XmNfillOnArm is set to True.

XmNarmPixmap
The pixmap that identifies the button when it is armed (and when its XmNlabelType is XmPIXMAP or XmPIXPMAP_AND_STRING). For a PushButton in a menu, this resource is disabled.

XmNdefaultButtonShadowThickness
The width of the shadow used to indicate a default PushButton.

XmNfillOnArm
If True (default), the PushButton widget fills the button (when armed) with the color specified by XmNarmColor. If False, the PushButton widget only switches the top and bottom shadow colors. For a PushButton in a menu, this resource is disabled (and assumed to be False).

XmNmultiClick
A flag that determines whether successive button clicks are processed or ignored. Possible values:

```
XmMULTICLICK_DISCARD    /* ignore successive button clicks; */
                        /* default value in a menu system   */
XmMULTICLICK_KEEP       /* count successive button clicks;  */
                        /* default value when not in a menu */
```

XmNshowAsDefault
Indicates the default PushButton by displaying a shadow. (In a menu, this resource is disabled.) This resource works in different ways:

If the width of the shadow is already specified through the resource XmNdefaultButtonShadowThickness, then XmNshowAsDefault behaves like a Boolean: that is, with a value of 0, no shadow is displayed; with a value greater than 0, a shadow is displayed.

If the width of the shadow has not been specified through the resource XmNdefaultButtonShadowThickness (i.e., it has a value of 0), then XmNshowAsDefault performs double duty: that is, a value greater than 0 says to highlight the PushButton as the default button and to use this value as the thickness of the shadow.

Callback Resources

PushButton defines the following callback resources:

Callback	Reason Constant
XmNactivateCallback	XmCR_ACTIVATE
XmNarmCallback	XmCR_ARM
XmNdisarmCallback	XmCR_DISARM

XmNactivateCallback
List of callbacks that are called when BSelect is pressed and released inside the widget.

XmNarmCallback
List of callbacks that are called when BSelect is pressed while the pointer is inside the widget.

XmNdisarmCallback
List of callbacks that are called when BSelect is released after it has been pressed inside the widget.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
    XEvent      *event;         /* event structure that triggered callback */
    int         click_count;    /* number of multi-clicks                  */
} XmPushButtonCallbackStruct;
```

click_count is meaningful only for XmNactivateCallback. Furthermore, if the XmN-multiClick resource is set to XmMULTICLICK_KEEP, then XmN-activate--Callback is called for each click, and the value of *click_count* is the number of clicks that have occurred in the last sequence of multiple clicks. If the XmNmultiClick resource is set to XmMULTICLICK_DISCARD, then *click_count* always has a value of 1.

Inherited Resources

PushButton inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. PushButton sets the default values of XmNmarginBottom, XmNmarginLeft, XmNmarginRight, and XmN-

marginTop dynamically based on the value of XmNshowAsDefault. If XmNarmPixmap is specified but XmNlabelPixmap is not, the default value of XmNlabelPixmap is set to the value of XmNarmPixmap. The default value of XmNborderWidth is reset to 0 by Primitive. In Motif 2.0 and earlier, the default values of XmNhighlightThickness and XmNshadowThickness are reset to 2. In Motif 2.1, the default values depend upon the XmDisplay XmNenableThinThickness resource: if True the default is 1, otherwise 2.

Resource	Inherited From	Resource	Inherited From
XmNaccelerator	XmLabel	XmNlayoutDirection	XmPrimitive
XmNaccelerators	Core	XmNmappedWhenManaged	Core
XmNacceleratorText	XmLabel	XmNmarginBottom	XmLabel
XmNalignment	XmLabel	XmNmarginHeight	XmLabel
XmNancestorSensitive	Core	XmNmarginLeft	XmLabel
XmNbackground	Core	XmNmarginRight	XmLabel
XmNbackgroundPixmap	Core	XmNmarginTop	XmLabel
XmNborderColor	Core	XmNmarginWidth	XmLabel
XmNborderPixmap	Core	XmNmnemonicCharSet	XmLabel
XmNborderWidth	Core	XmNmnemonic	XmLabel
XmNbottomShadowColor	XmPrimitive	XmNnavigationType	XmPrimitive
XmNbottomShadowPixmap	XmPrimitive	XmNpopupHandlerCallback	XmPrimitive
XmNcolormap	Core	XmNrecomputeSize	XmLabel
XmNconvertCallback	XmPrimitive	XmNrenderTable	XmLabel
XmNdepth	Core	XmNscreen	Core
XmNdestroyCallback	Core	XmNsensitive	Core
XmNfontList	XmLabel	XmNshadowThickness	XmPrimitive
XmNforeground	XmPrimitive	XmNstringDirection	XmLabel
XmNheight	Core	XmNtoolTipString	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNtopShadowPixmap	XmPrimitive
XmNhighlightOnEnter	XmPrimitive	XmNtranslations	Core
XmNhighlightPixmap	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNhighlightThickness	XmPrimitive	XmNunitType	XmPrimitive
XmNinitialResourcesPersistent	Core	XmNuserData	XmPrimitive
XmNlabelInsensitivePixmap	XmLabel	XmNwidth	Core
XmNlabelPixmap	XmLabel	XmNx	Core
XmNlabelString	XmLabel	XmNy	Core

Resource	Inherited From	Resource	Inherited From
XmNlabelType	XmLabel		

Translations

For PushButtons outside a Menu System:

Event	Action
MCtrl BSelect Press	ButtonTakeFocus()
BSelect Press	Arm()
BSelect Click	Activate() Disarm()
BSelect Release	Activate() Disarm()
BSelect Press 2+	MultiArm()
BSelect Release 2+	MultiActivate() Disarm()
BTransfer Press	ProcessDrag()
KSelect	ArmAndActivate()
KHelp	Help()

For PushButtons in a Menu System:

Event	Action
MCtrl BSelect Press	MenuButtonTakeFocus()
BSelect Press	BtnDown()
BMenu Press	BtnDown()
BMenu Release	BtnUp()
KActivate	ArmAndActivate()
KSelect	ArmAndActivate()
MAny KCancel	MenuShellPopdownOne() (1.2)
MAny KCancel	MenuEscape (2.0)

Action Routines

PushButton defines the following action routines:

Activate()

Displays the PushButton as unarmed, and invokes the list of call-backs specified by XmNactivateCallback. The button's appearance

may depend on the values of the resources XmNfillOnArm and XmNlabelPixmap.

Arm()

Displays the PushButton as armed, and invokes the list of callbacks specified by XmNarmCallback. The button's appearance may depend on the values of the resources XmNarmColor and XmNarmPixmap.

ArmAndActivate()

When the PushButton is in a menu, this action unposts the menu hierarchy and invokes the callbacks specified by the resources XmNarmCallback, XmNactivateCallback, and finally, XmNdisarmCallback.

When the PushButton is not in a menu, this action displays the PushButton as armed (as determined by the values of the resources XmNarmColor and XmNarmPixmap) and (assuming the button is not yet armed) invokes the list of callbacks specified by XmNarmCallback. After this occurs, the action displays the PushButton as unarmed and invokes the callbacks specified in XmNactivateCallback and XmNdisarmCallback.

BtnDown()

Unposts any menus that were posted by the parent menu of the PushButton, changes from keyboard traversal to mouse traversal, displays the PushButton as armed, and (assuming the button is not yet armed) invokes the callbacks specified by XmNarmCallback.

BtnUp()

Unposts the menu hierarchy, activates the PushButton, and invokes first the callbacks specified by XmNactivateCallback and then those specified by XmNdisarmCallback.

ButtonTakeFocus()

In Motif 2.0 and later, moves the current keyboard focus to the PushButton, without activating the widget.

Disarm()

Invokes the callbacks specified by XmNdisarmCallback.

Help()

Unposts the menu hierarchy, restores the previous keyboard focus, and invokes the callbacks specified by the XmNhelpCallback resource.

MenuButtonTakeFocus()
: In Motif 2.0 and later, moves the current keyboard focus to the PushButton, without activating the widget.

MenuShellPopdownOne()
: In Motif 1.2 and earlier, unposts the current menu and (unless the menu is a pulldown submenu) restores keyboard focus to the tab group or widget that previously had it. In a top-level pulldown menu pane attached to a menu bar, this action routine also disarms the cascade button and the menu bar.

MenuEscape()
: In Motif 2.0 and later, invokes the popdownOne method of the MenuShell class, which unposts the current menu, restores keyboard focus, and ungrabs the pointer where necessary. Any containing menu row column is disarmed.

MultiActivate()
: Increments the click_count member of the XmPushButtonCallbackStruct, displays the PushButton as unarmed (as determined by the resources XmNfillOnArm and XmNlabelPixmap), and invokes first the callbacks specified by XmNactivateCallback and then those specified by XmNdisarmCallback. This action routine takes effect only when the XmNmultiClick resource is set to XmMULTICLICK_KEEP.

MultiArm()
: Displays the PushButton as armed (as determined by the resources XmNarmColor and XmNarmPixmap) and invokes the list of callbacks specified by XmNarmCallback. This action routine takes effect only when the XmNmultiClick resource is set to XmMULTICLICK_KEEP.

ProcessDrag()
: In Motif 1.2 and later, initiates a drag and drop operation using the label of the PushButton.

Additional Behavior

PushButton has the following additional behavior:

<EnterWindow>
: Displays the PushButton as armed.

<LeaveWindow>
: Displays the PushButton as unarmed.

See Also

XmCreateObject(1), Core(2), XmLabel(2), XmPrimitive(2).

Name

XmPushButtonGadget widget class –a gadget that starts an operation when it is pressed.

Synopsis

Public Header:
<Xm/PushBG.h>

Class Name:
XmPushButtonGadget

Class Hierarchy:
Object → RectObj → XmGadget → XmLabelGadget → XmPushButtonGadget

Class Pointer:
xmPushButtonGadgetClass

Instantiation:
widget = `XmCreatePushButtonGadget` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmPushButtonGadgetClass,...)

Functions/Macros:
`XmCreatePushButtonGadget`(), `XmIsPushButtonGadget`()

Description

PushButtonGadget is the gadget variant of PushButton.

PushButtonGadget's new resources, callback resources, and callback structure are the same as those for PushButton.

Traits

PushButtonGadget holds the XmQTactivatable, XmQTmenuSavvy, and XmQTtakesDefault traits, which are inherited by any derived classes, and uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

Inherited Resources

PushButtonGadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. PushButtonGadget sets the default values of XmNmarginBottom, XmNmarginLeft, XmNmarginRight, and XmNmarginTop dynamically based on the value of XmNshowAsDefault. If XmNarmPixmap is specified but XmNlabelPixmap is not, the default value of XmNlabelPixmap is set to the value of XmNarmPixmap. The default value of XmNborderWidth is reset to 0 by Gadget.

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	RectObj	XmNlayoutDirection	XmGadget
XmNbackground	XmGadget	XmNnavigationType	XmGadget
XmNbackgroundPixmap	XmGadget	XmNsensitive	RectObj
XmNbottomShadowColor	XmGadget	XmNshadowThickness	XmGadget
XmNbottomShadowPixmap	XmGadget	XmNtoolTipString	XmGadget
XmNborderWidth	RectObj	XmNtopShadowColor	XmGadget
XmNdestroyCallback	Object	XmNtopShadowPixmap	XmGadget
XmNforeground	XmGadget	XmNtraversalOn	XmGadget
XmNheight	RectObj	XmNunitType	XmGadget
XmNhelpCallback	XmGadget	XmNuserData	XmGadget
XmNhighlightColor	XmGadget	XmNwidth	RectObj
XmNhighlightOnEnter	XmGadget	XmNx	RectObj
XmNhighlightPixmap	XmGadget	XmNy	RectObj
XmNhighlightThickness	XmGadget		

Behavior

As a gadget subclass, PushButtonGadget has no translations associated with it. However, PushButtonGadget behavior corresponds to the action routines of the PushButton widget. See the PushButton action routines for more information.

For PushButtonGadgets outside of a menu system, the following translations are defined:

Event	Action
MCtrl BSelect Press	ButtonTakeFocus()
BSelect Press	Arm()
BSelect Click	Activate() Disarm()

Event	Action
BSelect Release	Activate() Disarm()
BSelect Press 2+	MultiArm()
BSelect Release 2+	MultiActivate() Disarm()
BTransfer Press	ProcessDrag()
KSelect	ArmAndActivate()
KHelp	Help()

For PushButtonGadgets inside a menu system:

Event	Action
MCtrl BSelect Press	MenuButtonTakeFocus()
BSelect Press	BtnDown()
BMenu Press	BtnDown()
BMenu Release	BtnUp()
KActivate	ArmAndActivate()
KSelect	ArmAndActivate()
MAny KCancel	MenuShellPopdownOne() (1.2)
MAny KCancel	MenuEscape (2.0)

PushButtonGadget has additional behavior associated with <Enter> and <Leave>, which draw the shadow in the armed or unarmed state, respectively.

See Also

`XmCreateObject`(1), `Object`(2), `RectObj`(2), `XmGadget`(2), `XmLabelGadget`(2), `XmPushButton`(2).

Name

XmQuestionDialog – an unmanaged MessageBox as a child of a DialogShell.

Synopsis

Public Header:
<Xm/MessageB.h>

Instantiation:
widget = `XmCreateQuestionDialog` (parent, name,...)

Functions/Macros:
`XmCreateQuestionDialog`(), `XmMessageBoxGetChild`()

Description

An XmQuestionDialog is a compound object created by a call to `XmCreate-QuestionDialog`() that an application can use to ask the user a question. A QuestionDialog consists of a DialogShell with a MessageBox widget as its child. The MessageBox resource XmNdialogType is set to XmDIALOG_QUESTION.

A QuestionDialog includes four components: a symbol, a message, three buttons, and a separator between the message and the buttons. By default, the symbol is a question mark. In Motif 1.2, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Cancel**, and **Help** by default.

Default Resource Values

A QuestionDialog sets the following default values for MessageBox resources:

Name	Default
XmNdialogType	XmDIALOG_QUESTION
XmNsymbolPixmap	xm_question

Widget Hierarchy

When a QuestionDialog is created with a specified name, the DialogShell is named name*_popup* and the MessageBox is called *name*.

See Also

`XmCreateObject`(1), `XmMessageBoxGetChild`(1), `XmDialogShell`(2), `XmMessageBox`(2).

Name

XmRadioBox –a RowColumn that contains ToggleButtons.

Synopsis

Public Header:
<Xm/RowColumn.h>

Class Name:
XmRowColumn

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmRowColumn

Class Pointer:
xmRowColumnWidgetClass

Instantiation:
widget = `XmCreateRadioBox` (parent, name,...)

Functions/Macros:
`XmCreateRadioBox`(), `XmCreateSimpleRadioBox`(), `XmVaCreateSimpleRadioBox`(), `XmIsRowColumn`()

Description

An XmRadioBox is an instance of a RowColumn widget that contains ToggleButtonGadgets, only one of which can be selected at a given time. When a RadioBox is created with `XmCreateRadioBox()`: it does not automatically contain ToggleButtonGadget children; they are added by the application developer. A RadioBox can also be created by a call to `XmCreateSimpleRadioBox`(), which automatically creates the specified ToggleButtonGadget widgets as children.

A RadioBox is a RowColumn widget with its XmNrowColumnType resource set to XmWORK_AREA, XmNpacking set to XmPACK_COLUMN, and XmNradioAlwaysOne set to True, which means that one button is always selected. The XmNradioBehavior resource is set to True. The XmNmenuHistory resource indicates the last ToggleButtonGadget that was selected. The XmNisHomogenous resource is set to True and XmNentryClass is set to ToggleButtonGadget, to ensure that only ToggleButtonGadgets are added as children. RadioBox sets XmNvisibleWhenOff to True and XmNindicatorType to XmONE_OF_MANY for all of its ToggleButtonGadget children.

A RadioBox can be created by making a RowColumn with these resource values. When it is created in this way, a RadioBox does not automatically contain ToggleButtonGadget children; they are added by the application.

A RadioBox can also be created by a call to XmCreateSimpleRadioBox() or XmVaCreateSimpleRadioBox(). These routines automatically create the

RadioBox with ToggleButtonGadgets as children. The routines use the RowColumn resources associated with the creation of simple menus. For a RadioBox, the only type allowed in the XmNbuttonType resource is XmRADIOBUTTON. The name of each ToggleButtonGadget is button_*n*, where *n* is the number of the button, ranging from 0 to 1 less than the number of buttons in the RadioBox.

Default Resource Values

A RadioBox sets the following default values for its resources:

Name	Default
XmNentryClass	xmToggleButtonGadgetClass[a]
XmNisHomogenous	True
XmNnavigationType	XmTAB_GROUP
XmNradioAlwaysOne	True
XmNradioBehavior	True
XmNrowColumnType	XmWORK_AREA
XmNtraversalOn	True

a. Erroneously given as xmToggleButtonWidgetClass in 1st and 2nd editions

See Also

`XmCreateObject`(1), `XmVaCreateSimpleRadioBox`(1), `XmRowColumn`(2), `XmToggleButtonGadget`(2).

Name

XmRendition data type –an opaque type representing an entry in a render table

Synopsis

Public Header:
<Xm/Xm.h>

Instantiation:
rendition = XmRenditionCreate (....)

Functions/Macros:
`XmRenditionCreate`(), `XmRenditionFree`(), `XmRenditionRetrieve`(),
`XmRenditionUpdate`().

Availability

Motif 2.0 and later.

Description

XmRendition is an opaque data type used for rendering XmStrings.

A rendition consists of two parts: an XmStringTag, which is matched against tag elements within a compound string to be rendered, and rendering data such as color, font, line style.

The implementation of XmRendition is through a pseudo widget: although not a true widget, the object has resources and a resource style interface for setting and fetching values of the rendition. The object is created by `XmRenditionCreate`(), resources are fetched using `XmRenditionRetrieve`(), and set through `XmRenditionUpdate`(), and finally deallocated by `XmRenditionFree`(). Typically, a rendition forms an entry within a render table, and a rendition is merged into an existing render table, or used as the basis for a new table, through the function `XmRenderTableAddRenditions`(). Compound strings are rendered by successively matching tags within the compound string with tags associated with entries in a render table. Where there is an equivalence, that rendition is used to display the corresponding component of the compound string.

Resources within a rendition may have the value XmAS_IS, either explicitly assigned or as an implicit default. Where the value is XmAS_IS, the value is taken from the previous rendition used to render the compound string. If the previous rendition also has the value XmAS_IS, then the rendition before the previous one is inspected, and so on. A default value is provided if no previous renditions supply a resource value.

Renditions, and the render tables to which they belong, are sharply across widgets, and are reference counted. In the general case, widgets inherit a render table from the nearest ancestor which holds the XmQTspecifyRenderTable trait if their own render table is unspecified. It is important to deallocate a rendition through `XmRenditionFree`(), rather than directly invoking `XtFree`(), in order to maintain the reference count.

New Resources

XmRendition defines the following resources:

Name	Class	Type	Default	Access
XmNfont	XmCFont	XtPointer	XmAS_IS	CSG
XmNfontName	XmCFontName	String	XmAS_IS	CSG
XmNfontType	XmCFontType	XmFontType	XmAS_IS XmFONT_IS_XFT	CSG
XmNloadModel	XmCLoadModel	unsigned char	XmAS_IS	CSG
XmNrenditionBackground	XmCRenditionBackground	Pixel	XmUNSPECIFIED_PIXEL	CSG
XmNrenditionForeground	XmCRenditionForeground	Pixel	XmUNSPECIFIED_PIXEL	CSG
XmNstrikethruType	XmCStrikethruType	unsigned char	XmAS_IS	CSG
XmNtabList	XmCTabList	XmTabList	XmAS_IS	CSG
XmNtag	XmCTag	XmStringTag	""	C
XmNunderlineType	XmCUnderlineType	unsigned char	XmAS_IS	CSG

XmNfont

Specifies the font for the rendition. If specified, XmNloadModel is forced to XmLOAD_IMMEDIATE. Otherwise, the rendition automatically sets the value of the resource when the font associated with XmNfontName is loaded, either because the rendition is used to render a compound string, or as a side effect of a XmRenditionUpdate() call.

XmNfontName

Specifies either the name of a font, or a comma-separated list of font names (a font set). Each name is assumed to be in standard X Logical Font Description (XLFD) format. If both XmNfontName and XmNfont are specified for a rendition, the XmNfont resource takes precedence.

XmNfontType

Specifies whether the font associated with the rendition is a font or a fontset. Possible values:

XmFONT_IS_FONT XmFONT_IS_FONTSET

XmNloadModel
Specifies whether the font or fontset indicated by the XmNfontName resource is loaded when the rendition is created, or when first required. Possible values:

```
XmLOAD_IMMEDIATE        /* load on rendition create   */
XmLOAD_DEFERRED         /* load when font is required */
```

XmNrenditionBackground
Specifies the background color for the rendition.

XmNrenditionForeground
Specifies the foreground color for the rendition.

XmNstrikethruType
Specifies a line type for striking through a text segment. Possible values:

```
XmDOUBLE_DASHED_LINE    XmDOUBLE_LINE
XmSINGLE_DASHED_LINE    XmSINGLE_LINE
XmNO_LINE
```

XmNtabList
Specifies the tab list which specifies how compound strings containing tab components are laid out in columns.

XmNtag
Specifies a tag which is used in matching against XmStringTag components within compound strings, or in matching against other renditions. The value is taken from the tag parameter of the XmRenditionCreate() procedure which creates the rendition object. The value NULL is not valid, although the empty string "" is.

XmNunderlineType
Specifies a line type for underlining a text segment. Possible values:

```
XmDOUBLE_DASHED_LINE    XmDOUBLE_LINE
XmNO_LINE               XmSINGLE_DASHED_LINE
XmSINGLE_LINE
```

See Also

`XmRenditionCreate`(1), `XmRenditionFree`(1), `XmRenditionRetrieve`(1), `XmRenditionUpdate`(1), `XmRenderTableAddRenditions`(1), `XmRenderTableCopy`(1), `XmRenderTableFree`(1), `XmRenderTableGetRendition`(1), `XmRenderTableGetRenditions`(1), `XmRenderTableGetTags`(1), `XmRenderTableRemoveRenditions`(1).

Name

XmRowColumn widget class -a manager widget that arranges its children in rows and columns.

Synopsis

Public Header:
<Xm/RowColumn.h>

Class Name:
XmRowColumn

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmRowColumn

Class Pointer:
xmRowColumnWidgetClass

Instantiation:
widget = `XmCreateRowColumn` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmRowColumnWidgetClass,...)

Functions/Macros:
`XmCreateMenuBar`(), `XmCreateOptionMenu`(), `XmCreatePopupMenu`(), `XmCreatePulldownMenu`(), `XmCreateRadioBox`(), `XmCreateRowColumn`(), `XmCreateSimpleCheckBox`(), `XmCreateSimpleMenuBar`(), `XmCreateSimpleOptionMenu`(), `XmCreateSimplePopupMenu`(), `XmCreateSimplePulldownMenu`(), `XmCreateSimpleRadioBox`(), `XmCreateWorkArea`(), `XmIsRowColumn`(), `XmVaCreateSimpleCheckBox`(), `XmVaCreateSimpleMenuBar`(), `XmVaCreateSimpleOptionMenu`(), `XmVaCreateSimplePopupMenu`(), `XmVaCreateSimplePulldownMenu`(), `XmVaCreateSimpleRadioBox`()

Description

RowColumn provides an area in which children belonging to any widget type are displayed in rows and columns. RowColumn is a general-purpose manager widget class that can be configured into many layouts, such as a MenuBar, PopupMenu, PulldownMenu, OptionMenu, CheckBox, or RadioBox. Many of RowColumn's resources pertain only to a specific layout type.

In Motif 1.2 and later, a RowColumn that is configured as a PopupMenu or a PulldownMenu supports tear off menus. When a menu is torn off, it remains on the screen after a selection is made so that additional selections can be made. A

menu pane that can be torn off contains a tear-off button at the top of the menu. A tear-off button is a button that has a Separator-like appearance. The name of a tear-off button in a menu pane is TearOffControl. An application can set the following resources for a tear-off button: XmNbackground, XmNbackgroundPixmap, XmNbottomShadowColor, XmNforeground, XmNheight, XmNmargin, XmNseparatorType, XmNshadowThickness, and XmNtopShadowColor.

In Motif 2.0 and later, the mechanisms whereby pulldown menus are selected and posted have been rationalized. The Manager and Primitive classes support XmNpopupHandlerCallback resources which can be used to choose a popup menu to display in a given context. In addition, the RowColumn provides automatic posting of popups through extensions to the XmNpopupEnabled resource.

Traits

RowColumn holds the XmQTmenuSystem trait, which is inherited by any derived class, and uses the XmQTmenuSystem and XmQTmenuSavvy traits.

New Resources

RowColumn defines the following resources:

Name	Class	Type	Default	Access
XmNadjustLast	XmCAdjustLast	Boolean	True	CSG
XmNadjustMargin	XmCAdjustMargin	Boolean	True	CSG
XmNentryAlignment	XmCAlignment	unsigned char	XmALIGNMENT_BEGINNING	CSG
XmNentryBorder	XmCEntryBorder	Dimension	0	CSG
XmNentryClass	XmCEntryClass	WidgetClass	dynamic	CSG
XmNentryVerticalAlignment	XmCVerticalAlignment	unsigned char	XmALIGNMENT_CENTER	CSG
XmNisAligned	XmCIsAligned	Boolean	True	CSG
XmNisHomogeneous	XmCIsHomogenous	Boolean	dynamic	CSG
XmNlabelString	XmCXmString	XmString	NULL	C
XmNmarginHeight	XmCMarginHeight	Dimension	dynamic	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	dynamic	CSG
XmNmenuAccelerator	XmCAccelerators	String	dynamic	CSG
XmNmenuHelpWidget	XmCMenuWidget	Widget	NULL	CSG
XmNmenuHistory	XmCMenuWidget	Widget	NULL	CSG
XmNmenuPost	XmCMenuPost	String	NULL	CSG
XmNmnemonic	XmCMnemonic	KeySym	NULL	CSG
XmNmnemonicCharSet	XmCMnemonicCharSet	String	XmFONTLIST_DEFAULT_TAG	CSG

Name	Class	Type	Default	Access
XmNnumColumns	XmCNumColumns	short	1	CSG
XmNorientation	XmCOrientation	unsigned char	dynamic	CSG
XmNpacking	XmCPacking	unsigned char	dynamic	CSG
XmNpopupEnabled	XmCPopupEnabled	XtEnum	XmPOPUP_KEYBOARD	CSG
XmNradioAlwaysOne	XmCRadioAlwaysOne	Boolean	True	CSG
XmNradioBehavior	XmCRadioBehavior	Boolean	False	CSG
XmNresizeHeight	XmCResizeHeight	Boolean	True	CSG
XmNresizeWidth	XmCResizeWidth	Boolean	True	CSG
XmNrowColumnType	XmCRowColumnType	unsigned char	XmWORK_AREA	CG
XmNspacing	XmCSpacing	Dimension	dynamic	CSG
XmNsubMenuId	XmCMenuWidget	Widget	NULL	CSG
XmNtearOffModel	XmCTearOffModel	unsigned char	XmTEAR_OFF_DISABLED	CSG
XmNtearOffTitle	XmCTearOffTitle	XmString	NULL	CSG
XmNwhichButton	XmCWhichButton	unsigned int	dynamic	CSG

XmNadjustLast
: If True (default), the last row (or column) in the RowColumn widget is expanded so as to be flush with the edge.

XmNadjustMargin
: If True (default), text in each row (or column) will align with other text in its row (or column). This is done by forcing the margin resources (defined by the Label widget) to have the same value. For example, in a horizontally-oriented RowColumn widget, all items will have the same value for XmNmarginTop and XmNmarginBottom; in a vertically-oriented RowColumn widget, all items will have the same value for XmNmarginLeft and XmNmarginRight.

XmNentryAlignment
: When XmNisAligned is True, this resource tells RowColumn children how to align. The children must be subclasses of XmLabel or XmLabelGadget. If XmNrowColumnType is XmMENU_OPTION, the resource is forced to XmALIGNMENT_CENTER and cannot be changed. Possible values:

    XmALIGNMENT_BEGINNING
    XmALIGNMENT_CENTER
    XmALIGNMENT_END

XmNentryBorder
: The border width of a RowColumn widget's children.

XmNentryClass
The widget (or gadget) class to which children must belong when being added to a RowColumn widget. This resource is used only when the XmNisHomogeneous resource is set to True. XmNentryClass ensures that a MenuBar will have only cascade button children and that a RadioBox will have only toggle button children (or gadget variants of each class). XmNentryClass can have one of two default values. For a MenuBar, the default value is xmCascadeButtonWidgetClass. For a RadioBox, the default value is xmToggleButtonGadgetClass. Possible values:

```
xmToggleButtonGadgetClass     /* XmWORK_AREA with                */
                              /*      XmNradioBehavior True      */
xmCascadeButtonWidgetClass    /* XmMENU_BAR                      */
```

XmNentryVerticalAlignment
In Motif 1.2 and later, specifies how children that are subclasses of Label, Text, and TextField are aligned vertically. The resource has no effect if XmNorientation is XmVERTICAL or XmNpacking is XmPACK_TIGHT. Possible values:

XmALIGNMENT_BASELINE_BOTTOM
XmALIGNMENT_BASELINE_TOP
XmALIGNMENT_CONTENTS_BOTTOM
XmALIGNMENT_CENTER
XmALIGNMENT_CONTENTS_TOP

XmNisAligned
If True, enable the alignment specified in the XmNentryAlignment resource. Alignment is ignored in a label whose parent is a popup or pulldown MenuPane (for example, in an OptionMenu).

XmNisHomogeneous
If True, enforce the condition that all RowColumn children belong to the same class (the class specified by the XmNentryClass resource). When creating a RadioBox or a MenuBar, the default value of this resource is True; otherwise, it's False.

XmNlabelString
A label used only in option menus. A text string displays next to the selection area. By default, there is no label.

XmNmarginHeight

XmNmarginWidth
The spacing between an edge of the RowColumn widget and its nearest child. In popup and pulldown menus, the default is 0; in other types of RowColumn widgets, the default is 3 pixels.

XmNmenuAccelerator
: A pointer to a string that specifies an accelerator (keyboard shortcut) for use only in RowColumn widgets of type XmMENU_POPUP or XmMENU_BAR. In a popup menu, typing the accelerator posts the menu; in a menu bar, typing the accelerator highlights the first item and enables traversal in the menu bar. The string's format is like that of a translation but allows only a single key press event to be specified. The default value of this resource is KMenu (for popup menus) and KMenuBar (for menu bars).

XmNmenuHelpWidget
: The widget ID of the CascadeButton widget that serves as the Help button. This resource is meaningful only in RowColumn widgets of type XmMENU_BAR.

XmNmenuHistory
: The widget ID of the most recently activated menu entry. Since the most recently activated menu entry is also the choice that displays in an OptionMenu, this resource is useful for indicating the current selection in a RowColumn widget of type XmMENU_OPTION. In a RowColumn widget whose XmNradioBehavior resource is set to True, the XmNmenuHistory resource indicates the last toggle button to change from unselected to selected.

XmNmenuPost
: The string that describes the event for posting a menu. The value specifies an X event using translation table syntax. The default value depends on the type of RowColumn widget: for XmMENU_POPUP, the default is <Btn3Down>; for XmMENU_OPTION, XmMENU_BAR, and XmWORK_AREA, the default is <Btn1Down>; for XmMENU_PULLDOWN, this resource isn't meaningful.

XmNmnemonic
: The keysym of the key to press (in combination with the MAlt modifier) in order to post the pulldown menu associated with an option menu. This resource is meaningful only in option menus. In the label string, the first character matching this keysym will be underlined.

XmNmnemonicCharSet
: The character set for the option menu's mnemonic. The default value depends on the current language environment.

XmNnumColumns
: The number of columns (in a vertically-oriented RowColumn widget) or the number of rows (in a horizontally-oriented RowColumn widget). This resource is meaningful only when the XmNpacking resource is set to XmPACK_COLUMN.

XmNorientation
: The direction for laying out the rows and columns of children of a RowColumn widget. For all RowColumn widgets except a MenuBar, the default value is XmVERTICAL. Possible values:

```
XmVERTICAL      /* top-to-bottom creation      */
XmHORIZONTAL    /* left-to-right creation      */
```

XmNpacking

The method of spacing the items placed within a RowColumn widget. The default value is XmPACK_COLUMN for a RadioBox, and XmPACK_TIGHT for other types of RowColumn widget. Possible values:

```
XmPACK_TIGHT    /* give each box minimum sizing    */
XmPACK_COLUMN   /* pad boxes to align if needed    */
XmPACK_NONE     /* widget accommodates placement   */
```

XmNpopupEnabled

If True (default), keyboard shortcuts are in effect for popup menus. Set this resource to False if you want to disable accelerators and mnemonics in popup menus.

In Motif 2.0 and later, the resource changes type from Boolean to XtEnum in order to support the range of values required for the enhanced automatic popup mechanisms. The value XmPOPUP_DISABLED is equivalent to the behaviour described above for False, and XmPOPUP_KEYBOARD is equivalent to True. In addition, XmPOPUP_AUTOMATIC adds event handlers for automatic menu popup, and enables the keyboard for the menu. XmPOPUP_AUTOMATIC_RECURSIVE is similar, except that the search for a popup menu in a given context is not restricted to immediate children, and the most specific popup to display may be found in the parent of a target widget.

The new enumeration erroneously has the generic representation type XmREnum rather than a putative XmRPopupEnabled, and an enumeration for the values has not been installed within the standard representation types. Hence as of Motif 2.0, the type cannot be properly specified in a resource file. The problem persists in Motif 2.1.30.

Only True (XmPOPUP_KEYBOARD) and False (XmPOPUP_DISABLED) work in the resource file.

XmNradioAlwaysOne
: This resource is effective only when the XmNradioBehavior resource is True. XmNradioAlwaysOne, when set to True (default), ensures that one of the toggle buttons is always selected. Once this button is selected, clicking on it will not deselect it; it can be deselected only by selecting another toggle button. If XmNradioAlwaysOne is False, a selected toggle button can be deselected by clicking on it or by selecting another button.

XmNradioBehavior
: If True, the RowColumn widget acts like a RadioBox by setting two of the resources for its toggle button children. Namely, the XmNindicatorType resource defaults to XmONE_OF_MANY, and the XmNvisibleWhenOff resource defaults to True. The default value of the XmNradioBehavior resource is False, unless the RowColumn widget was created with the `XmCreateRadioBox`() routine.

XmNresizeHeight

XmNresizeWidth
: If True (default), the widget requests a new height or width when necessary. If False, no resize requests are made.

XmNrowColumnType
: The type of RowColumn widget to create. You can't change this resource after it's set. Convenience routines create a RowColumn widget of the appropriate type. Possible values:

XmWORK_AREA	XmMENU_PULLDOWN
XmMENU_BAR	XmMENU_OPTION
XmMENU_POPUP	

XmNspacing
: The horizontal and vertical spacing between children in the RowColumn widget. For RowColumn widgets of type XmOPTION_MENU or XmWORK_AREA, the default value is 3 pixels; for other RowColumn types, the default is 0.

XmNsubMenuId
: The widget ID for the pulldown menu pane to be associated with an OptionMenu. This resource is meaningful only in RowColumn widgets of type XmMENU_OPTION.

XmNtearOffModel
: In Motif 1.2 and later, specifies whether tear-off behavior is enabled for a RowColumn with XmN-row-ColumnType set to XmMENU_PULLDOWN or XmMENU_POPUP. In Motif 1.2, this resource cannot be set from a resource file unless a converter is installed by calling the function `XmRepTypeInstallTearOffModelConverter`(). Possible values:

XmTEAR_OFF_DISABLED	XmTEAR_OFF_ENABLED

In Motif 2.0 and later, the converter is automatically installed, and `XmRepTypeInstallTearOffModelConverter`() is obsolete.

XmNtearOffTitle
In Motif 2.0 and later, specifies the title of the TearOff shell.

XmNwhichButton
This resource has been superseded by the XmNmenuPost resource but is retained for compatibility with older releases of Motif.

New Constraint Resources

RowColumn defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNpositionIndex	XmCPositionIndex	short	XmLAST_POSITION	CSG

XmNpositionIndex
In Motif 1.2 and later, specifies the position of the widget in the RowColumn's list of children. A value of 0 indicates the beginning of the list, while XmLAST_POSITION places the child at the end of the list.

Simple Menu Creation Resources

The following resources are used with the simple menu creation routines. All resources have access C (create-only):

Name	Class	Type	Default
XmNbuttonAccelerators	XmCButtonAccelerators	StringTable	NULL
XmNbuttonAcceleratorText	XmCButtonAcceleratorText	XmStringTable	NULL
XmNbuttonCount	XmCButtonCount	int	0
XmNbuttonMnemonicCharSets	XmCButtonMnemonicCharSets	XmStringCharSetTable	NULL
XmNbuttonMnemonics	XmCButtonMnemonics	XmKeySymTable	NULL
XmNbuttons	XmCButtons	XmStringTable	NULL
XmNbuttonSet	XmCButtonSet	int	1
XmNbuttonType	XmCButtonType	XmButtonTypeTable	NULL
XmNoptionLabel	XmCOptionLabel	XmString	NULL
XmNoptionMnemonic	XmCOptionMnemonic	KeySym	NULL
XmNpostFromButton	XmCPostFromButton	int	-1
XmNsimpleCallback	XmCCallback	XtCallbackProc	NULL

XmNbuttonAccelerators
A list of accelerators, containing one item for each created title, separator, and button.

XmNbuttonAcceleratorText
A list of compound strings that represent the accelerators for the created buttons. The list contains one item for each created title, separator, and button.

XmNbuttonCount
The number of titles, separators, and menu buttons to create.

XmNbuttonMnemonicCharSets
A list of character sets to use for displaying button mnemonics. The list contains an item for each created title, separator, and button.

XmNbuttonMnemonics
A list of mnemonics associated with the buttons created. The list contains one item for each created title, separator, and button.

XmNbuttons
A list of compound strings that will serve as labels for the created buttons. The list contains one item for each created title, separator, and button.

XmNbuttonSet
The numeric position of the button to be initially set within a RadioBox or within an OptionMenu's pulldown submenu. The first button is specified as 0.

XmNbuttonType
A list of button types for the created buttons. The list contains one item for each created title, separator, and button. If this resource is not set, the buttons created will be CascadeButtonGadgets in a MenuBar and PushButtonGadgets in other types of RowColumn widget. The XmNbuttonType resource is an enumerated type whose possible values are:

XmPUSHBUTTON	XmCASCADEBUTTON
XmDOUBLE_SEPARATOR	XmCHECKBUTTON
XmRADIOBUTTON	XmSEPARATOR
XmTITLE	

XmNoptionLabel
A compound string with which to label the left side of an option menu.

XmNoptionMnemonic
The keysym of the key to press (in combination with the MAlt modifier) in order to post the pulldown menu associated with an option menu.

XmNpostFromButton
The numeric position of the cascade button (in the parent) from which the pull-down submenu is attached and subsequently posted. The first button is specified as 0.

XmNsimpleCallback
List of callbacks that are called when a button is pressed or when its value changes. For PushButtons and CascadeButtons, the callbacks are added to the XmNactivateCallback and for ToggleButtons they are added to the XmNvalueChangedCallback.

Callback Resources

RowColumn defines the following callback resources:

Callback	Reason Constant
XmNentryCallback	XmCR_ACTIVATE
XmNmapCallback	XmCR_MAP
XmNtearOffMenuActivateCallback	XmCR_TEAR_OFF_ACTIVATE
XmNtearOffMenuDeactivateCallback	XmCR_TEAR_OFF_DEACTIVATE
XmNunmapCallback	XmCR_UNMAP

XmNentryCallback
List of callbacks that are called when any button is pressed or when its value changes. When this resource is specified, the XmNactivateCallback and XmNvalueChangedCallback callbacks for all PushButtons, ToggleButtons, DrawnButtons, and CascadeButtons are disabled and instead call this callback. This resource must be specified when the RowColumn is created.

XmNmapCallback
List of callbacks that are called when the window associated with a RowColumn is going to be mapped.

XmNtearOffMenuActivateCallback
List of callbacks that are called when a tear-off menu pane is going to be torn off.

XmNtearOffMenuDeactivateCallback
List of callbacks that are called when a torn-off menu pane is going to be deactivated.

XmNunmapCallback
List of callbacks that are called when the window associated with a RowColumn is going to be unmapped.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;          /* the reason that the callback was called */
    XEvent      *event;          /* event structure that triggered callback */
    Widget      widget;          /* ID of activated RowColumn item          */
    char        *data;           /* value of application's client data      */
    char        *callbackstruct; /* created when item is activated          */
} XmRowColumnCallbackStruct;
```

The structure members *widget*, *data*, and *callbackstruct* are meaningful only when the callback *reason* is XmCR_ACTIVATE; otherwise, these structure members are set to NULL.

callbackstruct points to a structure that is created by the activation callback of the RowColumn item.

Inherited Resources

RowColumn inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. In Motif 2.0 and earlier, RowColumn sets the default value of XmNshadowThickness to 2 if XmN-rowColumnType is XmMENU_BAR, XmMENU_POPUP, or XmMENU_PULLDOWN; the resource is undefined when XmNrowColumnType is XmMENU_OPTION or XmWORK_AREA. In Motif 2.1 and later, the default value depends upon the XmDisplay XmNenableThinThickness resource: if True the default is 1, otherwise 2. The default value of XmNnavigationType is set to XmTAB_GROUP for a work area and XmNONE for an option menu; the resource is undefined for the other row column types. The default value of XmNtraversalOn is set to True for a work area or an option menu; the resource is undefined for the other row column types. The default value of XmNborderWidth is reset to 0 by XmManager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core

Resource	Inherited From	Resource	Inherited From
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Translations

The value of the XmN-rowColumnType resource determines the available translations. When XmNrowColumnType is XmWORK_AREA, RowColumn's translations are inherited from XmManager. When XmNrowColumnType is XmMENU_OPTION, RowColumn's translations are the traversal, KActivate, and KCancel translations inherited from XmManager, as well as the following:

Event	Action
BSelect Press	MenuBtnDown()
BSelect Release	MenuBtnUp()
KSelect	ManagerGadgetSelect()
KHelp	Help()

When XmNrowColumnType is XmMENU_BAR, XmMENU_PULLDOWN, or XmMENU_POPUP, RowColumn has the following translations (in PopupMenu systems, BMenu performs the BSelect actions as well):

Event	Action
BSelect Press	MenuBtnDown()
BSelect Release	MenuBtnUp()
KActivate	ManagerGadgetSelect()
KSelect	ManagerGadgetSelect()
Many KCancel	MenuGadgetEscape()

Event	Action
KHelp	Help()
KLeft	MenuGadgetTraverseLeft()
KRight	MenuGadgetTraverseRight()
KUp	MenuGadgetTraverseUp()
KDown	MenuGadgetTraverseDown()

Action Routines

RowColumn defines the following action routines:

Help()

Invokes any callbacks specified by the XmNhelpCallback resource.

ManagerGadgetSelect()

Arms and activates the gadget child (in a menu) that has focus. For a CascadeButtonGadget, its submenu is posted; for other gadget children, the menu hierarchy is unposted.

MenuBtnDown()

In a gadget child (in a menu), unposts any menus that were posted by the gadget's parent menu, turns mouse traversal on, and arms the gadget. If the child is a CascadeButtonGadget, its submenu is posted.

MenuBtnUp()

In a gadget child (in a menu), unposts the menu hierarchy and activates the gadget. If the child is a CascadeButtonGadget, this action posts the submenu and turns on keyboard traversal in the submenu.

MenuGadgetEscape()

Unposts the current menu and (unless the menu is a pulldown submenu) restores keyboard focus to the tab group or widget that previously had it (assuming an explicit focus policy). In a top-level pulldown menu pane attached to a menu bar, this action routine also disarms the cascade button and the menu bar.

MenuGadgetTraverseDown()

When the current menu item has a submenu and is in a MenuBar, disarms the current menu item, posts the submenu, and arms the first item in it. When the current menu item is in a menu pane, disarms the current menu item and arms the item below it, wrapping around to the top if necessary.

MenuGadgetTraverseLeft()

If the current menu item is in a MenuBar, disarms the current item and arms the MenuBar item to the left, wrapping around to the right if necessary. When the current item is in a menu pane, if the item is not at the left edge of the pane, disarms the current menu item and arms the item to its left. If the item is at the left edge of a submenu attached to the MenuBar, unposts the submenu, traverses to the MenuBar item to the left, and posts its submenu, wrapping if necessary.

MenuGadgetTraverseRight()

If the current menu item is in a MenuBar, disarms the current item and arms the MenuBar item to the right, wrapping around to the left if necessary. When the current item is in a menu pane, if the item is a CascadeButton, posts the associated submenu. If the current item is not at the right edge of the pane, disarms the current item and arms the item to the right, wrapping if necessary. Otherwise, unposts all submenus, traverses to the MenuBar item to the right, and posts its submenu, wrapping if necessary.

MenuGadgetTraverseUp()

Disarms the current menu item and arms the item above it, wrapping around to the bottom if necessary.

Additional Behavior

RowColumn has additional menu behavior:

KMenuBar

In a menu bar or in any menu pane cascaded from it, unposts the menu tree and (under an explicit focus policy) returns keyboard focus to the tab group that had it before entering the menu tree. In other non-popup menu panes, turns on keyboard traversal and sets the focus to the first menu bar item.

KMenu

Pops up the menu associated with the component with the keyboard focus and turns on keyboard traversal. In a popup menu system, unposts the menu tree and (under an explicit focus policy) returns keyboard focus to the tab group that had it before entering the menu tree.

See Also

XmCreateObject(1), XmGetMenuCursor(1), XmGetPostedFromWidget(1), XmGetTearOffControl(1), XmMenuPosition(1), XmOptionButtonGadget(1), XmOptionLabelGadget(1), XmRepTypeInstallTearOffModelConverter(1), XmSetMenuCursor(1), XmVaCreateSimpleCheckBox(1), XmVaCreateSimpleMenuBar(1), XmVaCreateSimpleOptionMenu(1), XmVaCreateSimplePopupMenu(1), XmVaCreateSimplePulldownMenu(1), XmVaCreateSimpleRadioBox(1), Composite(2), Constraint(2), Core(2), XmCascadeButton(2), XmCheckBox(2), XmManager(2), XmMenuBar(2), XmOptionMenu(2), XmPopupMenu(2), XmPulldownMenu(2), XmRadioBox(2).

Name

XmScale widget class –a manager widget that allows selection from a range of values.

Synopsis

Public Header:
<Xm/Scale.h>

Class Name:
XmScale

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmScale

Class Pointer:
xmScaleWidgetClass

Instantiation:
widget = `XmCreateScale` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmScaleWidgetClass,...)

Functions/Macros:
`XmCreateScale`(), `XmIsScale`(), `XmScaleGetValue`(), `XmScaleSetValue`(),
`XmScaleSetTicks`()

Description

A Scale displays a value from a range of values and allows a user to adjust the value. A Scale consists of a narrow, rectangular trough that contains a slider. The slider's position marks the current value within the range of values. Scale is a manager widget that orients its children along its axis. These children, typically labels, can be used as tick marks.

If the Scale is an input-output type, a user can change the value by moving the slider. An output-only Scale displays a value but does not allow the user to modify it. In Motif 2.0 and later, the XmNeditable resource controls the input-output type of the Scale; in Motif 1.2 and earlier, the programmer calls XtSetSensitive() or changes the inherited XmNsensitive resource to set the editable state.

In Motif 2.0 and later, the Scale supports tick marks directly through the XmScaleSetTicks() routine.

Traits

Scale holds the XmQTtransfer trait, which is inherited by any derived classes, and uses the XmQTspecifyRenderTable trait.

New Resources

Scale defines the following resources:

Name	Class	Type	Default	Access
XmNdecimalPoints	XmCDecimalPoints	short	0	CSG
XmNeditable	XmCEditable	Boolean	dynamic	CSG
XmNfontList	XmCFontList	XmFontList	dynamic	CSG
XmNhighlightOnEnter	XmCHighlightOnEnter	Boolean	False	CSG
XmNhighlightThickness	XmCHighlightThickness	Dimension	dynamic	CSG
XmNmaximum	XmCMaximum	int	100	CSG
XmNminimum	XmCMinimum	int	0	CSG
XmNorientation	XmCOrientation	unsigned char	XmVERTICAL	CSG
XmNprocessingDirection	XmCProcessingDirection	unsigned char	dynamic	CSG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNscaleHeight	XmCScaleHeight	Dimension	0	CSG
XmNscaleMultiple	XmCScaleMultiple	int	dynamic	CSG
XmNscaleWidth	XmCScaleWidth	Dimension	0	CSG
XmNshowArrows	XmCShowArrows	XtEnum	XmNONE	CSG
XmNshowValue	XmCShowValue	XtEnum	XmNONE	CSG
XmNsliderMark	XmCSliderMark	XtEnum	dynamic	CSG
XmNsliderSize	XmCSliderSize	int	dynamic	CSG
XmNsliderVisual	XmCSliderVisual	XtEnum	dynamic	CSG
XmNslidingMode	XmCSlidingMode	XtEnum	XmSLIDER	CSG
XmNtitleString	XmCTitleString	XmString	NULL	CSG
XmNvalue	XmCValue	int	dynamic	CSG

XmNdecimalPoints

A positive integer that determines how the slider's value will be displayed. The decimal point in the slider's value gets shifted to the right, and this resource specifies the number of decimal places to shift. For example, if the slider's value is 5678, then setting the XmNdecimalPoints[1] resource to 2 causes the widget to display the value as 56.78.

1.Erroneously given as XmdecimalPoints in 1st and 2nd editions.

XmNeditable
In Motif 2.0 and later, specifies whether the Scale responds to user input. The default depends upon the value of the XmNslidingMode resource. If the value is XmSLIDER, the default is True, and for the value XmTHERMOMETER the default is False.

XmNfontList
The font list used by the widget for the title. In Motif 2.0 and later, the XmFontList is obsolete, and is replaced by the XmRenderTable. Maintained for backwards compatibility, any specified render table takes precedence over the font list.

XmNhighlightOnEnter
Determines whether to draw the widget's highlighting rectangle whenever the cursor moves into the widget. This resource applies only when the shell has a focus policy of XmPOINTER. If the XmNhighlightOnEnter resource is True, highlighting is drawn; if False (default), highlighting is not drawn.

XmNhighlightThickness
The thickness of the highlighting rectangle. In Motif 2.0 and earlier, the default is 2. In Motif 2.1 and later, the default depends upon the XmDisplay XmNenableThinThickness resource: if True, the default is 1, otherwise 2.

XmNmaximum

XmNminimum
The maximum/minimum value of the slider.

XmNorientation
The direction in which the scale is displayed. Possible values:

```
XmVERTICAL                  /* top-to-bottom creation      */
XmHORIZONTAL                /* left-to-right creation      */
```

XmNprocessingDirection
Determines the position at which to display the slider's maximum and minimum values, with respect to the slider. Possible values:

```
XmMAX_ON_TOP                /* scale increases toward top      */
XmMAX_ON_BOTTOM             /* scale increases toward bottom */
XmMAX_ON_LEFT               /* scale increases toward left     */
XmMAX_ON_RIGHT              /* scale increases toward right  */
```

For vertically-oriented Scale widgets, the default value is XmMAX_ON_TOP. For horizontally-oriented Scale widgets, the default value is usually XmMAX_ON_RIGHT (depending on the value of the XmNstringDirection resource).

In Motif 2.0 and later, the XmNstringDirection resource is obsolete, and the default depends upon the XmNlayoutDirection value.

XmNrenderTable
In Motif 2.0 and later, specifies the XmRenderTable to use for both the title text string and the label displaying the current Scale value. If NULL, the value is found from the nearest ancestor holding the XmQTspecifyRenderTable trait, using the XmLABEL_RENDER_TABLE value of the ancestor.

XmNscaleHeight

XmNscaleWidth
The height or width of the slider area.

XmNscaleMultiple
The distance to move the slider when the user moves it by a multiple increment. The default value is calculated as (XmNmaximum – XmNminimum) / 10.

XmNshowArrows
In Motif 2.0 and later, specifies whether and how arrows are displayed on each end of the Scale. Possible values:

```
XmEACH_SIDE     /* arrow at both ends         */
XmMAX_SIDE      /* arrows at maximum end */
XmMIN_SIDE      /* arrows at minimum end */
XmNONE          /* arrows at neither end      */
```

XmNshowValue
In Motif 1.2 and earlier, a Boolean value which specifies whether the current scale value is displayed on an adjacent label. If True, the Scale displays the value beside the slider. If False, the value label isn't displayed. In Motif 2.0 and later, the type of the resource changes to an enumeration. XmNONE is equivalent to False, and does not display a value. XmNEAR_BORDER places the value adjacent to the border of the Scale, and XmNEAR_SLIDER places the value at the slider.

XmNsliderMark
In Motif 2.0 and later, specifies the appearance of the slider. The default depends upon the value of the XmNslidingMode resource. If the sliding mode is XmSLIDER, the default is XmETCHED_LINE. With a mode of XmTHERMOMETER, the default is XmNONE if the scale is editable, otherwise XmROUND_MARK. Possible values:

```
XmETCHED_LINE   /* drawn as an etched line                    */
XmNONE          /* drawn as a foreground rectangle            */
XmROUND_MARK    /* drawn as a shadowed circle                 */
XmTHUMB_MARK    /* three etched lines in foregrounded rectangle */
```

XmNsliderSize
In Motif 2.0 and later, an undocumented resource which represents the size of the slider in pixels.

XmNsliderVisual
In Motif 2.0 and later, specifies the color of the slider visual. The default is XmTROUGH_COLOR when the sliding model is XmTHERMOMETER, otherwise XmSHADOWED_BACKGROUND. Possible values:

```
XmBACKGROUND_COLOR      /* visual in background color          */
XmFOREGROUND_COLOR      /* visual in foreground color          */
XmSHADOWED_BACKGROUND   /* visual in background, with shadow */
XmTROUGH_COLOR          /* visual in trough color              */
```

XmNslidingMode
In Motif 2.0 and later, specifies the way in which the slider moves. Possible values:

```
XmSLIDER        /* slider moves freely between each end   */
XmTHERMOMETER   /* slider anchored to one end             */
```

XmNtitleString
The text string that appears as the title in the Scale widget.

XmNvalue
The current position of the slider along the scale. This resource must have a value between the values of XmNminimum and XmNmaximum.

Callback Resources

Scale defines the following callback resources:

Callback	Reason Constant
XmNconvertCallback	XmCR_OK
XmNdragCallback	XmCR_DRAG
XmNvalueChangedCallback	XmCR_VALUE_CHANGED

XmNconvertCallback
In Motif 2.0 and later, specifies a list of callbacks called when the slider is requested to convert a selection as part of a data transfer operation.

XmNdragCallback
List of callbacks that are called when the slider is being dragged.

XmNvalueChangedCallback
List of callbacks that are called when the position of the slider has changed.

Callback Structure

Convert callbacks are fully described within the sections covering the Uniform Transfer Model. See `XmTransfer`(1) for more details. For quick reference, a pointer to the following structure is passed to callbacks on the XmNconvertCallback list:

```
typedef struct {
int             reason;         /* the reason that the callback is invoked     */
XEvent          *event;         /* points to event that triggered callback     */
Atom            selection;      /* selection for which conversion is requested */
Atom            target;         /* the conversion target                       */
XtPointer       source_data;      /* selection source information              */
XtPointer       location_data;    /* information about data to be transferred */
int             flags;          /* input status of the conversion              */
XtPointer       parm;           /* parameter data for the target               */
int             parm_format;    /* format of parameter data                    */
unsigned long   parm_length;    /* number of elements in parameter data     */
Atom            parm_type;    /* the type of the parameter data              */
int             status;         /* output status of the conversion             */
XtPointer       value;          /* returned conversion data                    */
Atom            type;           /* type of conversion data returned            */
int             format;         /* format of the conversion data               */
unsigned long length;       /* number of elements in the conversion data   */
} XmConvertCallbackStruct;
```

Each drag and value changed callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
    XEvent      *event;         /* event structure that triggered callback */
    int         value;          /* new value of the slider                 */
} XmScaleCallbackStruct;
```

Inherited Resources

Scale inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. In Motif 2.0 and earlier, the default value of XmNshadowThickness is reset to 2. In Motif 2.1, the default value depends upon the XmDisplay XmNenableThinThickness resource: if True the default is 1, otherwise 2. The default value of XmNborderWidth is reset to 0 by XmManager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Translations

Scale does not define any new translations.

Behavior

Scale has the following behavior:

BSelect Press or BTransfer Press
: In the trough between the slider and an end of the Scale, moves the slider by one multiple increment in the direction of the end of the Scale. Calls the XmNvalueChangedCallback callbacks. Whether the value of the Scale is incremented or decremented depends on the value of XmNprocessingDirection. In the slider, starts interactive dragging of the slider.

BSelect Motion or BTransfer Motion
: If the button press occurred within the slider, the slider tracks the pointer and calls the XmNdragCallback callbacks.

BSelect Release or BTransfer Release
: If the button press occurs within the slider and the position of the slider has changed, the XmNvalueChangedCallback callbacks are invoked.

MCtrl BSelect Press
: In the trough between the slider and an end of the Scale, moves the slider to that end of the Scale and calls the XmNvalueChangedCallback callbacks. Whether the value of the Scale is incremented or decremented depends on the value of the XmNprocessingDirection resource.

KUp
KDown
: In a vertical Scale, moves the slider up or down one increment and calls the XmN-value-Changed-Callback callbacks. Whether the value of the Scale is incremented or decremented depends on the value of the XmN-processing-Direction resource.

KLeft
KRight
: In a horizontal Scale, moves the slider left or right one increment and calls the XmNvalue-Changed-Callback callbacks. Whether the value of the Scale is incremented or decremented depends on the value of the XmN-processing-Direction resource.

MCtrl KUp or KPageUp
MCtrl KDown or KPageDown
: In a vertical Scale, moves the slider up or down one multiple increment and calls the XmNvalueChangedCallback callbacks. Whether the value of the Scale is incremented or decremented depends on the value of the XmNprocessingDirection resource.

MCtrl KLeft or KPageLeft
MCtrl KRight or KPageRight
: In a horizontal Scale, moves the slider left or right one multiple increment and calls the XmNvalueChangedCallback callbacks. Whether the value of the Scale is incremented or decremented depends on the value of the XmNprocessingDirection resource.

KBeginLine or KBeginData
: Moves the slider to the Scale's minimum value and calls the XmNvalueChangedCallback callbacks.

KEndLine or KEndData
: Moves the slider to the Scale's maximum value and calls the XmNvalueChangedCallback callbacks.

KNextField
KPrevField
: Moves the keyboard focus to the first item in the next or previous tab group, wrapping if necessary.

KHelp
: Invokes the list of callbacks specified by XmNhelpCallback. If the Scale does not have any help callbacks, invokes those associated with the nearest ancestor that has them.

See Also

`XmCreateObject`(1), `XmScaleGetValue`(1), `XmScaleSetValue`(1), `XmScaleSetTicks`(1), `XmTransfer`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmManager`(2).

Name

XmScreen widget class – an object used to store screen-specific information.

Synopsis

Public Header:
<Xm/Screen.h>

Class Name:
XmScreen

Class Pointer:
xmScreenClass

Class Hierarchy:
Core → XmScreen

Instantiation:
widget = `XtAppInitialize`(...)

Functions/Macros:
`XmGetXmScreen`(), `XmIsScreen`()

Availability

Motif 1.2 and later.

Description

The Screen object stores screen-specific information for use by the toolkit. An application has a Screen object for each screen that it accesses. When an application creates its first shell on a screen, typically by calling `XtAppInitialize`() or `XtAppCreateShell`(), a Screen object is created automatically. There is no way to create a Screen independently. The function `XmGetXmScreen`() can be used to get the widget ID of the Screen object.

New Resources

Screen defines the following resources:

Name	Class	Type	Default	Access
XmNbitmapConversionModel	XmCBitmapConversionModel	XtEnum	XmMATCH_DEPTH	CSG
XmNcolorAllocationProc	XmCColorAllocationProc	XmAlloc-ColorProc	NULL	CSG
XmNcolorCalculationProc	XmCColorCalculationProc	XmScreen-ColorProc	NULL	CSG
XmNdarkThreshold	XmCDarkThreshold	int	dynamic	C
XmNdefaultCopyCursorIcon	XmCDefaultCopyCursorIcon	Widget	NULL	CSG
XmNdefaultInvalidCursorIcon	XmCDefaultInvalidCursorIcon	Widget	NULL	CSG
XmNdefaultLinkCursorIcon	XmCDefaultLinkCursorIcon	Widget	NULL	CSG

Name	Class	Type	Default	Access
XmNdefaultMoveCursorIcon	XmCDefaultMoveCursorIcon	Widget	NULL	CSG
XmNdefaultNoneCursorIcon	XmCDefaultNoneCursorIcon	Widget	NULL	CSG
XmNdefaultSourceCursorIcon	XmCDefaultSourceCursorIcon	Widget	NULL	CSG
XmNdefaultValidCursorIcon	XmCDefaultValidCursorIcon	Widget	NULL	CSG
XmNfont	XmCFont	XFontStruct *	NULL	CSG
XmNforegroundThreshold	XmCForegroundThreshold	int	dynamic	C
XmNhorizontalFontUnit	XmCHorizontalFontUnit	int	dynamic	CSG
XmNinsensitiveStippleBitmap	XmCInsensitiveStippleBitmap	Pixmap	50_foreground	C
XmNlightThreshold	XmCLightThreshold	int	dynamic	C
XmNmenuCursor	XmCCursor	Cursor[a]	arrow	C
XmNmoveOpaque	XmCMoveOpaque	Boolean	False	CSG
XmNunpostBehavior	XmCUnpostBehavior	unsigned char	XmUNPOST_AND_REPLAY	CSG
XmNuseColorObject	XmCUseColorObject	Boolean	False	C
XmNuserData	XmCUserData	XtPointer	NULL	CSG
XmNverticalFontUnit	XmCVerticalFontUnit	int	dynamic	CSG

a. Erroneously given as String in 1st and 2nd editions. The default is the arrow Cursor (XC_arrow).

XmNbitmapConversionModel
: In Motif 2.0 and later, specifies the way in which Xpm and Xbm files are converted to a pixmap. If the value is XmMATCH_DEPTH, the pixmap has the same depth as the widget to which it is associated. If the value is XmMATCH_DYNAMIC, Xbm files are converted to a pixmap of depth 1.

XmNcolorAllocationProc
: In Motif 2.0 and later, specifies an XmAllocColorProc procedure used for allocating color on the particular screen associated with the XmScreen object. If this is NULL, the default procedure `XAllocColor()` is used.

XmNcolorCalculationProc
: In Motif 2.0 and later, specifies an XmScreenColorProc procedure for calculating the default foreground, background, top shadow, bottom shadow, select colors on the particular screen associated with the XmScreen object. If this is NULL, color is calculated using a default screen-independent procedure. The default procedure can be changed by `XmSetColorCalculation`().

XmNdarkThreshold
: The level of perceived brightness (between 0 and 100) that is treated as a "dark" background color when computing default shadow and select colors.

XmNdefaultCopyCursorIcon
: The DragIcon used during a copy operation. When the value is NULL, a default system icon is used.

XmNdefaultInvalidCursorIcon
: The DragIcon used when the pointer is over an invalid drop site. When the value is NULL, a default system icon is used.

XmNdefaultLinkCursorIcon
: The DragIcon used during a link operation. When the value is NULL, a default system icon is used.

XmNdefaultMoveCursorIcon
: The DragIcon used during a move operation. When the value is NULL, a default system icon is used.

XmNdefaultNoneCursorIcon
: The DragIcon used when the pointer is not over a drop site. When the value is NULL, a default system icon is used.

XmNdefaultSourceCursorIcon
: The bitmap used as a cursor when an XmNsourceCursorIcon is not provided by the DragContext. When the value is NULL, a default system icon is used.

XmNdefaultValidCursorIcon
: The DragIcon used when the pointer is over a valid drop site. When the value is NULL, a default system icon is used.

XmNfont
: The font used in computing values for XmNhorizontalFontUnit and XmN-verticalFontUnit.

XmNforegroundThreshold
: The level of perceived brightness (between 0 and 100) that distinguishes between a "dark" and "light" background when computing the default foreground and highlight colors.

XmNhorizontalFontUnit
: The horizontal component of the font units that are used to convert geometry values when XmNshellUnitType or XmNunitType is set to Xm100TH_FONT_UNITS. If a value is not specified, the default is computed from the XmNfont resource.

XmNinsensitiveStippleBitmap
: In Motif 2.0 and later, specifies a default stipple for drawing widgets in insensitive state. Mostly used within the graphics contexts of Gadgets.

XmNlightThreshold
: The level of perceived brightness (between 0 and 100) that is treated as a "light" background color when computing default shadow and select colors.

XmNmenuCursor
The cursor that is used when the application posts a menu. Possible values include all of the cursors in the X cursor font.

XmNmoveOpaque
If False (default), an operation that moves a window displays an outline of the window during the operation. If True, a move operation displays a representation of the window.

XmNunpostBehavior
The behavior of a posted menu when the pointer button is pressed outside of the menu. Possible values:

```
XmUNPOST_AND_REPLAY     /* unposts the menu hierarchy and
replays event */
XmUNPOST                /* unposts the menu hierar-
chy            */
```

XmNuseColorObject
In Motif 2.0 and later, specifies whether colors are shareable between widgets, and whether an alteration to a color dynamically changes all widgets which reference the color.

XmNuserData
In Motif 2.0 and later, specifies a pointer to data that the application can attach to the structure representing the screen. The resource is unused internally.

XmNverticalFontUnit
The vertical component of the font units that are used to convert geometry values when XmNshellUnitType or XmNunitType is set to Xm100TH_FONT_UNITS or XmFONT_UNITS[1]. If a value is not specified, the default is computed from the XmNfont resource.

Procedures

The XmScreenColorProc has the following syntax:

typedef void (*XmScreenColorProc) (Screen *, XColor *, XColor *, XColor *, XColor *, XColor *)

```
Screen      *screen;      /* the screen                  */
XColor      *bg_color;    /* specifies the background    color */
XColor      *fg_color;    /* returns   the foreground    color */
XColor      *sel_color;   /* returns   the select        color */
XColor      *ts_color;    /* returns   the top    shadow color */
XColor      *bs_color;    /* returns   the bottom shadow color */
```

1.Erroneously given as Xm_FONT_UNITS in 2nd edition.

An XmScreenColorProc takes six arguments. The first argument is a pointer to the screen. The second argument, bg_color, is a pointer to an XColor structure that specifies the background color. The red, green, blue, and pixel fields in the structure contain valid values. The rest of the arguments are pointers to XColor structures for the colors that are to be calculated. The procedure fills in the red, green, and blue fields in these structures.

The XmAllocColorProc has the following syntax:

```
typedef void (*XmAllocColorProc) (Display *, Colormap, XColor *)

    Display     *display;       /* connection to X server          */
    Colormap    colormap;       /* a colormap in which to allocate color */
    XColor      *bs_color;      /* specifies and returns allocated color */
```

An XmAllocColorProc takes three arguments. The first argument is a pointer to the Display connection. The second argument is the Colormap where the color is to be allocated. The third argument is a pointer to an XColor structure for the color that is to be allocated. The programmer fills in the red, green, and blue fields in the structure to the required values, and the procedure returns the actually allocated values into the same fields.

Inherited Resources

None of the resources inherited by Screen are applicable.

See Also

`XmGetXmScreen`(1), `XmSetColorCalculation`(1), `Core`(2), `XmDisplay`(2).

Name

XmScrollBar widget class – a widget to control the scrolling of the viewing area in another widget.

Synopsis

Public Header:
<Xm/ScrollBar.h>

Class Name:
XmScrollBar

Class Hierarchy:
Core → XmPrimitive → XmScrollBar

Class Pointer:
xmScrollBarWidgetClass

Instantiation:
widget = `XmCreateScrollBar` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmScrollBarWidgetClass,...)

Functions/Macros:
`XmCreateScrollBar`(), `XmIsScrollBar`(), `XmScrollBarGetValues`(),
`XmScrollBarSetValues`()

Description

A ScrollBar allows users to reposition data that is too large to fit in the viewing window. Although a ScrollBar can be used as a standalone widget, it is normally used in a ScrolledWindow. A ScrollBar consists of a rectangular strip, called the scroll region or trough, and two arrows placed on either end of the scroll region. Within the scroll region is a smaller, movable rectangle called the slider. To scroll the data, users can click on one of the arrows, click in the scroll region, or drag the slider. The application typically sets the XmNsliderSize resource such that the size of the slider relative to the size of the scroll region corresponds to the percentage of total data that is currently displayed.

Traits

ScrollBar holds the XmQTnavigator trait, which is inherited by any derived classes.

New Resources

ScrollBar defines the following resources:

Name	Class	Type	Default	Access
XmNeditable	XmCEditable	Boolean	dynamic	CSG
XmNincrement	XmCIncrement	int	1	CSG
XmNinitialDelay	XmCInitialDelay	int	250	CSG
XmNmaximum	XmCMaximum	int	dynamic	CSG
XmNminimum	XmCMinimum	int	0	CSG
XmNorientation	XmCOrientation	unsigned char	XmVERTICAL	CSG
XmNpageIncrement	XmCPageIncrement	int	10	C
XmNprocessingDirection	XmCProcessingDirection	unsigned char	dynamic	CSG
XmNrepeatDelay	XmCRepeatDelay	int	50	CSG
XmNshowArrows	XmCShowArrows	XtEnum	XmEACH_SIDE	CSG
XmNsliderMark	XmCSliderMark	XtEnum	dynamic	CSG
XmNsliderSize	XmCSliderSize	int	dynamic	CSG
XmNsliderVisual	XmCSliderVisual	XtEnum	dynamic[a]	CSG
XmNslidingMode	XmCSlidingMode	XtEnum	XmSLIDER	CSG
XmNsnapBackMultiple	XmCSnapBackMultiple	unsigned short	65535	CSG
XmNtroughColor	XmCTroughColor	Pixel	dynamic	CSG
XmNvalue	XmCValue	int	dynamic	CSG

a. Erroneously given as XmSHADOWED_BACKGROUND in 2nd edition.

XmNeditable
: In Motif 2.0 and later, specifies whether the ScrollBar responds to user input. The default depends upon the value of the XmNslidingMode resource. If the value is XmSLIDER, the default is True, and for the value XmTHERMOMETER the default is False.

XmNincrement
: The amount the value changes due to the user's moving the slider one increment.

XmNinitialDelay
: The number of milliseconds a button must remain pressed before triggering continuous slider movement.

XmNmaximum

XmNminimum
: The maximum/minimum value of the slider.

XmNorientation
The direction in which the scale is displayed. Possible values:

```
XmVERTICAL          /* top-to-bottom creation */
XmHORIZONTAL        /* left-to-right creation */
```

XmNpageIncrement
The amount the value changes due to the user's moving the slider one page increment.

XmNprocessingDirection
Determines the position at which to display the slider's maximum and minimum values, with respect to the slider. Possible values:

```
XmMAX_ON_TOP        /* scale increases toward top    */
XmMAX_ON_BOTTOM     /* scale increases toward bottom */
XmMAX_ON_LEFT       /* scale increases toward left   */
XmMAX_ON_RIGHT      /* scale increases toward right */
```

For vertically oriented ScrollBar widgets, the default value is XmMAX_ON_TOP. For horizontally oriented ScrollBar widgets, the default value is usually XmMAX_ON_RIGHT (depending on the value of the XmNstringDirection resource).

XmNrepeatDelay
The number of milliseconds a button must remain pressed before continuing further slider motions, once the XmNinitialDelay time has been triggered.

XmNshowArrows
In Motif 1.2 and earlier, a Boolean value which indicates whether arrows are displayed. If True, arrows are displayed; if False, they are not.

In Motif 2.0 and later, the resource is represented by an enumerated type: if XmEACH_SIDE, arrows are displayed at each end of the ScrollBar, XmMAX_SIDE displays both[1] arrows at the end where the maximum value is displayed, XmMIN_SIDE displays both[2] arrows at the minimum value end, and XmNONE does not display any arrows.

XmNsliderMark
In Motif 2.0 and later, specifies the appearance of the slider. The default depends upon the value of the XmNslidingMode resource. If the sliding mode is XmSLIDER, the default is XmETCHED_LINE. With a mode of XmTHERMOMETER, the default is XmNONE if the scale is editable, otherwise XmROUND_MARK. Possible values:

1.Erroneously given as *one* arrow in 2nd edition.

2.Erroneously given as *one* arrow in 2nd edition

```
XmETCHED_LINE       /* drawn as an etched line                          */
XmNONE              /* drawn as a foreground rectangle                  */
XmROUND_MARK        /* drawn as a shadowed circle                       */
XmTHUMB_MARK        /* three etched lines in foregrounded rectangle */
```

XmNsliderSize
The slider's length. The length ranges from 1 to the value of XmNmaximum – XmNminimum. By default, the value is computed to be:

(XmNmaximum – XmNminimum) / 10.

XmNsliderVisual
In Motif 2.0 and later, specifies the color of the slider visual. The default is XmTROUGH_COLOR when the sliding model is XmTHERMOMETER, otherwise XmSHADOWED_BACKGROUND. Possible values:

```
XmBACKGROUND_COLOR      /* visual in background color           */
XmFOREGROUND_COLOR      /* visual in foreground color           */
XmSHADOWED_BACKGROUND   /* visual in background, with shadow */
XmTROUGH_COLOR          /* visual in trough color               */
```

XmNslidingMode
In Motif 2.0 and later, specifies the way in which the slider moves. Possible values:

```
XmSLIDER        /* slider moves freely between each end     */
XmTHERMOMETER   /* slider anchored to one end               */
```

XmNsnapBackMultiple
In Motif 2.0 and later, specifies a distance, which if exceeded, causes the ScrollBar to snap back to its original settings. The resource comes into effect when the user drags the mouse outside the bounds of the ScrollBar. The resource is measured in terms of multiples of the ScrollBar width. For example, the value 0 (zero) causes the slider to snap back as soon as the pointer moves outside the ScrollBar, the value 1 snaps back at one ScrollBar width, etc. The default is very large, in order to disable snap back even if the size of the screen is abnormal.

XmNtroughColor
The color of the slider's trough.

XmNvalue
The slider's position. The position ranges from the value of XmNminimum to the value of (XmNmaximum – XmNsliderSize).

Callback Resources

ScrollBar defines the following callback resources:

Callback	Reason Constant
XmNdecrementCallback	XmCR_DECREMENT
XmNdragCallback	XmCR_DRAG
XmNincrementCallback	XmCR_INCREMENT
XmNpageDecrementCallback	XmCR_PAGE_DECREMENT
XmNpageIncrementCallback	XmCR_PAGE_INCREMENT
XmNtoBottomCallback	XmCR_TO_BOTTOM
XmNtoTopCallback	XmCR_TO_TOP
XmNvalueChangedCallback	XmCR_VALUE_CHANGED

XmNdecrementCallback
: List of callbacks that are called when the value of the ScrollBar decreases by one increment.

XmNdragCallback
: List of callbacks that are called for each change in position when the slider is being dragged.

XmNincrementCallback
: List of callbacks that are called when the value of the ScrollBar increases by one increment.

XmNpageDecrementCallback
: List of callbacks that are called when the value of the ScrollBar decreases by one page increment.

XmNpageIncrementCallback
: List of callbacks that are called when the value of the ScrollBar increases by one page increment.

XmNtoBottomCallback
: List of callbacks that are called when the slider is moved to the maximum value of the ScrollBar.

XmNtoTopCallback
: List of callbacks that are called when the slider is moved to the minimum value of the ScrollBar.

XmNvalueChangedCallback
: List of callbacks that are called at the end of a slider drag operation. These callbacks are also called in place of each of the other ScrollBar callbacks that reports a value change when the callback resource is NULL.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;     /* the reason that the callback was called */
    XEvent      *event;     /* event structure that triggered callback */
    int         value;      /* value of the slider's new location      */
    int         pixel;      /* coordinate where selection occurred     */
} XmScrollBarCallbackStruct;
```

pixel is meaningful only when the callback *reason* is XmCR_TO_TOP or XmCR_TO_BOTTOM. The *pixel* member specifies the location at which the mouse button selection occurred, giving the x-coordinate in the case of a horizontal ScrollBar and the y-coordinate in the case of a vertical ScrollBar.

Inherited Resources

ScrollBar inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. XmNnavigationType to XmSTICKY_TAB_GROUP, and XmN-traversalOn to False. The default value of XmNborderWidth is reset to 0 by Manager.

In versions of Motif prior to 2.0, the default value of XmNhighlightThickness is reset to zero by the ScrollBar. In Motif 2.0, the default is reset to 2 if the ScrollBar's parent is a ScrolledWindow, zero otherwise. In Motif 2.1, if the XmNenableThinThickness resource of XmDisplay is True, the default is 1 if the ScrollBar's parent is a ScrolledWindow, zero otherwise.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNhighlightThickness	XmPrimitive
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNlayoutDirection	XmPrimitive
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnavigationType	XmPrimitive
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmPrimitive
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmPrimitive	XmNsensitive	Core
XmNbottomShadowPixmap	XmPrimitive	XmNshadowThickness	XmPrimitive
XmNcolormap	Core	XmNtoolTipString	XmPrimitive
XmNconvertCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNdepth	Core	XmNtopShadowPixmap	XmPrimitive
XmNdestroyCallback	Core	XmNtranslations	Core

Resource	Inherited From	Resource	Inherited From
XmNforeground	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNheight	Core	XmNunitType	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNuserData	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNwidth	Core
XmNhighlightOnEnter	XmPrimitive	XmNx	Core
XmNhighlightPixmap	XmPrimitive	XmNy	Core

Translations

The translations for ScrollBar include those from Primitive, plus the following:

Event	Action
BSelect Press	Select()
BSelect Release	Release()
BSelect Press Moved	Moved()
BTransfer Press	Select()
BTransfer Release	Release()
BTransfer Press Moved	Moved()
MCtrl BSelect Press	TopOrBottom()
MCtrl BSelect Release	Release()
KUp	IncrementUpOrLeft(0)
MCtrl KUp	PageUpOrLeft(0)
KDown	IncrementDownOrRight(0)
MCtrl KDown	PageDownOrRight(0)
KLeft	IncrementUpOrLeft(1)
MCtrl KLeft	PageUpOrLeft(1)
KRight	IncrementDownOrRight(1)
MCtrl KRight	PageDownOrRight(1)
KPageUp	PageUpOrLeft(0)
KPageDown	PageDownOrRight(0)
KPageLeft	PageUpOrLeft(1)
KPageRight	PageDownOrRight(1)
KBeingLine	TopOrBottom()
KEndLine	TopOrBottom()
KBeginData	TopOrBottom()
KEndData	TopOrBottom()

Event	Action
KNextField	PrimitiveNextTabGroup()
KPrevField	PrimitivePrevTabGroup()
KActivate	PrimitiveParentActivate()
KCancel	CancelDrag()
KHelp	PrimitiveHelp()

Action Routines

ScrollBar defines the following action routines:

CancelDrag()

In Motif 1.2 and later, cancels the scrolling operation and returns the slider to its previous location if the event happened during a drag. Otherwise, passes the event to the parent if it is a manager.

IncrementDownOrRight(flag):

Moves the slider by one increment--downward if flag is 0; to the right if flag is 1. Depending on the value of the XmNprocessingDirection resource, the slider's movement invokes the list of callbacks specified by either the XmNincrementCallback or the XmNdecrementCallback resource (or XmNvalueChangedCallback if the appropriate callback resource is NULL).

IncrementUpOrLeft(flag):

Same as IncrementDownOrRight except that the slider moves upward if flag is 0 and to the left if flag is 1.

Moved()

This action applies when the mouse button is pressed in the slider. When this is done, moving the pointer moves the slider along with it and also invokes the callbacks specified by XmNdragCallback.

PageDownOrRight(flag):

Moves the slider by one page increment--downward if flag is 0; to the right if flag is 1. Depending on the value of the XmNprocessingDirection resource, the slider's movement invokes the callbacks listed in either XmNpageIncrementCallback or XmNpageDecrementCallback (or XmNvalueChangedCallback if the appropriate callback resource is NULL).

PageUpOrLeft(flag):

Same as IncrementDownOrRight except that the slider moves upward if flag is 0 and to the left if flag is 1.

PrimitiveHelp()

Invokes the list of callbacks specified by XmNhelpCallback. If the ScrollBar doesn't have any help callbacks, the Help() routine invokes those associated with the nearest ancestor that has them.

PrimitiveNextTabGroup()
PrimitivePrevTabGroup()

Traverses to the first item in the next/previous tab group, wrapping if necessary.

PrimitiveParentActivate()

In Motif 1.2 and later, passes the event to the parent if it is a manager.

Release()

If the Moved() action changes the slider's position, then the Release() action invokes the callbacks specified by XmNvalueChangedCallback.

Select()

The results of this action depend on the location in which its applied: Within an arrow, this action is the same as IncrementDownOrRight() or IncrementUpOrLeft()--incrementing or decrementing according to the value of the XmN-processingDirection resource, and invoking the appropriate increment or decrement callback. Within the scrolling area that lies between an arrow and the slider, this action works like the page increment action routines--moving by one page increment according to the value of the XmNprocessingDirection resource, and invoking the appropriate page increment or page decrement callback. Within either of these locations, keeping the button pressed repeats the incremental movement of the slider. This behavior is triggered when the duration of the button press exceeds the value of the XmNinitialDelay resource; the slider movement then repeats with a time interval specified by the XmNrepeatDelay resource. Within the slider, this action begins slider dragging, which is subsequently affected by the actions Moved() and Release().

TopOrBottom()

Moves the slider to its minimum value and invokes the callbacks specified by XmNtoTopCallback, or moves the slider to its maximum value and invokes the callbacks specified by XmNtoBottomCallback. The direction of the slider's movement depends on the value of the XmNprocessingDirection resource. This action can be applied using either keyboard or mouse events.

See Also

`XmCreateObject`(1), `XmScrollBarGetValues`(1), `Core`(2), `XmPrimitive`(2).

Name

XmScrolledList –a List as a child of a ScrolledWindow.

Synopsis

Public Header:
<Xm/List.h>

Instantiation:
widget = `XmCreateScrolledList` (parent, name,...)

Functions/Macros:
`XmCreateScrolledList`(), `XmCreateScrolledWindow`()

Description

An XmScrolledList is a compound object created by a call to XmCreateScrolledList() that provides scroll bars for a list that is not visible all at once. A ScrolledList consists of a ScrolledWindow widget with a List widget as its child.

A ScrolledList automatically creates the necessary scroll bars. The ScrolledWindow resource XmNscrollingPolicy is set to XmAPPLICATION_DEFINED and XmNvisualPolicy is set to XmVARIABLE. The ScrolledWindow resource XmNscrollBarDisplayPolicy is set to XmSTATIC, but no initial value is set for the List XmNscrollBarDisplayPolicy resource.

Default Resource Values

A ScrolledList sets the following default values for ScrolledWindow resources:

Name	Default
XmNscrollBarDisplayPolicy	XmSTATIC
XmNscrollingPolicy	XmAPPLICATION_DEFINED
XmNvisualPolicy	XmVARIABLE

Widget Hierarchy

When a ScrolledList is created with a specified name, the ScrolledWindow is named name*SW* and the List is called *name*. The horizontal and vertical scroll bars are named HorScrollBar and VertScrollBar, respectively.

See Also

`XmCreateObject`(1), `XmList`(2), `XmScrolledWindow`(2).

Name

XmScrolledText –a Text widget as a child of a ScrolledWindow.

Synopsis

Public Header:
<Xm/Text.h>

Instantiation:
`widget = XmCreateScrolledText (parent, name,...)`

Functions/Macros:
`XmCreateScrolledText`(), `XmCreateScrolledWindow`()

Description

An XmScrolledText is a compound object created by a call to XmCreateScrolled-Text() that provides scroll bars for text that is not visible all at once. A Scrolled-Text object consists of a ScrolledWindow widget with a multi-line Text widget as its child.

ScrolledText automatically creates the necessary scroll bars. The ScrolledWin-dow resource XmNscrollingPolicy is set to XmAPPLICATION_DEFINED, XmNvisualPolicy is set to XmVARIABLE and XmNscrollBarDisplayPolicy is set to XmSTATIC.

Default Resource Values

ScrolledText sets the following default values for ScrolledWindow resources:

Name	Default
XmNscrollBarDisplayPolicy	XmSTATIC
XmNscrollingPolicy	XmAPPLICATION_DEFINED
XmNvisualPolicy	XmVARIABLE

Widget Hierarchy

When a ScrolledText object is created with a specified name, the ScrolledWin-dow is named name*SW* and the Text widget is called *name*. The horizontal and vertical scroll bars are named HorScrollBar and VertScrollBar respectively.

See Also

`XmCreateObject`(1), `XmScrolledWindow`(2), `XmText`(2).

Name

XmScrolledWindow widget class – a manager widget that provides scroll bars for the data display.

Synopsis

Public Header:
<Xm/ScrolledW.h>

Class Name:
XmScrolledWindow

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmScrolledWindow

Class Pointer:
xmScrolledWindowWidgetClass

Instantiation:
widget = XmCreateScrolledWindow (parent, name,...)
or
widget = XtCreateWidget (name, xmScrolledWindowWidgetClass,...)

Functions/Macros:
`XmCreateScrolledList`(), `XmCreateScrolledText`(), `XmCreateScrolledWindow`(), `XmIsScrolledWindow`(), `XmScrollVisible`(), `XmScrolledWindowSetAreas`()

Description

ScrolledWindow provides a scrollable view of data that may not be visible all at once. ScrollBars allow a user to scroll the visible part of the window through the larger display. A ScrolledWindow widget can be created so that it scrolls automatically without application intervention or so that an application provides support for all scrolling operations. When scrolling is handled automatically, ScrolledWindow creates the scroll bars, which are named HorScrollBar and VertScrollBar.

Each of the ScrolledWindow regions is associated with a ScrolledWindow resource; `XmScrolledWindowSetAreas`() sets the associated resources. If an application does not call `XmScrolledWindowSetAreas`(), the widget may still set some of the standard regions. If ScrollBars are added as children, the XmNhorizontalScrollBar and XmNverticalScrollBar resources may be set if

they have not already been specified. Any child that is not a ScrollBar is used for the XmNworkWindow. If you want to be certain about which widgets are used for the different regions, it is wise to call `XmScrolledWindowSetAreas`() explicitly.

Traits

ScrolledWindow holds the XmQTscrollFrame trait, which is inherited by any derived classes, and uses the XmQTnavigator trait.

New Resources

ScrolledWindow defines the following resources:

Name	Class	Type	Default	Access
XmNautoDragModel	XmCAutoDragModel	XtEnum	XmAUTO_DRAG_ENABLED	G
XmNclipWindow	XmCClipWindow	Widget	dynamic	G
XmNhorizontalScrollBar	XmCHorizontalScrollBar	Widget	dynamic	CSG
XmNscrollBarDisplayPolicy	XmCScrollBarDisplayPolicy	unsigned char	dynamic	CSG
XmNscrollBarPlacement	XmCScrollBarPlacement	unsigned char	XmBOTTOM_RIGHT	CSG
XmNscrolledWindowMarginHeight	XmCScrolledWindowMargin-Height	Dimension	0	CSG
XmNscrolledWindowMarginWidth	XmCScrolledWindowMargin-Width	Dimension	0	CSG
XmNscrollingPolicy	XmCScrollingPolicy	unsigned char	XmAPPLICATION_DEFINED	CG
XmNspacing	XmCSpacing	Dimension	4	CSG
XmNverticalScrollBar	XmCVerticalScrollBar	Widget	dynamic	CSG
XmNvisualPolicy	XmCVisualPolicy	unsigned char	dynamic	C
XmNworkWindow	XmCWorkWindow	Widget	NULL	CSG

XmNautoDragModel
: In Motif 2.0 and later, specified whether automatic drag is enabled. Possible values:

  XmAUTO_DRAG_ENABLED XmAUTO_DRAG_DISABLED

XmNclipWindow
: The widget ID of the clipping area. The clipping window exists only when the XmNvisualPolicy resource is set to XmCONSTANT. The XmNclipWindow resource cannot be set to a new value.

XmNhorizontalScrollBar
The widget ID of the horizontal ScrollBar.

XmNscrollBarDisplayPolicy
Controls the placement of ScrollBars, depending on the value of the XmNscrollingPolicy resource. Possible values:

```
XmSTATIC          /* vertical ScrollBar always displays      */
XmAS_NEEDED       /* add ScrollBar when view is clipped      */
```

If XmNscrollingPolicy is set to XmAUTOMATIC, then XmNscrollBarDisplayPolicy defaults to a value of XmAS_NEEDED, and ScrollBars are displayed only when the workspace cannot fit within the clip area. If XmNscrollingPolicy is set to XmAPPLICATION_DEFINED, then XmNscrollBarDisplayPolicy defaults to (and must remain with) a value of XmSTATIC. This means that ScrollBars will always be displayed.

XmNscrollBarPlacement
The positions of the ScrollBars relative to the work window. The default value of this resource depends on the value of the XmNstringDirection resource. Possible values:

```
XmTOP_LEFT        /* vertical ScrollBar on left; horizontal on top      */
XmBOTTOM_LEFT     /* vertical ScrollBar on left; horizontal on bottom */
XmTOP_RIGHT       /* vertical ScrollBar on right; horizontal on top     */
XmBOTTOM_RIGHT    /* vertical ScrollBar on right; horizontal on bottom */
```

XmNscrolledWindowMarginHeight
The spacing at the top and bottom of the ScrolledWindow.

XmNscrolledWindowMarginWidth
The spacing at the right and left sides of the ScrolledWindow.

XmNscrollingPolicy
Determines how automatic scrolling occurs. Possible values:

```
XmAUTOMATIC           /* ScrolledWindow handles scrolling    */
XmAPPLICATION_DEFINED /* application handles scrolling        */
```

XmNspacing
The distance between each ScrollBar and the work window.

XmNverticalScrollBar
The widget ID of the vertical ScrollBar.

XmNvisualPolicy
The visual layout policy of the ScrolledWindow. In Motif 2.0 and later, the resource is obsolete, and the internal widget initialization functions ensure that the policy is consistent. Possible values:

```
XmCONSTANT      /* viewing area is clipped if needed;                    */
                /* default when XmNscrollingPolicy is XmAUTOMATIC */
XmVARIABLE      /* layout grows or shrinks; default otherwise            */
```

XmNworkWindow
The widget ID of the viewing area.

Callback Resources

ScrolledWindow defines the following callback resources:

Callback	Reason Constant
XmNtraverseObscuredCallback	XmCR_OBSCURED_TRAVERSAL

XmNtraverseObscuredCallback
List of callbacks that are called when the keyboard focus is moved to a widget or gadget that is obscured from view.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int                 reason;             /* reason that callback was called */
    XEvent              *event;             /* event that triggered callback   */
    Widget          traversal_destination;  /* widget or gadget to traverse to */
    XmTraversalDirection   direction;       /* direction of traversal          */
} XmTraverseObscuredCallbackStruct;
```

New Constraint Resources

In Motif 2.0 and later, ScrolledWindow defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNscrolledWindowChildType	XmCScrolledWindowChildType	unsigned char	dynamic	CG

XmNscrolledWindowChildType
Specifies the logical type of child of the ScrolledWindow. Possible values:

```
XmHOR_SCROLLBAR     /* horizontal ScrollBar                          */
XmVERT_SCROLLBAR    /* vertical   ScrollBar                          */
XmSCROLL_HOR        /* horizontal ScrollBar - horizontal scrolling only */
XmSCROLL_VERT       /* vertical   ScrollBar - vertical   scrolling only */
XmWORK_AREA         /* work area child                               */
XmCLIP_WINDOW       /* XmClipWindow                                  */
XmNO_SCROLL         /* no child scrolling                            */
```

The values XmSCROLL_HOR, XmSCROLL_VERT, and XmNO_SCROLL are only valid if the scrolling policy is XmAUTOMATIC.

Inherited Resources

ScrolledWindow inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. ScrolledWindow sets the default value of XmNshadowThickness dynamically. The default value of XmNborderWidth is reset to 0 by XmManager.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Widget Hierarchy

When the ScrolledWindow has an XmNvisualPolicy of XmCONSTANT (the XmNscrollingPolicy is XmAUTOMATIC) the ScrolledWindow creates an additional clip widget which is used as the parent of any added work window: the additional widget acts as the logical viewport, and the XmNclipWindow resource is set to the ID of this.

Before Motif 2.0, the clip widget has the name ScrolledWindowClipWindow. In Motif 2.0 and later, the name is changed to ClipWindow: any resources, and XtNameToWidget() or similar code which relies upon the name should be changed accordingly.

Translations

The translations for ScrolledWindow include those from Manager.

Additional Behavior

ScrolledWindow has the following additional behavior when the XmNscrolling-Policy resource is XmAUTOMATIC:

See Also

`XmCreateObject`(1), `XmScrollVisible`(1), `XmScrolledWindowSetAreas`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmManager`(2), `XmScrollBar`(2), `XmScrolledList`(2), `XmScrolledText`(2).

Name

XmSelectionBox widget class – a widget for selecting one of a list of alternatives.

Synopsis

Public Header:
<Xm/SelectioB.h>

Class Name:
XmSelectionBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmBulletinBoard → XmSelectionBox

Class Pointer:
xmSelectionBoxWidgetClass[1]

Instantiation:
widget = `XmCreateSelectionBox` (parent, name,...)
or
widget = `XtCreateWidget` (name, XmSelectionBoxWidgetClass,...)

Functions/Macros:
`XmCreateSelectionBox`(), `XmCreateSelectionDialog`(), `XmCreatePromptDialog`(),
`XmIsSelectionBox`(), `XmSelectionBoxGetChild`()

Description

SelectionBox is a composite widget that displays a scrollable list of alternatives from which the user can choose items. A SelectionBox contains a text field in which the user can enter a selection, the scrollable list of selections, labels for the text field and the scrollable list, a separator, and a group of three or four buttons. The names of these components in the SelectionBox are Items, ItemsList, Selection, Text, and Separator, respectively.

1.Erroneously given as XmSelectionBoxWidgetClass in 1st and 2nd editions.

In Motif 1.2 and later, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Apply**, **Cancel**, and **Help** by default. The **Apply** button is created but not always managed. If the parent of the SelectionBox is a DialogShell the button is managed, otherwise it is not.

You can customize a SelectionBox by removing existing children or adding new children. Use `XmSelectionBoxGetChild`() to retrieve the widget ID of an existing child and then unmanage the child. With Motif 1.2 and later, multiple widgets can be added as children of a SelectionBox. The first child is considered a work area and is placed based on the value of the XmNchildPlacement resource. If a menu bar is added, it is placed at the top of the window. Any buttons are placed after the **OK** button. Any additional children are placed below the message. In Motif 1.1, only a single widget can be added as a child of a SelectionBox. This child is placed below the selection text and acts as a work area.

Traits

SelectionBox uses the XmQTactivatable and XmQTaccessTextual traits.

New Resources

SelectionBox defines the following resources:

Name	Class	Type	Default	Access
XmNapplyLabelString	XmCApplyLabelString	XmString	dynamic	CSG
XmNcancelLabelString	XmCCancelLabelString	XmString	dynamic	CSG
XmNchildPlacement	XmCChildPlacement	unsigned char	XmPLACE_ABOVE_SELECTION	CSG
XmNdialogType	XmCDialogType	unsigned char	dynamic	CG
XmNhelpLabelString	XmCHelpLabelString	XmString	dynamic	CSG
XmNlistItems	XmCItems	XmStringTable	NULL	CSG
XmNlistItemCount	XmCItemCount	int	0	CSG
XmNlistLabelString	XmCListLabelString	XmString	dynamic	CSG
XmNlistVisibleItemCount	XmCListVisibleItemCount	int	dynamic	CSG
XmNminimizeButtons	XmCMinimizeButtons	Boolean	False	CSG
XmNmustMatch	XmCMustMatch	Boolean	False	CSG
XmNokLabelString	XmCOkLabelString	XmString	dynamic	CSG
XmNselectionLabelString	XmCSelectionLabelString	XmString	dynamic	C
XmNtextAccelerators	XmCTextAccelerators	XtAccelerators	default	CSG
XmNtextColumns	XmCColumns	short	dynamic	CSG
XmNtextString	XmCTextString	XmString	dynamic	CSG

XmNapplyLabelString
The string that labels the **Apply** button. In Motif 1.2 and later, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Apply".

XmNcancelLabelString
The string that labels the **Cancel** button. In Motif 1.2 and later, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Cancel".

XmNchildPlacement
In Motif 1.2 and later, determines the placement of the work area child. Possible values:

```
XmPLACE_ABOVE_SELECTION     /* above the text area */
XmPLACE_BELOW_SELECTION     /* below the text area */
XmPLACE_TOP                 /* above the list area */
```

XmNdialogType
Determines which children of the SelectionBox widget will be initially created and managed. Possible values:

```
XmDIALOG_WORK_AREA      /* default, when parent isn't a DialogShell */
XmDIALOG_PROMPT         /* all children except list and label       */
XmDIALOG_SELECTION      /* default, when parent is a DialogShell    */
XmDIALOG_COMMAND        /* only list, selection label and text field */
XmDIALOG_FILE_SELECTION /* all standard children                    */
```

Note that in Release 1.1, Command and FileSelectionBox are separate widget classes, and they can no longer be created by setting XmNdialogType.

XmNhelpLabelString
The string that labels the **Help** button. In Motif 1.2 and later, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Help".

XmNlistItems
The items in the SelectionBox list. A call to XtGetValues() returns the actual list items (not a copy), so don't have your application free these items.

XmNlistItemCount
The number of items in the SelectionBox list.

XmNlistLabelString
The string that labels the SelectionBox list. The default string is NULL when the XmN-dialogType resource is set to XmDIALOG_PROMPT; otherwise, in Motif 1.2 and later, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Items".

XmNlistVisibleItemCount
The number of items that appear in the SelectionBox list. The default value depends on the height of the list. This resource has a value of 0 when the XmNdialogType resource is set to XmDIALOG_PROMPT, and otherwise has a default of 8.

XmNminimizeButtons
If False (default), all buttons are standardized to be as wide as the widest button and as high as the highest button. If True, buttons will keep their preferred size.

XmNmustMatch
If True, the selection that a user types in the text edit field must match an existing entry in the SelectionBox list. If False (default), the typed selection doesn't need to match a list entry. (When the user activates the Ok button, the widget calls one of two lists of callbacks: if this resource is True but the selections don't match, then the SelectionBox widget calls the callbacks specified by the XmNnoMatch-Callback resource; if this resource is False or if the selections do match, then the widget calls the callbacks specified by the XmNokCallback resource.)

XmNokLabelString
The string that labels the **Ok** button. In Motif 1.2 and later, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "OK"**.**

XmNselectionLabelString
The string that labels the text edit field. In Motif 1.2 and later, the default value is locale-dependent. In the C locale, and in Motif 1.1, the default value is "Selection".

XmNtextAccelerators
The translations to add to the SelectionBox's Text widget child. The default bindings allow the up and down keys to be used in selecting list items. This resource is meaningful only when the SelectionBox widget is using the default values in the XmN-accelerators resource.

XmNtextColumns
The number of columns in the Text widget.

XmNtextString
The text string that appears in the text edit selection field.

Callback Resources

SelectionBox defines the following callback resources:

Callback	Reason Constant
XmNapplyCallback	XmCR_APPLY
XmNcancelCallback	XmCR_CANCEL
XmNnoMatchCallback	XmCR_NO_MATCH

Callback	Reason Constant
XmNokCallback	XmCR_OK

XmNapplyCallback
List of callbacks that are called when the **Apply** button is activated.

XmNcancelCallback
List of callbacks that are called when the Cancel button is activated.

XmNnoMatchCallback
List of callbacks that are called when the user types a selection in the text area that does not match an item in the list.

XmNokCallback
List of callbacks that are called when the **OK** button is activated. If XmNmustMatch is True and the selection text does not match an item in the list, the XmNnoMatchCallback callbacks are called instead.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;     /* the reason that the callback was called */
    XEvent      *event;     /* event structure that triggered callback */
    XmString    value;      /* selection string that was either chosen */
                            /* from the SelectionBox list or typed in */
    int         length;     /* number of bytes of value            */
} XmSelectionBoxCallbackStruct;
```

Inherited Resources

SelectionBox inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. The default value of XmNborderWidth is reset to 0 by XmManager. BulletinBoard sets the values of XmNinitialFocus to the text entry area, XmN-defaultButton to the Cancel button, and resets the default XmNshadowThickness from 0 to 1 if the SelectionBox is a child of a DialogShell.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNlabelFontList	XmBulletinBoard
XmNallowOverlap	XmBulletinBoard	XmNlabelRenderTable	XmBulletinBoard
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNautoUnmanage	XmBulletinBoard	XmNmapCallback	XmBulletinBoard
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNmarginHeight	XmBulletinBoard

Resource	Inherited From	Resource	Inherited From
XmNborderColor	Core	XmNmarginWidth	XmBulletinBoard
XmNborderPixmap	Core	XmNnavigationType	XmManager
XmNborderWidth	Core	XmNnoResize	XmBulletinBoard
XmNbottomShadowColor	XmManager	XmNnumChildren	Composite
XmNbottomShadowPixmap	XmManager	XmNpopupHandlerCallback	XmManager
XmNbuttonFontList	XmBulletinBoard	XmNresizePolicy	XmBulletinBoard
XmNbuttonRenderTable	XmBulletinBoard	XmNscreen	Core
XmNcancelButton	XmBulletinBoard	XmNsensitive	Core
XmNchildren	Composite	XmNshadowThickness	XmManager
XmNcolormap	Core	XmNshadowType	XmBulletinBoard
XmNdefaultButton	XmBulletinBoard	XmNstringDirection	XmManager
XmNdefaultPosition	XmBulletinBoard	XmNtextFontList	XmBulletinBoard
XmNdepth	Core	XmNtextRenderTable	XmBulletinBoard
XmNdestroyCallback	Core	XmNtextTranslations	XmBulletinBoard
XmNdialogStyle	XmBulletinBoard	XmNtopShadowColor	XmManager
XmNdialogTitle	XmBulletinBoard	XmNtopShadowPixmap	XmManager
XmNfocusCallback	XmBulletinBoard	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNunmapCallback	XmBulletinBoard
XmNhighlightColor	XmManager	XmNuserData	XmManager
XmNhighlightPixmap	XmManager	XmNwidth	Core
XmNinitialFocus	XmManager	XmNx	Core
XmNinitialResourcesPersistent	Core	XmNy	Core
XmNinsertPosition	Composite		

Translations

The translations for SelectionBox are inherited from BulletinBoard.

Action Routines

SelectionBox defines the following action routines:

SelectionBoxUpOrDown(flag):

This action applies when the location cursor is within the item list. This action selects a list item from one of four possible positions and uses this item to replace the selection text. A flag value of 0, 1, 2, or 3 selects the previous, next, first, or last item, respectively.

These four action routines are respectively bound to KUp, KDown, KBeginData, and KEndData, which represent four of the default accelerators in the XmNtextAccelerators resource.

SelectionBoxRestore()
: Like SelectionBoxUpOrDown except that this action replaces the selection text with the current list item. This action clears the selection text if no list item is currently selected. This action routine is bound to KRestore, a default accelerator for XmNtextAccelerators.

Additional Behavior

SelectionBox has the following additional behavior:

MAny KCancel
: For a sensitive **Cancel** button, invokes the XmNactivateCallback callbacks.

KActivate
: For the button that has keyboard focus (or else the default button), invokes the callbacks in XmNactivateCallback. In a List or Text widget, this event calls the associated List or Text action before the associated SelectionBox action.

<Ok Button Activated>
: Invokes the XmNokCallback callback or the XmNnoMatchCallback if XmN-mustMatch is True and the text does not match an item in the list.

<Apply Button Activated>
: Invokes the XmNapplyCallback callbacks.

<Cancel Button Activated>
: Invokes the XmNcancelCallback callbacks.

<Help Button Activated>
: Invokes the XmNhelpCallback callbacks.

<MapWindow>
: Invokes the callbacks for XmNmapCallback if the parent is a DialogShell.

<UnmapWindow>
: Invokes the callbacks for XmNunmapCallback if the parent is a DialogShell.

See Also

XmCreateObject(1), XmSelectionBoxGetChild(1), Composite(2), Constraint(2), Core(2), XmBulletinBoard(2), XmManager(2), XmPromptDialog(2), XmSelectionDialog(2).

Name

XmSelectionDialog –an unmanaged SelectionBox as a child of a Dialog Shell.

Synopsis

Public Header:
<Xm/SelectionB.h>

Instantiation:
widget = `XmCreateSelectionDialog` (parent, name,...)

Functions/Macros:
`XmCreateSelectionBox`(), `XmCreateSelectionDialog`(), `XmSelectionBoxGetChild`()

Description

An XmSelectionDialog is a compound object created by a call to XmCreateSelectionDialog() that an application can use to allow a user to make a selection from a dialog box. A SelectionDialog consists of a DialogShell with an unmanaged SelectionBox widget as its child. The SelectionBox resource XmNdialogType is set to XmDIALOG_SELECTION.

A SelectionDialog displays a scrollable list of alternatives from which the user can choose items. A SelectionDialog also contains a text field in which the user can edit a selection, labels for the text field and for the scrollable list, and four buttons. In Motif 1.2 and later, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Apply**, **Cancel**, and **Help** by default.

Default Resource Values

A SelectionDialog sets the following default values for SelectionBox resources:

Name	Default
XmNdialogType	XmDIALOG_SELECTION

Widget Hierarchy

When a SelectionDialog is created with a specified name, the DialogShell is named name*_popup* and the SelectionBox is called *name***.**

See Also

`XmCreateObject`(1), `XmSelectionBoxGetChild`(1), `XmDialogShell`(2), `XmSelectionBox`(2).

Name

XmSeparator widget class – a widget that draws a line to separate other widgets visually.

Synopsis

Public Header:
<Xm/Separator.h>

Class Name:
XmSeparator

Class Hierarchy:
Core → XmPrimitive → XmSeparator

Class Pointer:
xmSeparatorWidgetClass

Instantiation:
widget = `XmCreateSeparator` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmSeparatorWidgetClass,...)

Functions/Macros:
`XmCreateSeparator`(), `XmIsSeparator`()

Description

A Separator is a widget that draws a horizontal or vertical line between components in an application. Several line styles are available for the Separator. A pixmap separator can also be made by specifying a pixmap for the Core resource XmNbackgroundPixmap and then setting XmNseparatorType to XmNO_LINE.

Traits

Separator holds the XmQTmenuSavvy trait, which is inherited by any derived classes.

New Resources

Separator defines the following resources:

Name	Class	Type	Default	Access
XmNmargin	XmCMargin	Dimension	0	CSG
XmNorientation	XmCOrientation	unsigned char	XmHORIZONTAL	CSG
XmNseparatorType	XmCSeparatorType	unsigned char	XmSHADOW_ETCHED_IN	CSG

XmNmargin
The spacing on either end of the Separator. This would be the left and right margins for a horizontally drawn Separator and the top and bottom margins for a vertically drawn Separator.

XmNorientation
The direction in which to display the Separator. Possible values:

```
XmVERTICAL          /* top-to-bottom creation */
XmHORIZONTAL        /* left-to-right creation  */
```

XmNseparatorType
The line style in which to draw the Separator. Possible values:

XmNO_LINE
XmSINGLE_DASHED_LINE
XmSHADOW_ETCHED_IN
XmSINGLE_LINE
XmDOUBLE_DASHED_LINE
XmSHADOW_ETCHED_OUT
XmDOUBLE_LINE

Inherited Resources

Separator inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. Separator sets the default values of XmNhighlightThickness to 0 XmNtraversalOn to False. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNhighlightThickness	XmPrimitive
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNlayoutDirection	XmPrimitive
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnavigationType	XmPrimitive
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmPrimitive
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmPrimitive	XmNsensitive	Core
XmNbottomShadowPixmap	XmPrimitive	XmNshadowThickness	XmPrimitive
XmNcolormap	Core	XmNtoolTipString	XmPrimitive
XmNconvertCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNdepth	Core	XmNtopShadowPixmap	XmPrimitive
XmNdestroyCallback	Core	XmNtranslations	Core

Resource	Inherited From	Resource	Inherited From
XmNforeground	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNheight	Core	XmNunitType	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNuserData	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNwidth	Core
XmNhighlightOnEnter	XmPrimitive	XmNx	Core
XmNhighlightPixmap	XmPrimitive	XmNy	Core

See Also

`XmCreateObject`(1), `Core`(2), `XmPrimitive`(2).

Name

XmSeparatorGadget widget class –a gadget that draws a line to separate other widgets visually.

Synopsis

Public Header:
<Xm/SeparatoG.h>

Class Name:
XmSeparatorGadget

Class Hierarchy:
Object → RectObj → XmGadget → XmSeparatorGadget

Class Pointer:
xmSeparatorGadgetClass

Instantiation:
widget = `XmCreateSeparatorGadget` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmSeparatorGadgetClass,...)

Functions/Macros:
`XmCreateSeparatorGadget`(), `XmIsSeparatorGadget`()

Description

SeparatorGadget is the gadget variant of Separator. SeparatorGadget's new resources are the same as those for Separator.

Traits

SeparatorGadget holds the XmQTcareParentVisual, XmQTaccessColors, and XmQTmenuSavvy traits, which are inherited by any derived class.

Inherited Resources

SeparatorGadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. SeparatorGadget sets the default values of XmNhighlightThickness to 0 and XmNtraversalOn to False. The default value of XmNborderWidth is reset to 0 by Gadget.

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	RectObj	XmNlayoutDirection	XmGadget
XmNbackground	XmGadget	XmNnavigationType	XmGadget
XmNbackgroundPixmap	XmGadget	XmNsensitive	RectObj
XmNbottomShadowColor	XmGadget	XmNshadowThickness	XmGadget
XmNbottomShadowPixmap	XmGadget	XmNtoolTipString	XmGadget
XmNborderWidth	RectObj	XmNtopShadowColor	XmGadget

Resource	Inherited From	Resource	Inherited From
XmNdestroyCallback	Object	XmNtopShadowPixmap	XmGadget
XmNforeground	XmGadget	XmNtraversalOn	XmGadget
XmNheight	RectObj	XmNunitType	XmGadget
XmNhelpCallback	XmGadget	XmNuserData	XmGadget
XmNhighlightColor	XmGadget	XmNwidth	RectObj
XmNhighlightOnEnter	XmGadget	XmNx	RectObj
XmNhighlightPixmap	XmGadget	XmNy	RectObj
XmNhighlightThickness	XmGadget		

See Also

`XmCreateObject`(1), `Object`(2), `RectObj`(2), `XmGadget`(2), `XmSeparator`(2).

Name

XmSimpleSpinBox widget class –a widget for cycling through a set of choices

Synopsis

Public Header:
<Xm/SSpinB.h>

Class Name:
XmSimpleSpinBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmSpinBox → XmSimpleSpinBox

Class Pointer:
xmSimpleSpinBoxWidgetClass

Instantiation:
widget = `XmCreateSimpleSpinBox` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmSimpleSpinBoxWidgetClass,...)

Functions/Macros:
`XmSimpleSpinBoxAddItem`(), `XmSimpleSpinBoxDeletePos`(), `XmSimpleSpinBoxSetItem`(), `XmCreateSimpleSpinBox`()

Availability

Motif 2.1 and later.

Description

A subclass of SpinBox which allows the user to increment or decrement the value in a TextField, or to cycle through a set of values. The TextField is created internally by the widget.

The SimpleSpinBox may not be subclassed.

New Resources

SimpleSpinBox defines the following resources:

Name	Class	Type	Default	Access
XmNarrowSensitivity	XmCArrowSensitivity	unsigned char	XmARROWS_SENSITIVE	CSG
XmNcolumns	XmCColumns	short	20	CSG
XmNdecimalPoints	XmCDecimalPoints	short	0	CSG
XmNeditable	XmCEditable	Boolean	True	CSG

Name	Class	Type	Default	Access
XmNincrementValue	XmCIncrementValue	int	1	CSG
XmNmaximumValue	XmCMaximumValue	int	10	CSG
XmNminimumValue	XmCMinimumValue	int	0	CSG
XmNnumValues	XmCNumValues	int	0	CSG
XmNposition	XmCPosition	int	0	CSG
XmNpositionType	XmCPositionType	unsigned char	XmPOSITION_VALUE	CG
XmNspinBoxChildType	XmCSpinBoxChildType	unsigned char	XmString	CSG
XmNtextField	XmCTextField	Widget	dynamic	G
XmNvalues	XmCValues	XmStringTable	NULL	CSG
XmNwrap	XmCWrap	Boolean	True	CSG

XmNarrowSensitivity
Specifies the sensitivity of the arrows within the SimpleSpinBox. Possible values:

```
XmARROWS_SENSITIVE              /* both arrows sensitive          */
XmARROWS_DECREMENT_SENSITIVE
                                /* increment arrowbutton insensitive */
XmARROWS_INCREMENT_SENSITIVE
                                /* decrement arrowbutton insensitive */
XmARROWS_INSENSITIVE            /* both arrows insensitive        */
```

XmNcolumns
Specifies the number of columns in the TextField.

XmNdecimalPoints
A positive integer that determines how the TextField's value will be displayed. The decimal point in the TextField's value gets shifted to the right, and this resource specifies the number of decimal places to shift. For example, if the TextField's value is 5678, then setting the XmNdecimalPoints[1] resource to 2 causes the widget to display the value as 56.78. The resource has no effect if XmNspinBoxChildType is not XmNUMERIC.

XmNeditable
Specifies whether the TextField accepts user input.

XmNincrementValue
Specifies the amount to increment the XmNposition resource. The resource has no effect if XmNspinBoxChildType is not XmNUMERIC.

1.Erroneously given as XmdecimalPoints in 2nd edition.

XmNmaximumValue
Specifies the largest value. The resource has no effect if XmNspinBoxChildType is not XmNUMERIC.

XmNminimumValue
Specifies the smallest value. The resource has no effect if XmNspinBox-ChildType is not XmNUMERIC.

XmNnumValues
Specifies the number of items in the list determined by the XmNvalues resource. The resource has no effect if XmNspinBoxChildType is not XmSTRING.

XmNposition
Depends upon the value of the XmNpositionType and XmNspinBoxChildType resources, and is used to calculate the current value of the SimpleSpinBox.

If XmNspinBoxChildType is XmSTRING, the position resource is used simply as an index into the XmNvalues array.

If XmNspinBoxChildType is XmNUMERIC and XmNpositionType is XmPOSITION_VALUE, the position resource is used directly for the actual value to display. The position value is bounded by XmNminimumValue and XmNmaximumValue. When XmNpositionType is XmPOSITION_INDEX, the position is interpreted as an index into a set of values, bounded by the XmNminimumValue and XmNmaximumValue resource. A number is in the set depending upon the XmNincrementValue resource: position zero corresponds to the value XmNminimumValue, position n is the value given by:

XmNminimumValue + (n * XmNincrementValue)

XmNpositionType
Specifies how the XmNposition resource is to be interpreted. Possible values:

```
XmPOSITION_INDEX    /* position is an index into an array */
XmPOSITION_VALUE    /* position is a direct value        */
```

XmNspinBoxChildType
Specifies the type of data to be displayed. Possible values:

```
XmNUMERIC       /* value is defined by maximum, minimum,    */
                /* increment resources */
XmSTRING        /* value is defined by the values array                */
```

XmNtextField
Specifies the text field created by the SimpleSpinBox to display the current value.

XmNvalues
Specifies the array of compound strings forming the validset of items for the SimpleSpinBox. Only has effect if XmNspinBoxChildType is XmSTRING.

XmNwrap
Specifies whether the SpinBox wraps around the set of values.

Inherited Resources

SimpleSpinBox inherits the resources shown below. The resources are listed alphabetically, along with the superclass that defines them. SimpleSpinBox resets the default value of XmNshadowThickness to 1.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNarrowLayout	XmSpinBox	XmNmappedWhenManaged	Core
XmNarrowOrientation	XmSpinBox	XmNmarginHeight	XmSpinBox
XmNarrowSize	XmSpinBox	XmNmarginWidth	XmSpinBox
XmNbackground	Core	XmNmodifyVerifyCallback	XmSpinBox
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNrepeatDelat	XmSpinBox
XmNbottomShadowColor	XmManager	XmNscreen	Core
XmNbottomShadowPixmap	XmManager	XmNsensitive	Core
XmNchildren	Composite	XmNshadowThickness	XmManager
XmNcolormap	Core	XmNspacing	XmSpinBox
XmNdefaultArrowSensitivity	XmSpinBox	XmNstringDirection	XmManager
XmNdepth	Core	XmNtopShadowColor	XmManager
XmNdestroyCallback	Core	XmNtopShadowPixmap	XmManager
XmNdetailShadowThickness	XmSpinBox	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNvalueChangedCallback	XmSpinBox
XmNhighlightPixmap	XmManager	XmNwidth	Core
XmNinitialDelay	XmSpinBox	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Widget Hierarchy

When a SimpleSpinBox is created with a specified name, the automatically created TextField child is named name*_TF*.

Translations

The translations for SimpleSpinBox are those of SpinBox.

See Also

`XmSimpleSpinBoxAddItem`(1), `XmSimpleSpinBoxDeletePos`(1), `XmSimpleSpinBoxSetItem`(1), `XmCreateSimpleSpinBox`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmManager`(2), `XmSpinBox`(2).

Name

XmSpinBox widget class – a composite widget which controls cycling through a set of choices

Synopsis

Public Header:
<Xm/SpinB.h>

Class Name:
XmSpinBox

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmSpinBox

Class Pointer:
xmSpinBoxWidgetClass

Instantiation:
widget = `XmCreateSpinBox` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmSpinBoxWidgetClass,...)

Functions/Macros:
`XmSpinBoxValidatePosition`(), `XmCreateSpinBox`()

Availability

Motif 2.0 and later.

Description

SpinBox is a manager which allows the user to cycle through sets of choices. At the minimum, the widget contains a single Text or TextField, which is associated with a group of values. A pair of ArrowButtons are provided which, when pressed, insert the next or previous group item into the Text.

SpinBox can control multiple traversable children, each possessing their own set of values. The ArrowButtons cycle the values of the child with the current focus.

The values associated with any textual child of the SpinBox are specified through constraint resources. Logically, a textual child is considered to be either numeric or string based, as specified by the XmNspinBoxChildType resource. If XmNUMERIC, a set of constraints control the current value, and place an upper and lower bound upon the range through which the value may rotate. If XmSTRING, an array of compound strings is specified for the child, and the ArrowButtons cycle through the set of values by incrementing or decrementing an index into the array.

The location of the ArrowButtons relative to the textual children is controlled through the XmNarrowLayout resource, although this is affected by any specified XmNlayoutDirection. The arrows automatically created by the SpinBox are drawn, and not real widgets.

Traits

SpinBox holds the XmQTnavigator trait, which is inherited by any derived classes, and uses the XmQTaccessTextual trait.

New Resources

SpinBox defines the following resources:

Name	Class	Type	Default	Access
XmNarrowLayout	XmCArrowLayout	unsigned char	XmARROWS_END	CSG
XmNarrowOrientation	XmCArrowOrientation	unsigned char	XmARROWS_VERTICAL	CSG
XmNarrowSize	XmCArrowSize	Dimension	16	CSG
XmNdefaultArrowSensitivity	XmCDefaultArrowSensitivity	unsigned char	XmARROWS_SENSITIVE	CSG
XmNdetailShadowThickness	XmCDetailShadowThickness	Dimension	dynamic	CSG
XmNinitialDelay	XmCInitialDelay	unsigned int	250	CSG
XmNmarginHeight	XmCMarginHeight	Dimension	2	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	2	CSG
XmNrepeatDelat	XmCRepeatDelay	unsigned int	200	CSG
XmNspacing	XmCSpacing	Dimension	2	CSG

XmNarrowLayout

Specifies the location of the drawn arrows relative to the textual children. Possible values:

```
XmARROWS_BEGINNING          /* both arrows placed vertically   before text */
XmARROWS_END                /* both arrows placed vertically   after text    */
XmARROWS_FLAT_BEGINNING     /* both arrows placed horizontally before text*/
XmARROWS_FLAT_END           /* both arrows placed horizontally after text */
XmARROWS_SPLIT              /* one arrow placed at each end                */
```

The interpretation of beginning and end depends upon the XmNlayoutDirection resource inherited from XmManager. This also affects whether it is the increment or decrement arrow which is to the left or right. If the layout direction is XmLEFT_TO_RIGHT, the decrement arrow is to the left, and both XmARROWS_BEGINNING and XmARROWS_FLAT_BEGINNING place both arrows on the left of the Text. If the direction is XmRIGHT_TO_LEFT, it is

the increment arrow which is to the left, XmARROWS_END and XmARROWS_FLAT_END which place both arrows on the left of the Text.

XmNarrowOrientation
In Motif 2.1, specifies whether arrows point vertically or horizontally. If the orientation is XmARROWS_VERTICAL, the decrement arrow points downwards, and the increment arrow points upwards. If orientation is XmARROWS_HORIZONTAL, the decrement arrow points to the left, and the increment arrow points to the right. This is reversed if the XmNlayoutDirection is XmRIGHT_TO_LEFT.

Note that this is not the same as the XmNarrowLayout resource, which positions the components of the SpinBox relative to each other, whereas XmNarrowOrientation rotates an arrow within its given position.

XmNarrowSize
Specifies the width (and height) of the drawn arrows, measured in pixels.

XmNdefaultArrowSensitivity
Specifies the default sensitivity of the drawn arrows to user input. The resource is overridden by the XmNarrowSensitivity constraint resource of the textual child which has the focus. Possible values:

```
XmARROWS_DECREMENT_SENSITIVE
                                /* only the decrement arrow accepts input */
XmARROWS_INCREMENT_SENSITIVE
                                /* only the increment arrow accepts input */
XmARROWS_INSENSITIVE    /* both arrows are insensitive          */
XmARROWS_SENSITIVE      /* both arrows are sensitive            */
```

XmNdetailShadowThickness
Specifies the thickness of the shadow used for drawing the arrow shapes. Values of 0, 1, or 2 are implemented. In Motif 2.0, the default value is 2. In Motif 2.1, the default value depends upon the XmDisplay XmNenableThinThickness resource: if True the default is 1, otherwise 2.

XmNinitialDelay
Specifies the time interval in milliseconds (after pressing the mouse) which is to elapse before the SpinBox triggers automatic spinning. If the value is zero, the value defaults to that specified by the XmNrepeatDelay resource.

XmNmarginWidth
Specifies the space between the left edge of the SpinBox and the leftmost child, and the space between the right edge of the SpinBox and the rightmost child.

XmNmarginHeight
Specifies the space between the top edge of the SpinBox and the topmost child, and the space between the bottom edge of the SpinBox and the bottom child.

XmNrepeatDelay
Specifies the time interval in milliseconds (when holding down the mouse) which is to elapse before the SpinBox triggers automatic spinning: with the mouse held down, the SpinBox repeatedly spins until such time as the mouse is released. If the value is zero, automatic spinning is disabled.

XmNspacing
Specifies the horizontal and vertical spacing between items in the SpinBox.

New Constraint Resources

SpinBox defines the following constraint resources for its children:

Name	Class	Type	Default	Access
XmNarrowSensitivity	XmCArrowSensitivity	unsigned char	XmARROWS_DEFAULT_SENSITIVITY	CSG
XmNdecimalPoints	XmCDecimalPoints	short	0	CSG
XmNincrementValue	XmCIncrementValue	int	1	CSG
XmNmaximumValue	XmCMaximumValue	int	10	CSG
XmNminimumValue	XmCMinimumValue	int	0	CSG
XmNnumValues	XmCNumValues	int	0	CSG
XmNposition	XmCPosition	int	0	CSG
XmNpositionType	XmCPositionType	unsigned char	XmPOSITION_VALUE	CSG
XmNspinBoxChildType	XmCSpinBoxChildType	unsigned char	XmString	CSG
XmNvalues	XmCValues	XmStringTable	NULL	CSG
XmNwrap	XmCWrap	Boolean	True	CSG

XmNarrowSensitivity
Specifies the sensitivity of the arrowbuttons. Possible values:

```
XmARROWS_DEFAULT_SENSITIVITY
                        /* inherit XmNdefaultArrowSensitivity   */
XmARROWS_DECREMENT_SENSITIVE
                        /* only decrement arrow accepts input   */
XmARROWS_INCREMENT_SENSITIVE
                        /* only increment arrow accepts input   */
XmARROWS_INSENSITIVE    /* both arrows are insensitive          */
XmARROWS_SENSITIVE      /* both arrows are sensitive            */
```

XmNdecimalPoints
Specifies the number of decimal places used when displaying numeric values. The value is zero padded where necessary.

XmNincrementValue
Specifies the amount to increment or decrement the numeric value in the SpinBox text when the increment or decrement arrow is pressed. The resource is only used when the SpinBox type is XmNUMERIC.

XmNmaximumValue
Specifies the greatest value allowed in an XmNUMERIC SpinBox.

XmNminimumValue
Specifies the smallest value allowed in an XmNUMERIC SpinBox.

XmNnumValues
Specifies the number of values in the XmNvalues array. The resource is only used when the SpinBox type is XmSTRING.

XmNposition
The interpretation of this resource depends upon the value of the XmNpositionType and XmNspinBoxChildType resources.

When XmNpositionType is XmPOSITION_INDEX, the position is interpreted as an index into an array of values. If XmNspinBoxChildType is XmSTRING, these are defined by the XmNvalues resource. If the child type is XmNUMERIC, then the array of values is a set of numbers bounded by the XmNminimumValue and XmNmaximumValue resource. A number is in the set depending upon the XmNincrementValue resource: position zero corresponds to the value XmNminimumValue, position n is the value given by:

XmNminimumValue + (n * XmNincrementValue)

When the XmNpositionType is XmPOSITION_VALUE and the XmNspinBoxChildType is XmNUMERIC, the position is the actual value to display, and is bounded by XmNminimumValue and XmNmaximumValue. If XmNspinBoxChildType is XmSTRING, position remains an index into the XmNvalues resource array.

XmNpositionType
Specifies how the XmNposition resource is to be interpreted. Possible values:

```
XmPOSITION_INDEX   /* position is an index into an array      */
XmPOSITION_VALUE   /* position is a direct value              */
```

XmNspinBoxChildType
Specifies the type of data to be displayed. Possible values:

```
XmNUMERIC          /* choices defined by maximum,            */
                   /* minimum, increment values              */
XmSTRING           /* choices defined by the values array    */
```

XmNvalues
Specifies the array of compound strings forming the set of items for the SimpleSpinBox. The resource only has effect if XmNspinBoxChildType is XmSTRING.

XmNwrap
Specifies whether the SpinBox wraps around the set of values. If the current position is at the maximum value, the increment arrow is pressed, and XmNwrap is True, the requested position becomes the minimum value. Similarly if the current position is at the minimum, and the decrement arrow is pressed with wrap enabled, the requested position is at the maximum. If wrap is False in the given circumstances, the bell is rung and the selection is unchanged.

Callback Resources

SpinBox defines the following callback resources:

Callback	Reason Constant
XmNmodifyVerifyCallback	XmCR_SPIN_FIRST XmCR_SPIN_LAST XmCR_SPIN_NEXT XmCR_SPIN_PRIOR
XmNvalueChangedCallback	XmCR_OK XmCR_SPIN_FIRST XmCR_SPIN_LAST XmCR_SPIN_NEXT XmCR_SPIN_PRIOR

XmNmodifyVerifyCallback
List of callbacks called before the SpinBox position is changed.

XmNvalueChangedCallback
List of callbacks called after the SpinBox position is changed.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;           /* the reason that the callback was called */
    XEvent      *event;           /* points to event that triggered callback */
    Widget      widget;           /* the textual child affected by callback  */
    Boolean     doit;             /* whether to perform the changes          */
    int         position;         /* specifies the index of the next value   */
    XmString    value;            /* specifies the next value                */
    Boolean     crossed_boundary; /* whether the SpinBox has wrapped         */
} XmSpinBoxCallbackStruct;
```

reason indicates why the callback is invoked. If *reason* is XmCR_SPIN_FIRST, the SpinBox position is either XmNminimum, or the index of the first item in the XmNvalues array, depending upon whether the SpinBox type is XmNUMERIC or XmSTRING respectively. If *reason* is XmCR_SPIN_LAST, the position is either XmNmaximum, or the index of the last item in the XmNvalues array. If *reason* is XmCR_SPIN_NEXT, the increment arrow is armed. If *reason* is XmCR_SPIN_PRIOR, the decrement arrow is armed. If *reason* is XmCR_OK, an arrow is disarmed.

widget is the ID of the textual component affected by the callback. This is the text which has the current focus.

doit is a flag indicating whether the action is to be performed. The default value is True, although a programmer may set the value False for whatever reason to prevent the item associated with position and value being displayed at the current instant. *doit* is only relevant to XmNmodifyVerifyCallback callbacks.

position is equivalent to the XmNposition resource, and specifies the next position to display. An XmNmodifyVerifyCallback procedure may alter the value in order to force the SpinBox to display a particular item.

value specifies the new item to be displayed in widget. *value* is a temporary compound string which is freed after callback procedures are finished. The programmer should copy *value* if this is required outside of the callback procedures.

crossed_boundary[1] specifies whether the SpinBox has crossed the upper or lower bound as specified by XmNminimum and XmNmaximum, or the first and last compound string in the XmNvalues array.

Inherited Resources

SpinBox inherits the resources shown below. The resources are listed alphabetically, along with the superclass that defines them.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNinsertPosition	Composite
XmNancestorSensitive	Core	XmNlayoutDirection	XmManager
XmNbackground	Core	XmNmappedWhenManaged	Core
XmNbackgroundPixmap	Core	XmNnavigationType	XmManager
XmNborderColor	Core	XmNnumChildren	Composite
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmManager
XmNborderWidth	Core	XmNscreen	Core

1.Erroneously given as crossing_boundary in 2nd edition.

Resource	Inherited From	Resource	Inherited From
XmNbottomShadowColor	XmManager	XmNsensitive	Core
XmNbottomShadowPixmap	XmManager	XmNshadowThickness	XmManager
XmNchildren	Composite	XmNstringDirection	XmManager
XmNcolormap	Core	XmNtopShadowColor	XmManager
XmNdepth	Core	XmNtopShadowPixmap	XmManager
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmManager	XmNtraversalOn	XmManager
XmNheight	Core	XmNunitType	XmManager
XmNhelpCallback	XmManager	XmNuserData	XmManager
XmNhighlightColor	XmManager	XmNwidth	Core
XmNhighlightPixmap	XmManager	XmNx	Core
XmNinitialFocus	XmManager	XmNy	Core
XmNinitialResourcesPersistent	Core		

Translations

The translations for SpinBox include those of XmManager.

Event	Action
BSelect Press	SpinBArm()
BSelect Release	SpinBDisarm()
<EnterWindow>	SpinBEnter()
<LeaveWindow)	SpinBLeave()
KUp	SpinBPrior()
KDown	SpinBNext()
KLeft	SpinBLeft()
KRight	SpinBRight()
KBeginData	SpinBFirst()
KEndData	SpinBLast()

Action Routines

SpinBox defines the following action routines:

SpinBArm()

Draws the arrow shape under the pointer in armed form. If the XmNinitialDelay resource is specified, a timer is initialized to the value of the initial delay period. If the mouse is not released before the timer expires, the SpinBox starts automatically selecting successive items in the SpinBox.XmNmodifyVerifyCallback procedures are called with the position element adjusted to the selected value, depending upon the type of arrow, and subsequently XmNvalueChangedCallback procedures are called if the doit element of the XmSpinBoxCallbackStruct is still True after the modify verify callbacks are finished. The position element of the structure is used to determine the value which is inserted into the textual child of the SpinBox which has the focus.

SpinBDisarm()

Draws the arrow shape in disarmed form. If any period specified by the XmNrepeatDelay or XmNinitialDelay periods have not expired, XmNmodifyVerifyCallback procedures are called with the position element adjusted to the required value, and subsequently XmNvalueChangedCallback procedures are called if the doit element of the XmSpinBoxCallbackStruct is still True. The item as reflected by the final value of the position element of the structure is inserted into the current traversable text.

SpinBPrior()

Draws the decrement arrow in armed form, and invokes any XmNmodifyVerifyCallback procedures, with the position element of the XmSpinBoxCallbackStruct suitably decremented, taking into account possible wrapping. Thereafter, if the doit element is still True, any XmNvalueChangedCallback procedures are invoked. The item as reflected by the final value of the position element of the structure is inserted into the current traversable text.

SpinBNext()

Draws the increment arrow in armed form, and invokes any XmNmodifyVerifyCallback procedures, with the position element of the XmSpinBoxCallbackStruct suitably incremented, taking into account possible wrapping. Thereafter, if the doit element is still True, any XmNvalueChangedCallback procedures are invoked. The item as reflected by the final value of the position element of the structure is inserted into the current traversable text.

SpinBLeft()

If the XmNlayoutDirection is XmLEFT_TO_RIGHT, invokes the SpinBPrior() action, otherwise invokes SpinBNext().

SpinBRight()

If the XmNlayoutDirection is XmLEFT_TO_RIGHT, invokes the SpinBNext() action, otherwise invokes SpinBPrior().

SpinBFirst()

XmNmodifyVerifyCallback procedures are invoked with the position element of the XmSpinBoxCallbackStruct set to zero. Thereafter, if the doit element is still True, any XmNvalueChangedCallback procedures are invoked. The item as reflected by the final value of the position element of the structure is inserted into the current traversable text.

SpinBLast()

XmNmodifyVerifyCallback procedures are invoked with the position element of the XmSpinBoxCallbackStruct set to the last position. Thereafter, if the doit element is still True, any XmNvalueChangedCallback procedures are invoked. The item as reflected by the final value of the position element of the structure is inserted into the current traversable text.

SpinBEnter()

If the containing shell has a focus policy of XmPOINTER, draws the highlight border around the child traversable text widget which has the focus.

SpinBLeave()

If the containing shell has a focus policy of XmPOINTER, erases the highlight border around the child traversable text widget which has the focus.

See Also

`XmSpinBoxValidatePosition`(1), `XmCreateObject`(1), `Composite`(2), `Constraint`(2), `Core`(2), `XmManager`(2).

Name

XmTabStack – The TabStack widget class

Synopsis

Public Header:
<Xm/TabStack.h>

Class Name:
XmTabStack

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmTabStack

Class Pointer:
xmTabStackWidgetClass

Instantiation:

```
widget = XmCreateTabStack(parent, name,...)
or
widget = XtCreateWidget(name, xmTabStackWidget-
Class,...)
```

Functions/Macros:

```
XmCreateTabStack(), XmTabStackGetSlectedTab(), XmTab-
StackSelectTab(), XmTabStackIndexToWidget(), XmTab-
StackXYToWidget().
```

Availability

OpenMotif 2.2 and later. (Contributed Widget).

Description

The XmTabStack widget manages a group of widgets such that only one widget in the group is visible at a time. Each child is associated with a "tab" that displays a text label and/or a pixmap. By selecting the "tab" the user interactively determines which child is displayed. This widget exhibits behavior similar to the Microsoft Windows(TM) Tab Control.

The tabs can be configured to appear above, below, to the right, and to the left of a work area with the text oriented in any of the four cardinal directions.

The TabStack allows the user to select, either by pointer or keyboard traversal, tabs. When a tab is selected it changes appearance so that it appears to be raised above the other tabs. When a tab is selected the child associated with the tab is made visible. One tab is selected at all times.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNfontList	XmCFontList	XmFontList	Dynamic	CSG
XmNhighlightThickness	XmCHighlightThickness	Dimension	2	CSG
XmNstackedEffect	XmCStackedEffect	Boolean	True	CSG
XmNtabAutoSelect	XmCTabAutoSelect	Boolean	True	CG
XmNtabCornerPercent	XmCTabCornerPercent	int	40	CSG
XmNtabLabelSpacing	XmCTabLabelSpacing	Dimension	2	CSG
XmNtabMarginHeight	XmCTabMarginHeight	Dimension	3	CSG
XmNtabMarginWidth	XmCTabMarginWidth	Dimension	3	CSG
XmNtabMode	XmCTabMode	int	XmTABS_BASIC	CSG
XmNtabOffset	XmCTabOffset	Dimension	10	CSG
XmNtabOrientation	XmCTabOrientation	int	Dynamic	CSG
XmNtabSelectColor	XmCTabSelectColor	Pixel	Dynamic	CSG
XmNtabSelectedCallback	XmCCallback	XtCallbackList	NULL	CS
XmNtabSelectPixmap	XmCTabSelectPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNtabSide	XmCTabSide	int	XmTABS_ON_TOP	CSG
XmNtabStyle	XmCTabStyle	int	XmTABS_BEVELED	CSG
XmNuniformTabSize	XmCUniformTabSize	Boolean	True	CSG
XmNuseImageCache	XmCUseImageCache	Boolean	True	CSG

XmNfontList

Specifies the XmFontList to use when drawing the label strings for the tabs.

XmNhighlightThickness

Specifies the thickness of the rectangle drawn around the label string and label pixmap of the tab with keyboard traversal.

XmNstackedEffect
: Specifies if the visuals should depict a stack of folders, True, or if the XmTabStack should use all available space for its children, False.

XmNtabAutoSelect
: Specifies if a tab is automatically selected when it receives keyboard traversal.

XmNtabCornerPercent
: Specifies the percent of the font height that should be used for the corner visual.

XmNtabLabelSpacing
: Specifies the amount of space to leave between a text label and a pixmap in the tab area.

XmNtabMarginHeight
: Specifies the vertical border that is placed around the label area of a tab.

XmNtabMarginWidth
: Specifies the horizontal border that is placed around the label area of a tab.

XmNtabMode
: Specifies the mode in which the XmTabStack distributes the tabs. Valid values for this resource include:

  XmTABS_BASIC
  : Distributes the tabs in either a vertical or horizontal row and clips the tabs if there is not enough room to display all the tabs.

  XmTABS_STACKED
  : Distributes the tabs in either a vertical or horizontal row. If there is not enough room to display all the tabs additional rows are added. When a tab is selected, its row is moved next to the children in the stack.

  XmTABS_STACKED_STATIC
  : Distributes the tabs in either a vertical or horizontal row. If there is not enough room to display all the tabs additional rows are added. The positions of rows are not changed when tabs are selected.

XmNtabOffset
: Specifies the amount of indentation used to stagger the tab rows when displaying tabs in either the **XmTABS_STACKED** or **XmTABS_STACKED_STATIC** mode.

XmNtabOrientation
: Specifies the orientation of the tab, and the rotation factor of the tab label. Valid values for this resource include:

  XmTAB_ORIENTATION_DYNAMIC
  : Specifies that the orientation of the tabs should be calculated dynamically based on the **XmNtabSide** resource.

  XmTABS_LEFT_TO_RIGHT
  : Specifies that the text appears at the default rotation.

  XmTABS_RIGHT_TO_LEFT
  : Specifies that the text appears upside down.

XmTABS_TOP_TO_BOTTOM
Specifies that the text should be rotated to the vertical position with the first character drawn at the lowest y position and the bottom of the text faces the lowest x position.
XmTABS_BOTTOM_TO_TOP
Specifies that the text should be rotated to the vertical, position with the first character drawn at the highest y position and the bottom of the text faces the highest x position.

XmNtabSelectColor
Specifies the color of the selected tab.

XmNtabSelectedCallback
Specifies the list of callbacks to call when a child becomes the selected tab.

XmNtabSelectPixmap
Specifies the pixmap of the selected tab.

XmNtabSide
Specifies the location of the tab with respect to the children of the XmTabStack. Valid values for this resource include:
XmTABS_ON_TOP
Specifies that the tabs should be placed above the children.
XmTABS_ON_BOTTOM
Specifies that the tabs should be placed below the children.
XmTABS_ON_RIGHT
Specifies that the tabs should be placed to the right of the children.
XmTABS_ON_LEFT
Specifies that the tabs should be placed to the left of the children.

XmNtabStyle
Specifies the appearance of the tabs associated with the children of the XmTabStack. Valid values for this resource include:
XmTABS_BEVELED
Draws the corners of the tabs as an angled line.
XmTABS_ROUNDED
Draws the corners of the tabs as a quarter of a circle.
XmTABS_SQUARED
Draws the tabs as rectangles.

XmNuniformTabSize
Determines if all tabs should have a uniform major dimension, where the major dimension is width if the tab orientation is **XmTABS_LEFT_TO_RIGHT** or **XmTABS_RIGHT_TO_LEFT** or height if the tab orientation is **XmTABS_TOP_TO_BOTTOM** or **XmTABS_BOTTOM_ TO_TOP**.

XmNuseImageCache
Determines if the Xmmages used for rotating text and pixmaps should be cached. This increases performance but uses up more memory.

Inherited Resources

Icon Box inherits behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass

Resource	Inherited from	Resource	Inherited from
XmNentryLabelAlignment	XmBulletinBoard	XmNuserData	XmManager
XmNentryLabelFontList	XmBulletinBoard	XmNchildren	Composite
XmNentryLabelPixmap	XmBulletinBoard	XmNinsertPosition	Composite
XmNentryLabelString	XmBulletinBoard	XmNnumChildren	Composite
XmNentryLabelType	XmBulletinBoard	XmNaccelerators	Core
XmNfillStyle	XmBulletinBoard	XmNancestorSensitive	Core
XmNshowEntryLabel	XmBulletinBoard	XmNbackground	Core
XmNstretchable	XmBulletinBoard	XmNbackgroundPixmap	Core
XmNbottomShadowColor	XmManager	XmNborderColor	Core
XmNbottomShadowPixmap	XmManager	XmNborderPixmap	Core
XmNforeground	XmManager	XmNborderWidth	Core
XmNhelpCallback	XmManager	XmNcolormap	Core
XmNhighlightColor	XmManager	XmNdepth	Core
XmNhighlightPixmap	XmManager	XmNdestroyCallback	Core
XmNinitialFocus	XmManager	XmNheight	Core
XmNlayoutDirection	XmManager	XmNinitialResourcesPersistent	Core
XmNnavigationType	XmManager	XmNmappedWhenManaged	Core
XmNpopupHandlerCallback	XmManager	XmNscreen	Core
XmNshadowThickness	XmManager	XmNsensitive	Core
XmNstringDirection	XmManager	XmNtranslations	Core
XmNtopShadowColor	XmManager	XmNwidth	Core
XmNtopShadowPixmap	XmManager	XmNx	Core
XmNtraversalOn	XmManager	XmNy	Core
XmNunitType	XmManager		

See Also

XmCreateObject(1), Composite(2), Constraint(2), Core(2), XmManager(2), XmNotebook, XmBulletinBoard(2).

Name

XmTemplateDialog –an unmanaged MessageBox as a child of DialogShell.

Synopsis

Public Header:
<Xm/MessageB.h>

Instantiation:
`widget` = `XmCreateTemplateDialog` (parent, name,...)

Functions/Macros:
`XmCreateTemplateDialog`(), `XmMessageBoxGetChild`()

Description

An XmTemplateDialog is a compound object created by a call to `XmCreateTemplateDialog`() that an application can use to present a customized message to the user. A TemplateDialog consists of a DialogShell with an unmanaged MessageBox widget as its child. The MessageBox resource XmNdialogType is set to XmDIALOG_TEMPLATE. By default, a TemplateDialog includes only a separator. An application can create a customized dialog by adding children to the TemplateDialog. To create the standard components associated with a MessageBox, an application needs only specify the label string and callback resources for the desired buttons, and the TemplateDialog automatically creates the relevant button. Setting either the XmNmessageString or XmNsymbolPixmap resource creates a message or a symbol Label.

Default Resource Values

A TemplateDialog sets the following default values for MessageBox resources:

Name	Default
XmNdialogType	XmDIALOG_TEMPLATE

Widget Hierarchy

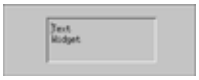
When a TemplateDialog is created with a specified name, the DialogShell is named name*_popup* and the MessageBox is called *name*.

See Also

`XmCreateObject`(1), `XmMessageBoxGetChild`(1), `XmDialogShell`(2), `XmMessageBox`(2).

Name

XmText widget class – text-editing widget.



Synopsis

Public Header:
<Xm/Text.h>

Class Name:
XmText

Class Hierarchy:
Core → XmPrimitive → XmText

Class Pointer:
xmTextWidgetClass

Instantiation:
widget = `XmCreateText` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmTextWidgetClass,...)

Functions/Macros:
`XmCreateScrolledText`(), `XmCreateText`(), `XmIsText`(), XmText... routines

Description

A Text widget provides a text editor that allows text to be inserted, modified, deleted, and selected. Text provides both single-line and multi-line text editing capabilities.

Traits

Text holds the XmQTaccessTextual and XmQTtransfer traits, which are inherited in any derived classes, and uses the XmQTaccessTextual, XmQTspecifyRenderTable, XmQTnavigator and XmQTscrollFrame traits.

New Resources

Text defines the following resources:

Name	Class	Type	Default	Access
XmNautoShowCursorPosition	XmCAutoShowCursorPosition	Boolean	True	CSG
XmNcursorPosition	XmCCursorPosition	XmTextPosition	0	CSG
XmNeditable	XmCEditable	Boolean	True	CSG
XmNeditMode	XmCEditMode	int	see below	CSG

Name	Class	Type	Default	Access
XmNmarginHeight	XmCMarginHeight	Dimension	5	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	5	CSG
XmNmaxLength	XmCMaxLength	int	largest integer	CSG
XmNsource	XmCSource	XmTextSource	default source	CSG
XmNtotalLines	XmCTotalLines	int	1	CG
XmNtopCharacter	XmNtopCharacter	XmTextPosition	0	CSG
XmNvalue	XmCValue	String	""	CSG
XmNvalueWcs	XmCValueWcs	wchar_t	(Wchar_t *) ""	CSG
XmNverifyBell	XmCVerifyBell	Boolean	dynamic	CSG

XmNautoShowCursorPosition
: If True (default), the visible portion of the Text widget will always contain the insert cursor. The Text widget will scroll its contents, if necessary, to ensure that the cursor remains visible.

XmNcursorPosition
: The location at which to place the current insert cursor. Values for this resource are relative to the beginning of the text; the first character position is defined as 0.

XmNeditable
: If True (default), the user is allowed to edit the text string; if False, the user is not allowed to do so.

XmNeditMode
: Determines which group of keyboard bindings to use. Possible values:

```
XmMULTI_LINE_EDIT     /* key bindings for multi-line edits          */
XmSINGLE_LINE_EDIT    /* key bindings for single line edits (default) */
```

XmNmarginHeight

XmNmarginWidth
: The spacing between the edges of the widget and the text. (Top and bottom edges for height; left and right for width.)

XmNmaxLength
: The maximum length of the text string that a user can enter from the keyboard. This resource does not affect strings that are entered via the XmNvalue resource or the `XmTextSetString`() routine.

XmNsource
: A source that the Text widget uses for displaying text, thereby allowing Text widgets to share the same text source.

XmNtotalLines
In Motif 2.1, specifies the number of lines within the Text widget buffer. The value takes into account word wrapping.

XmNtopCharacter
The location of the text to display at the top of the window. Values for this resource are relative to the beginning of the text, with the first character position defined as 0.

XmNvalue
The string value to display in the Text widget, expressed as a char *. If XmNvalue and XmNvalueWcs are both defined, XmNvalueWcs takes precedence. Use `XtSetValues`() to copy string values to the internal buffer and use `XtGetValues`() to return the value of the internal buffer.

XmNvalueWcs
In Motif 1.2, the string value to display in the Text widget, expressed as a wchar_t *. If XmNvalue and XmNvalueWcs are both defined, XmNvalueWcs takes precedence. Use `XtSetValues`() to copy string values to the internal buffer and use `XtGetValues`() to return the value of the internal buffer. This resource cannot be set in a resource file.

XmNverifyBell
If True, a bell will sound when a verification produces no action. The default value depends upon the XmNaudibleWarning resource of any VendorShell ancestor.

Text Input Resources

Name	Class	Type	Default	Access
XmNpendingDelete	XmCPendingDelete	Boolean	True	CSG
XmNselectionArray	XmCSelectionArray	XtPointer	default array	CSG
XmNselectionArrayCount	XmCSelectionArrayCount	int	4	CSG
XmNselectThreshold	XmCSelectThreshold	int	5	CSG

XmNpendingDelete
If True (default), the Text widget's pending delete mode is on, meaning that selected text will be deleted as soon as the next text insertion occurs.

XmNselectionArray
The array of possible actions caused by multiple mouse clicks. UIL does not define these values for the Text widget. Possible values:

```
XmSELECT_POSITION    /* single-click; reset position of insert cursor */
XmSELECT_WORD        /* double-click; select a word                   */
XmSELECT_LINE        /* triple-click; select a line                   */
```

```
XmSELECT_ALL        /* quadruple-click; select all text        */
```

XmNselectionArrayCount
The number of items in the array specified by XmNselectionArray.

XmNselectThreshold
The number of pixels the insertion cursor must be dragged during selection in order to select the next character.

Text Output Resources

Name	Class	Type	Default	Access
XmNblinkRate	XmCBlinkRate	int	500	CSG
XmNcolumns	XmCColumns	short	dynamic	CSG
XmNcursorPositionVisible	XmCCursorPositionVisible	Boolean	dynamic	CSG
XmNfontList	XmCFontList	XmFontList	dynamic	CSG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNresizeHeight	XmCResizeHeight	Boolean	False	CSG
XmNresizeWidth	XmCResizeWidth	Boolean	False	CSG
XmNrows	XmCRows	short	dynamic	CSG
XmNwordWrap	XmCWordWrap	Boolean	False	CSG

XmNblinkRate
The time in milliseconds that the cursor spends either being visible or invisible. A value of 0 prevents the cursor from blinking.

XmNcolumns
The number of character spaces that should fit horizontally in the text window. The XmNwidth resource determines the default value of XmNcolumns, but if no width has been set, the default is 20. See also XmNrows.

XmNcursorPositionVisible
If True, the text cursor will be visible. In Motif 2.1, if the text widget has an XmPrintShell as ancestor, the default is False. Otherwise the default is True.

XmNfontList
The font list used for the widget's text. In Motif 2.0 and later, the XmFontList is obsolete, and is subsumed into the XmRenderTable. If both a render table and font list are specified, the render table takes precedence.

XmNrenderTable
In Motif 2.0 and later, specifies the render table for the Text. If unspecified, the value of the resource is inherited from the nearest ancestor which holds the XmQTspecifyRenderTable trait, using the XmTEXT_RENDER_TABLE value of the ancestor so found.

XmNresizeHeight
: If False (default), the Text widget will not expand vertically to fit all of the text (in other words, the widget will need to have scrollbars so that the rest of the text can be scrolled into view). If True, the Text widget always begins its display with the text at the beginning of the source. This resource has no effect in a ScrolledText widget whose XmNscrollVertical resource is set to True.

XmNresizeWidth
: If False (default), the Text widget will not expand horizontally to fit its text. If True, the widget tries to change its width. This resource has no effect when the XmNwordWrap resource is set to True.

XmNrows
: The number of character spaces that should fit vertically in the text window. The XmNheight resource determines the default value of XmNrows, but if no height has been set, the default is 1. This resource is meaningful only when XmNeditMode is XmMULTI_LINE_EDIT. See also XmNcolumns.

XmNwordWrap
: If False (default), does not break lines automatically between words (in which case text can disappear beyond the window's edge). If True, breaks lines at spaces, tabs, or new lines. This resource is meaningful only when XmNeditMode is XmMULTI_LINE_EDIT.

Scrolled Text Resources

Name	Class	Type	Default	Access
XmNscrollHorizontal	XmCScroll	Boolean	True	CG
XmNscrollLeftSide	XmCScrollSide	Boolean	dynamic	CG
XmNscrollTopSide	XmCScrollSide	Boolean	False	CG
XmNscrollVertical	XmCScroll	Boolean	True	CG

XmNscrollHorizontal
: If True, the Text widget adds a horizontal ScrollBar. The default is True; however, the value changes to False if the widget is in a ScrolledWindow whose XmNscrollingPolicy resource is set to XmAUTOMATIC. This resource is meaningful only when XmNeditMode is XmMULTI_LINE_EDIT.

XmNscrollLeftSide
: If True, the vertical ScrollBar is placed to the left of the scrolled text window. The default value depends on how the XmNstringDirection resource is set. This resource is meaningful only when XmNeditMode is XmMULTI_LINE_EDIT and when XmNscrollVertical is True.

XmNscrollTopSide
If True, the horizontal ScrollBar is placed above the scrolled text window, rather than below by default.

XmNscrollVertical
If True, the Text widget adds a vertical ScrollBar. The default is True; however, the value changes to False if the widget is in a ScrolledWindow whose XmNscrollingPolicy resource is set to XmAUTOMATIC.

Callback Resources

Text defines the following callback resources:

Callback	Reason Constant
XmNactivateCallback	XmCR_ACTIVATE
XmNdestinationCallback	XmCR_OK
XmNfocusCallback	XmCR_FOCUS
XmNgainPrimaryCallback	XmCR_GAIN_PRIMARY
XmNlosePrimaryCallback	XmCR_LOSE_PRIMARY
XmNlosingFocusCallback	XmCR_LOSING_FOCUS
XmNmodifyVerifyCallback	XmCR_MODIFYING_TEXT_VALUE
XmNmodifyVerifyCallbackWcs	XmCR_MODIFYING_TEXT_VALUE
XmNmotionVerifyCallback	XmCR_MOVING_INSERT_CURSOR
XmNvalueChangedCallback	XmCR_VALUE_CHANGED

XmNactivateCallback
List of callbacks that are called when the user causes the Text widget to be activated.

XmNdestinationCallback
List of callbacks that are called when the Text is the destination of a transfer operation.

XmNfocusCallback
List of callbacks that are called when the Text widget receives the input focus.

XmNgainPrimaryCallback
List of callbacks that are called when the Text widget gains ownership of the primary selection.

XmNlosePrimaryCallback
List of callbacks that are called when the Text widget loses ownership of the primary selection.

XmNlosingFocusCallback
List of callbacks that are called when the Text widget loses the input focus.

XmNmodifyVerifyCallback
List of callbacks that are called before the value of the Text widget is changed. If there are callbacks for both XmNmodifyVerifyCallback and XmNmodifyVerifyCallbackWcs, the XmNmodifyVerifyCallback callbacks are called first.

XmNmodifyVerifyCallbackWcs
List of callbacks that are called before the value of the Text widget is changed. If there are callbacks for both XmNmodifyVerifyCallback and XmNmodifyVerifyCallbackWcs, the XmNmodifyVerifyCallback callbacks are called first.

XmNmotionVerifyCallback
List of callbacks that are called before the insertion cursor is moved in the Text widget.

XmNvalueChangedCallback
List of callbacks that are called after the value of the Text widget is changed.

Callback Structure

Destination callbacks are fully described within the sections covering the Uniform Transfer Model. See `XmTransfer`(1) for more details. For quick reference, a pointer to the following structure is passed to callbacks on the XmNdestinationCallback list:

```
typedef struct {
    int         reason;          /* the reason that the callback is invoked */
    XEvent      *event;          /* points to event that triggered callback */
    Atom        selection;       /* the requested selection type, as an Atom */
    XtEnum      operation;       /* the type of transfer requested          */
    int         flags;       /* whether destination and source are same */
    XtPointer   transfer_id;     /* unique identifier for the request       */
    XtPointer   destination_data;     /* information about the destination */
    XtPointer   location_data;       /* information about the data        */
    Time        time;            /* time when the transfer operation started */
} XmDestinationCallbackStruct;
```

With the exception of the above destination callback, each callback function is passed the following structure:

```
typedef struct {
    int             reason;      /* the reason that the callback was called */
    XEvent          *event;      /* event structure that triggered callback */
} XmAnyCallbackStruct;
```

In addition, the callback resources XmNlosingFocusCallback, XmNmodifyVerifyCallback, and XmNmotionVerifyCallback reference the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
    XEvent      *event;         /* points to event that triggered callback */
    Boolean     doit;           /* do the action (True) or undo it (False) */
    long        currInsert;     /* the insert cursor's current position    */
    long        newInsert;      /* desired new position of insert cursor   */
    long        startPos;       /* start of text to change                 */
    long        endPos;         /* end of text to change                   */
    XmTextBlocktext;            /* describes the text to insert            */
} XmTextVerifyCallbackStruct, *XmTextVerifyPtr;
```

start_pos specifies the location at which to start modifying text. *start_pos* is unused if the callback resource is XmNmotionVerifyCallback, and is the same as the *current_insert* member if the callback resource is XmNlosingFocusCallback.

end_pos specifies the location at which to stop modifying text (however, if no text was modified, *end_pos* has the same value as *start_pos*). *end_pos* is unused if the callback resource is XmNmotionVerifyCallback, and is the same as the *current_insert* member if the callback resource is XmNlosingFocusCallback.

text points to the structure below, which specifies information about the text to be inserted.

```
typedef struct {
    char        *ptr;           /* pointer to the text to insert           */
    int         length;         /* length of this text                     */
    XmTextFormat   format;      /* text format (e.g., FMT8BIT, FMT16BIT) */
} XmTextBlockRec, *XmTextBlock;
```

The callback resource XmNmodifyVerifyCallbackWcs references the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
    XEvent      *event;         /* structure that triggered callback       */
    Boolean     doit;           /* do the action (True) or undo it (False) */
    long        current_insert; /* the insert cursor's current position    */
    long        new_insert;     /* desired new position of insert cursor   */
    long        start_pos;      /* start of text to change                 */
    long        end_pos;        /* end of text to change                   */
    XmTextBlockWcstext;         /* describes the text to insert            */
} XmTextVerifyCallbackStructWcs, *XmTextVerifyPtrWcs;
```

All of the fields in this structure are the same as the fields in the XmTextVerify-CallbackStruct except text, which points to the structure below and specifies information about the text to be inserted.

```
typedef struct {
    wchar_t     *wcsptr;    /* pointer to the text to insert    */
    int         length;     /* length of this text              */
} XmTextBlockRecWcs, *XmTextBlockWcs;
```

Inherited Resources

Text inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. Text sets the default value of XmN-navigationType to XmTAB_GROUP. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNhighlightThickness	XmPrimitive
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNlayoutDirection	XmPrimitive
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnavigationType	XmPrimitive
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmPrimitive
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmPrimitive	XmNsensitive	Core
XmNbottomShadowPixmap	XmPrimitive	XmNshadowThickness	XmPrimitive
XmNcolormap	Core	XmNtoolTipString	XmPrimitive
XmNconvertCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNdepth	Core	XmNtopShadowPixmap	XmPrimitive
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNheight	Core	XmNunitType	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNuserData	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNwidth	Core
XmNhighlightOnEnter	XmPrimitive	XmNx	Core
XmNhighlightPixmap	XmPrimitive	XmNy	Core

Translations

The translations for Text include those from Primitive, as well as the following. (Note that some of the associated actions will be reversed for a language environment in which text is not read from left to right.)

Event	Action	Event	Action
BSelect Press	grab-focus()	MShift KPageDown	next-page(extend)
BSelect Motion	extend-adjust()	KPageLeft	page-left()
BSelect Release	extend-end()	KPageRight	page-right()
BExtend Press	extend-start()	KBeginLine	beginning-of-line()
BExtend Motion	extend-adjust()	MShift KBeginLine	beginning-of-line(extend)[a]
BExtend Release	extend-end()	KEndLine	end-of-line
BToggle Press	move-destination()	MShift KEndLine	end-of-line(extend)
BTransfer Press	process-bdrag() (1.2) secondary-start (1.1)	KBeginData	beginning-of-file()
BTransfer Motion	secondary-adjust()	MShift KBeginData	beginning-of-file(extend)
BTransfer Release	copy-to()	KEndData	end-of-file()
MCtl BTransfer Press	process-bdrag() (1.2) secondary-start (1.1)	MShift KEndData	end-of-file(extend)
MCtrl BTransfer Motion	secondary-adjust()	KTab	process-tab()
MCtrl BTransfer Release	copy-to()	KNextField	next-tab-group()
MAlt BTransfer Press	process-bdrag() (1.2) secondary-start (1.1)	KPrevField	prev-tab-group()
MAlt BTransfer Motion	secondary-adjust()	KEnter	process-return()
MAlt BTransfer Release	copy-to()	KActivate	activate()
MShift BTransfer Press	process-bdrag()	KDelete	delete-next-character()
MShift BTransfer Motion	secondary-adjust()	KBackSpace	delete-previous-character()
MShift BTransfer Release	move-to()	KAddMode	toggle-add-mode()
MAlt MCtrl BTransfer Release	copy-to()	KSpace	self-insert()
MAlt MShift BTransfer Release	move-to()	MShift KSpace	insert
KUp	process-up()	KSelect	set-anchor()
MShift KUp	process-shift-up()	KExtend	key-select()
MCtrl KUp	backward-paragraph()	MAny KCancel	process-cancel()
MShift MCtrl KUp	backward-paragraph(extend)	KClear	clear-selection()
KDown	process-down()	KSelectAll	select-all()
MShift KDown	process-shift-down()	KDeselectAll	deselect-all()
MCtrl KDown	forward-paragraph()	KCut	cut-clipboard()

Event	Action	Event	Action
MShift MCtrl KDown	forward-paragraph(extend)	KCopy	copy-clipboard()
KLeft	backward-character()	KPaste	paste-clipboard()
MShift KLeft	key-select(left)	KPrimaryCut	cut-primary()
MCtrl KLeft	backward-word()	KPrimaryCopy	copy-primary()
MShift MCtrl KLeft	backward-word(extend)	KPrimaryPaste	copy-primary()
KRight	forward-character()	KQuickCut	quick-cut-set() (1.1)
MShift KRight	key-select(right)	KQuickCopy	quick-copy-set() (1.1)
MCtrl KRight	forward-word()	KQuickPaste	quick-copy-set() (1.1)
MShift MCtrl KRight	forward-word(extend)	KQuickExtend	do-quick-action() (1.1)
KPageUp	previous-page()	KHelp	Help()
MShift KPageUp	previous-page(extend)	KAny	self-insert()
KPageDown	next-page()		

a. Erroneously given as beginning-of-file(extend) in 1st and 2nd editions.

Action Routines

Text defines the action routines below. For actions that involve movement such as next, previous, start, end, back, forward, etc., the actual cursor movement depends on whether the layout direction is left-to-right or right-to-left. In addition, some actions accept an optional argument, extend. When applied with no argument, these actions move the cursor; when applied with the extend argument, these actions move the cursor but also extend the text selection. In all descriptions, the term cursor refers to the insertion cursor.

activate()

Invokes the callbacks specified by XmNactivateCallback.

backward-character()

Moves the cursor back one character.

backward-paragraph(extend)

Moves the cursor back to the first non-blank character that follows a blank line (or back to the start of the text if there is no previous blank line). If the cursor is already located at a non-blank character (i.e., if it's already at the beginning of the paragraph), the cursor moves to the start of the previous paragraph. (Multi-line edit mode only.)

backward-word(extend)

Moves the cursor back to the first non-blank character that follows a blank character (or back to the start of the line if there is no previ-

ous blank character). If the cursor is already located at a non-blank character (i.e., if it's already at the beginning of a word), the cursor moves to the start of the previous word.

beep()
: Makes the terminal beep.

beginning-of-file(extend)
: Moves the cursor to the start of the text.

beginning-of-line(extend)
: Moves the cursor to the start of the line.

clear-selection()
: Replaces each character (except a newline) with a space, effectively clearing the current selection.

copy-clipboard()
: Copies the current text selection into the clipboard.

copy-primary()
: Inserts a copy of the primary selection at the cursor location.

copy-to()
: Inserts a copy of the secondary selection at the cursor location, or, if there is no secondary selection, inserts a copy of the primary selection at the pointer location.

cut-clipboard()
: Deletes the current selection and moves it to the clipboard.

cut-primary()
: Deletes the primary selection and inserts it at the cursor.

delete-next-character()
delete-previous-character()
: If the cursor is inside the selection and XmNpendingDelete is True, deletes the selection. Otherwise, deletes the character following/preceding the cursor.

delete-next-word()
delete-previous-word()
: If the cursor is inside the selection and XmNpendingDelete is True, deletes the selection. Otherwise, deletes from the character following/preceding the cursor to the next/previous space, tab, or end of line.

delete-selection()
: Deletes the current selection.

delete-to-end-of-line()
: Deletes forward from the character after the cursor up to and including the end of the line.

delete-to-start-of-line()
: Deletes back from the character before the cursor up to and including the beginning of the line.

deselect-all()
: Deselects the current selection.

do-quick-action(0
: In Motif 1.1, Ends a secondary selection and does the action that was started by either of the actions quick-copy-set or quick-cut-set.

end-of-file(extend)
: Moves the cursor to the end of the text.

end-of-line(extend)
: Moves the cursor to the end of the line.

extend-adjust()
: Selects text that is between the anchor and the pointer location, while deselecting text that is outside this area. As a result of this action, when the pointer moves past lines of text, these lines are selected and the current line is selected up to the position of the pointer.

extend-end()
: Moves the cursor to the pointer location and ends the selection performed by extend-adjust.

extend-start()
: Adjusts the anchor in preparation for selecting text via the extend-adjust action.

forward-character()
: Moves the cursor forward one character.

forward-paragraph(extend)
: Moves the cursor forward to the first non-blank character that follows a blank line. If the cursor is already located at a non-blank character (i.e., if it's already at the beginning of the paragraph), the cursor moves to the start of the next paragraph. (Multi-line edit mode only.)

forward-word(extend)
: Moves the cursor forward to the first blank character that follows a non-blank character (or forward to the end of the line if there is no blank character to move to). If the cursor is already located at a blank character (i.e., if it's already at the end of a word), the cursor moves to the end of the next word.

grab-focus()
: Processes multi-clicks as defined in the XmNselectionArray resource. By default, one click resets the cursor to the pointer location, two clicks select a word, three clicks select a line, and four clicks select all of the text.

Help()
: Invokes the list of callbacks specified by XmNhelpCallback. If the Text widget doesn't have any help callbacks, this action routine invokes those associated with the nearest ancestor that has them.

insert-string(text)
: Inserts text at the cursor, or replaces the current selection with text (when XmNpendingDelete is True).

key-select(direction)
: Extends the selection and moves the cursor one character to the right (when direction is right), one character to the left (direction is left). If no direction is specified, the selection is extended, although the insertion cursor is not moved.

kill-next-character()
kill-next-word()
kill-previous-character()
kill-previous-word()
: These four actions are similar to their delete action counterparts, but the kill actions have the added feature of storing the deleted text in the cut buffer.

kill-selection()
: Deletes the current selection and stores this text in the cut buffer.

kill-to-end-of-line()
: Deletes forward from the character after the cursor up to and including the end of the line; stores this text in the cut buffer.

kill-to-start-of-line()
: Deletes back from the character before the cursor up to and including the beginning of the line; stores this text in the cut buffer.

move-destination()
: Moves the cursor to the pointer location, leaving existing selections unaffected.

move-to()
: Deletes the secondary selection and inserts it at the cursor, or, if there is no secondary selection, deletes the primary selection and inserts it at the pointer location.

newline()
: If the cursor is inside the selection and XmNpendingDelete is True, deletes the selection and inserts a newline at the cursor. Otherwise, only inserts a newline at the cursor.

newline-and-backup()
: If the cursor is inside the selection and XmNpendingDelete is True, deletes the selection, inserts a newline at the cursor and moves the cursor to the end of the previous line. Otherwise, only inserts a newline and then moves the cursor to the end of the previous line.

newline-and-indent()
: If the cursor is inside the selection and XmNpendingDelete is True, deletes the selection, inserts a newline at the cursor, and adds blanks (as needed) so that the cursor aligns with the first non-blank character in the previous line. Otherwise, only inserts a newline and adds blanks (as needed) so that the cursor aligns with the first non-blank character in the previous line.

next-line()
: Places the cursor on the next line.

next-page(extend)
: Moves the cursor one page forward.

next-tab-group()
: Traverses to the next tab group.

page-left()
page-right()
: Scrolls the visible area one page to the left or right.

paste-clipboard()
: Pastes text from the clipboard to the position before the cursor.

prev-tab-group()
: Traverses to the previous tab group.

previous-line()
: Places the cursor on the previous line.

previous-page(extend)
: Moves the cursor one page backward.

process-bdrag()
: In Motif 1.2, copies the current selection to the insertion cursor if text is selected, the location cursor is outside of the selection, and no motion is detected. Performs a secondary selection and copies the selection to the position where text was last edited if the cursor is outside of the selection and motion is detected. Otherwise, initiates a drag and drop operation using the current selection.

process-cancel()
: Cancels the extend-adjust() or secondary-adjust() actions that are currently being applied, restoring the selection to its previous state.

process-down()
process-up()
: If XmNnavigationType is XmNONE, descends/ascends to the adjacent widget in the tab group (single-line edit mode only). Moves the cursor one line down/up (multi-line edit mode only).

process-home()
: Moves the cursor to the start of the line. (Similar to beginning-of-line.)

process-return()
: Invokes the XmNactivateCallback callbacks (in single-line editing) or inserts a newline (in multi-line editing).

process-shift-down()
process-shift-up()
: Moves the cursor one line down or up (in multi-line editing only).

process-tab()
: Traverses to the next tab group (in single-line editing) or inserts a tab (in multi-line editing).

quick-copy-set()
: In Motif 1.1, marks this text location as the start of the secondary selection to use in quick copying.

quick-cut-set()
: In Motif 1.1, marks this text location as the start of the secondary selection to use in quick cutting.

redraw-display()
: Redraws the text in the viewing window.

scroll-one-line-down()
scroll-one-line-up()
: Scrolls the text region one line down or up.

secondary-adjust()
: Extends the secondary selection to the location of the pointer.

secondary-notify()
: Inserts a copy of the secondary selection at the destination cursor.

secondary-start()
: In Motif 1.1, marks this text location as the start of a secondary selection.

select-adjust()
: Extends the selection via the multiple mouse clicks defined by the XmNselectionArray resource.

select-all()
: Selects all text.

select-end()
: Ends the selection made using the select-adjust() action.

select-start()
: Begins a text selection.

self-insert()
: The basic method of inserting text. Typing at the keyboard inserts new text and (if XmNpendingDelete is True) replaces selected text that the cursor is in.

set-anchor()
: Changes the anchor point used when making extended selections; changes the destination cursor used for secondary selections.

set-insertion-point()
: Sets the position of the cursor.

set-selection-hint()
: Sets the selection's text source and the selection's location.

toggle-add-mode()
: Turns Add Mode either on or off.

toggle-overstrike()
: Changes the text insertion mode. When a character is typed into a Text widget, by default it is inserted at the location of the insertion cursor. In overstrike mode, an inserted character replaces the current character that immediately follows the insertion cursor. When the insertion cursor is at the end of a line in overstrike mode, inserted characters are appended to the line.

traverse-home()
traverse-next()
traverse-prev()
: Traverse within the tab group to the first widget, the next widget, and the previous widget, respectively.

unkill()
: Restores the most recently deleted text to the cursor's location.

Additional Behavior

Text has the following additional behavior:

<FocusIn>
: Draws a solid insertion cursor and makes it blink.

<FocusOut>
: Draws a stippled I-beam insertion cursor, unless the widget is the destination of a data transfer.

See Also

`XmCreateObject(1), XmTextClearSelection(1), XmTextCopy(1), XmTextCut(1), XmTextDisableRedisplay(1), XmTextEnableRedisplay(1), XmTextFindString(1), XmTextFindStringWcs(1), XmTextGetBaseline(1), XmTextGetCursorPosition(1), XmTextGetEditable(1), XmTextGetInsertionPosition(1),`

XmTextGetLastPosition(1), XmTextGetMaxLength(1), XmTextGetSelection(1), XmTextGetSelectionPosition(1), XmTextGetSelectionWcs(1), XmTextGetSource(1), XmTextGetString(1), XmTextGetStringWcs(1), XmTextGetSubstring(1), XmTextGetSubstringWcs(1), XmTextGetTopCharacter(1), XmTextInsert(1), XmTextInsertWcs(1), XmTextPaste(1), XmTextPosToXY(1), XmTextRemove(1), XmTextReplace(1), XmTextReplaceWcs(1), XmTextScroll(1), XmTextSetAddMode(1), XmTextSetCursorPosition(1), XmTextSetEditable(1), XmTextSetHighlight(1), XmTextSetInsertionPosition(1), XmTextSetMaxLength(1), XmTextSetSelection(1), XmTextSetSource(1), XmTextSetString(1), XmTextSetStringWcs(1), XmTextSetTopCharacter(1), XmTextShowPosition(1), XmTextXYToPos(1), XmTransfer(1), Core(2), XmRendition(2), XmPrimitive(2), XmPrinShell(2), XmTextField(2).

Name

XmTextField widget class – a single-line text-editing widget.

Synopsis

Public Header:
<Xm/TextF.h>

Class Name:
XmTextField

Class Hierarchy:
Core → XmPrimitive → XmTextField

Class Pointer:
xmTextFieldWidgetClass

Instantiation:
widget = `XmCreateTextField` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmTextWidgetClass,...)

Functions/Macros:
`XmCreateTextField`(), `XmIsTextField`(), XmTextField... routines

Description

A TextField widget provides a single-line text editor that has a subset of the functionality of the Text widget.

Traits

TextField holds the XmQTaccessTextual and XmQTtransfer traits, which are inherited in any derived classes, and uses the XmQTaccessTextual and XmQTspecifyRenderTable traits.

New Resources

TextField defines the following resources:

Name	Class	Type	Default	Access
XmNblinkRate	XmCBlinkRate	int	500	CSG
XmNcolumns	XmCColumns	short	dynamic	CSG
XmNcursorPosition	XmCCursorPosition	XmTextPosition	0	CSG
XmNcursorPositionVisible	XmCCursorPositionVisible	Boolean	dynamic	CSG
XmNeditable	XmCEditable	Boolean	True	CSG
XmNfontList	XmCFontList	XmFontList	dynamic	CSG

Name	Class	Type	Default	Access
XmNmarginHeight	XmCMarginHeight	Dimension	5	CSG
XmNmarginWidth	XmCMarginWidth	Dimension	5	CSG
XmNmaxLength	XmCMaxLength	int	largest integer	CSG
XmNpendingDelete	XmCPendingDelete	Boolean	True	CSG
XmNrenderTable	XmCRenderTable	XmRenderTable	dynamic	CSG
XmNresizeWidth	XmCResizeWidth	Boolean	False	CSG
XmNselectionArray	XmCSelectionArray	XtPointer	default array	CSG
XmNselectionArrayCount	XmCSelectionArrayCount	int	3	CSG
XmNselectThreshold	XmCSelectThreshold	int	5	CSG
XmNvalue	XmCValue	String	""	CSG
XmNvalueWcs	XmCValueWcs	wchar_t	(Wchar_t *) ""	CSG
XmNverifyBell	XmCVerifyBell	Boolean	dynamic	CSG

XmNblinkRate
: The time in milliseconds that the cursor spends either being visible or invisible. A value of 0 prevents the cursor from blinking.

XmNcolumns
: The number of character spaces that should fit horizontally in the text window. The XmNwidth resource determines the default value of XmNcolumns, but if no width has been set, the default is 20.

XmNcursorPosition
: The location at which to place the current insert cursor. Values for this resource are relative to the beginning of the text, with the first character position defined as 0.

XmNcursorPositionVisible
: If True, the text cursor will be visible. In Motif 2.1, if the text field has an XmPrintShell as ancestor, the default is False. Otherwise the default is True.

XmNeditable
: If True (default), the user is allowed to edit the text string; if False, the user is not allowed to do so.

XmNfontList
: The font list used for the widget's text. In Motif 2.0 and later, the XmFontList is obsolete, and is subsumed into the XmRenderTable. If both a render table and font list are specified, the render table takes precedence.

XmNmarginHeight

XmNmarginWidth
The spacing between the edges of the widget and the text. (Top and bottom edges for height; left and right for width.)

XmNmaxLength
The maximum length of the text string that a user can enter from the keyboard. This resource doesn't affect strings that are entered via the XmNvalue resource or the `XmTextFieldSetString`() routine.

XmNpendingDelete
If True (default), the TextField widget's pending delete mode is on, meaning that selected text will be deleted as soon as the next text insertion occurs.

XmNrenderTable
In Motif 2.0 and later, specifies the render table for the TextField. If unspecified, the value of the resource is inherited from the nearest ancestor which holds the XmQTspecifyRenderTable trait, using the XmTEXT_RENDER_TABLE value of the ancestor so found.

XmNresizeWidth
If False (default), the TextField widget will not expand horizontally to fit its text. If True, the widget tries to change its width.

XmNselectionArray
The array of possible actions caused by multiple mouse clicks. UIL does not define these values for the Text widget. Possible values:

```
XmSELECT_POSITION     /* single-click; reset position of insert cursor   */
XmSELECT_WORD         /* double-click; select a word                      */
XmSELECT_LINE         /* triple-click; select a line                      */
```

XmNselectionArrayCount
The number of items in the array specified by XmNselectionArray.

XmNselectThreshold
The number of pixels the insertion cursor must be dragged during selection in order to select the next character.

XmNvalue
The string value to display in the TextField widget, expressed as a char *. If XmNvalue and XmNvalueWcs are both defined, XmNvalueWcs takes precedence. Use XtSetValues() to copy string values to the internal buffer and use XtGetValues() to return the value of the internal buffer.

XmNvalueWcs
In Motif 1.2 and later, the string value to display in the TextField widget, expressed as a wchar_t *. If XmNvalue and XmNvalueWcs are both defined, XmNvalueWcs takes precedence. Use XtSetValues() to copy string values to the

internal buffer and use XtGetValues() to return the value of the internal buffer. This resource cannot be set in a resource file.

XmNverifyBell
If True, a bell will sound when a verification produces no action.

Callback Resources

TextField defines the same callback resources and references the same callback structures as a single line Text widget.

Inherited Resources

TextField inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. TextField sets the default value of XmNnavigationType to XmTAB_GROUP. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNaccelerators	Core	XmNhighlightThickness	XmPrimitive
XmNancestorSensitive	Core	XmNinitialResourcesPersistent	Core
XmNbackground	Core	XmNlayoutDirection	XmPrimitive
XmNbackgroundPixmap	Core	XmNmappedWhenManaged	Core
XmNborderColor	Core	XmNnavigationType	XmPrimitive
XmNborderPixmap	Core	XmNpopupHandlerCallback	XmPrimitive
XmNborderWidth	Core	XmNscreen	Core
XmNbottomShadowColor	XmPrimitive	XmNsensitive	Core
XmNbottomShadowPixmap	XmPrimitive	XmNshadowThickness	XmPrimitive
XmNcolormap	Core	XmNtoolTipString	XmPrimitive
XmNconvertCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNdepth	Core	XmNtopShadowPixmap	XmPrimitive
XmNdestroyCallback	Core	XmNtranslations	Core
XmNforeground	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNheight	Core	XmNunitType	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNuserData	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNwidth	Core
XmNhighlightOnEnter	XmPrimitive	XmNx	Core
XmNhighlightPixmap	XmPrimitive	XmNy	Core

Translations

TextField has the same translation as a Text widget whose XmNeditMode resource is set to XmSINGLE_LINE_EDIT.

Action Routines

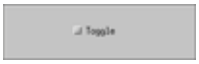
TextField defines the same action routines as a Text widget whose XmNeditMode resource is set to XmSINGLE_LINE_EDIT.

See Also

XmCreate*Object*(1), XmTextClearSelection(1), XmTextCopy(1), XmTextCut(1), XmTextGetBaseline(1), XmTextGetCursorPosition(1), XmTextGetEditable(1), XmTextGetInsertionPosition(1), XmTextGetLastPosition(1), XmTextGetMaxLength(1), XmTextGetSelection(1), XmTextGetSelectionPosition(1), XmTextGetSelectionWcs(1), XmTextGetString(1), XmTextGetStringWcs(1), XmTextGetSubstring(1), XmTextGetSubstringWcs(1), XmTextInsert(1), XmTextInsertWcs(1), XmTextPaste(1), XmTextPosToXY(1), XmTextRemove(1), XmTextReplace(1), XmTextReplaceWcs(1), XmTextScroll(1), XmTextSetAddMode(1), XmTextSetCursorPosition(1), XmTextSetEditable(1), XmTextSetHighlight(1), XmTextSetInsertionPosition(1), XmTextSetMaxLength(1), XmTextSetSelection(1), XmTextSetSource(1), XmTextSetString(1), XmTextSetStringWcs(1), XmTextSetTopCharacter(1), XmTextShowPosition(1), XmTextXYToPos(1), Core(2), XmPrimitive(2), XmPrintShell(2), XmRendition(2), XmText(2).

Name

XmToggleButton widget class –a button widget that maintains a Boolean state.



Synopsis

Public Header:
<Xm/ToggleB.h>

Class Name:
XmToggleButton

Class Hierarchy:
Core → XmPrimitive → XmLabel → XmToggleButton

Class Pointer:
xmToggleButtonWidgetClass

Instantiation:
widget = `XmCreateToggleButton` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmToggleButtonWidgetClass,...)

Functions/Macros:
`XmCreateToggleButton`(), `XmToggleButtonGetState`(), `XmToggleButtonSetState`(), `XmToggleButtonSetValue`(), `XmIsToggleButton`()

Description

A ToggleButton is a button that is either set or unset. ToggleButtons are typically used in groups, called RadioBoxes and CheckBoxes, depending on the behavior of the buttons. In a RadioBox, a ToggleButton displays *one-of-many* behavior, which means that only one button in the group can be set at a time. When a button is selected, the previously selected button is unset. In a CheckBox, a ToggleButton displays *n-of-many* behavior, which means that any number of ToggleButtons can be set at one time. ToggleButton uses an indicator to show its state; the shape of the indicator specifies the type of behavior. A diamond-shaped indicator is used for one-of-many ToggleButtons and a square-shaped indicator is used for n-of-many ToggleButtons.

In Motif 2.0 and later, a ToggleButton can have three possible states: set, unset, and indeterminate, depending upon the value of the XmNtoggleMode resource. If the value is XmTOGGLE_BOOLEAN, the ToggleButton has the two states, set and unset. If the value is XmTOGGLE_INDETERMINATE, the ToggleButton has three states.

Traits

ToggleButton uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

New Resources

ToggleButton defines the following resources, all with CSG access:

Name	Class	Type	Default
XmNdetailShadowThickness	XmCDetailShadowThickness	Dimension	dynamic
XmNfillOnSelect	XmCFillOnSelect	Boolean	dynamic
XmNindeterminate InsensitivePixmap	XmCIndeterminate InsensitivePixmap	Pixmap	XmUNSPECIFIED_PIXMAP
XmNindeterminatePixmap	XmCIndeterminatePixmap	Pixmap	XmUNSPECIFIED_PIXMAP
XmNindicatorOn	XmCIndicatorOn	unsigned char	XmINDICATOR_FILL
XmNindicatorSize	XmCIndicatorSize	Dimension	dynamic
XmNindicatorType	XmCIndicatorType	unsigned char	dynamic
XmNselectColor	XmCSelectColor	Pixel	dynamic
XmNselectInsensitivePixmap	XmCSelectInsensitivePixmap	Pixmap	XmUNSPECIFIED_PIXMAP
XmNselectPixmap	XmCSelectPixmap	Pixmap	XmUNSPECIFIED_PIXMAP
XmNset	XmCSet	unsigned char	XmUNSET
XmNspacing	XmCSpacing	Dimension	4
XmNtoggleMode	XmCToggleMode	unsigned char	XmTOGGLE_BOOLEAN
XmNunselectColor	XmCUnselectColor	Pixel	dynamic
XmNvisibleWhenOff	XmCVisibleWhenOff	Boolean	dynamic

XmNdetailShadowThickness

In Motif 2.0 and later, specifies the thickness of the shadow on the indicator. In Motif 2.0 the default value is 2. In Motif 2.1 and later, the default value depends upon the XmDisplay XmNenableThinThickness resource: if True the default is 1, otherwise 2.

XmNfillOnSelect

If True, selection of this ToggleButton fills the indicator with the color given by the XmNselectColor resource and switches the button's top and bottom shadow colors.

If the ToggleButton is unselected, the top and bottom shadow colors are switched. In Motif 2.0 and later, the indicator is filled with the color given by the XmNunselectColor resource.

If the ToggleButton is in the indeterminate state as specified by the XmNset resource, the indicator is half filled with the XmNselectColor and half with the XmNunselectColor values.

If fill on select is False, only the top and bottom shadow colors are switched.

When XmNindicatorOn is XmINDICATOR_NONE, XmNfillOnSelect is True, and XmNset is XmSET, the background of the entire button is filled with the XmNselectColor.

In Motif 1.2 and earlier, the default value is set to the value of XmNindicatorOn. In Motif 2.0 and later, the default depends upon both XmNindicatorOn and XmNindicatorType resources.

XmNindeterminateInsensitivePixmap
Specifies the pixmap to use if XmNset is XmINDETERMINATE and the Toggle is insensitive. The resource has no effect if the inherited Label resource XmNlabelType is XmSTRING.

XmNindeterminatePixmap
Specifies the pixmap to use if XmNset is XmINDETERMINATE and the Toggle is sensitive. The resource has no effect if the inherited Label resource XmNlabelType is XmSTRING.

XmNindicatorOn
In Motif 1.2 and earlier, the resource is a Boolean value. If True (default), the indicator is visible and its shadows are switched when the button is toggled. If False, the indicator is invisible and no space is set aside for it; in addition, the shadows surrounding the button are switched when it is toggled.
In Motif 2.0 and later, the resource is an enumerated type, and it specifies the type of indicator required. Possible values:

```
XmINDICATOR_NONE        /* no indicator                  */
XmINDICATOR_FILL        /* check box or box              */
XmINDICATOR_BOX         /* shadowed box, in Motif 2.1    */
XmINDICATOR_CHECK       /* checkmark                     */
XmINDICATOR_CHECK_BOX   /* checkmark enclosed in a box   */
XmINDICATOR_CROSS       /* cross                         */
XmINDICATOR_CROSS_BOX   /* cross enclosed in a box       */
```

If the value of the XmDisplay object's XmNenableToggleVisual resource is True, XmINDICATOR_FILL is equivalent to XmINDICATOR_CHECK_BOX, otherwise XmINDICATOR_BOX.

XmNindicatorSize
The size of the indicator. This value changes if the size of the button's text string or pixmap changes.

XmNindicatorType
Determines whether the indicator is drawn as a diamond (signifying a one-of-many indicator) or as a square (signifying an n-of-many indicator). Possible values:

```
XmN_OF_MANY                 /* creates a square button                  */
XmONE_OF_MANY       /* creates either round- or diamond-shaped button */
XmONE_OF_MANY_ROUND  /* creates a round-shaped button  (2.0)        */
XmONE_OF_MANY_DIAMOND /* creates a diamond-shaped button (2.1)  */
```

In Motif 2.0, the value XmONE_OF_MANY is diamond shaped. In Motif 2.1, XmONE_OF_MANY produces either a diamond or a round shape, depending upon the value of the XmDisplay XmNenableToggleVisual resource. If this is True, the shape is round. The default value is XmONE_OF_MANY for a ToggleButton in a RadioBox widget, and XmN_OF_MANY otherwise. This resource only sets the indicator; it is RowColumn's XmNradioBehavior resource that actually enforces radioButton or checkButton behavior.

XmNselectColor
The color with which to fill the indicator when the button is selected. On a color display, the default is a value between the background color and the bottom shadow color; on a monochrome display, the default is the foreground color.

In Motif 2.0 and later, the following Pixel values are pre-defined for special meaning:

```
XmDEFAULT_SELECT_COLOR     /* a color between the background      */
                           /* and bottom shadow                   */
XmREVERSED_GROUND_COLORS   /* select is foreground,               */
                           /* text drawn in background            */
XmHIGHLIGHT_COLOR      /* select color same as highlight color    */
```

XmNselectInsensitivePixmap
The pixmap used for an insensitive ToggleButton when it's selected. An unselected, insensitive ToggleButton uses the pixmap specified by the Label resource XmNlabelInsensitivePixmap. However, if this Label resource wasn't specified, it is set to the value of XmNselectInsensitivePixmap. This resource is meaningful only when the Label resource XmNlabelType is set to XmPIXMAP or XmPIXMAP_AND_STRING.

XmNselectPixmap
The pixmap used for a (sensitive) ToggleButton when it's selected. An unselected ToggleButton uses the pixmap specified by the Label resource XmNlabelPixmap. This resource is meaningful only when the Label resource XmNlabelType is set to XmPIXMAP or XmPIXMAP_AND_STRING.

XmNset
The selection state of the button. In Motif 1.2 and earlier, a simple Boolean value. In Motif 2.0 and later, a Toggle can be in three states: on, off, and indeterminate, and this resource is changed to an enumerated type. Possible values:

```
XmUNSET
XmSET
XmINDETERMINATE
```

If XmNtoggleMode is XmTOGGLE_INDETERMINATE, the Toggle cycles between XmSET, XmINDETERMINATE, XmUNSET, and then back to XmSET when pressed. If toggle mode is XmTOGGLE_BOOLEAN, the widget simply cycles between XmSET and XmUNSET.

Note that not all versions of Motif 2.1 allow the enumerated values to be specified in an external resource file. Version 2.1.30 sources have the problem fixed.

XmNspacing
The distance between the Toggle indicator and its label.

XmNtoggleMode
In Motif 2.0 and later, specifies whether the Toggle has two or three states. Possible values:

```
XmTOGGLE_BOOLEAN        /* two states   */
XmTOGGLE_INDETERMINATE  /* three states */
```

XmNunselectColor
In Motif 2.0 and later, specifies a color for filling the indicator shape. The resource behaves similarly to XmNselectColor, except that it is effective when XmNset is XmUNSET.

XmNvisibleWhenOff
If True, the Toggle indicator remains visible when the button is unselected. This is the default behavior in a RadioBox. The default is False in a menu.

Callback Resources

ToggleButton defines the following callback resources:

Callback	Reason Constant
XmNarmCallback	XmCR_ARM
XmNdisarmCallback	XmCR_DISARM
XmNvalueChangedCallback	XmCR_ACTIVATE

XmNarmCallback
List of callbacks that are called when BSelect is pressed while the pointer is inside the widget.

XmNdisarmCallback
List of callbacks that are called when BSelect is released after it has been pressed inside the widget.

XmNvalueChangedCallback
List of callbacks that are called when the value of the ToggleButton is changed.

Callback Structure

Each callback function is passed the following structure:

```
typedef struct {
    int         reason;         /* the reason that the callback was called */
    XEvent      *event;         /* points to event that triggered callback */
    int         set;            /* the state of the button                 */
} XmToggleButtonCallbackStruct;
```

set indicates the state of the Toggle, and is one of XmSET, XmUNSET, or XmINDETERMINATE.

Inherited Resources

ToggleButton inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. ToggleButton sets the default values of XmNmarginBottom, XmNmarginTop, XmNmarginWidth, and XmNshadowThickness dynamically. The default value of XmNborderWidth is reset to 0 by Primitive. In Motif 2.0 and earlier, the default value of XmNhighlightThickness is reset to 2. In Motif 2.1 and later, the default value depends upon the XmDisplay XmNenableThinThickness resource: if True the default is 1, otherwise 2.

Resource	Inherited From	Resource	Inherited From
XmNaccelerator	XmLabel	XmNlayoutDirection	XmPrimitive
XmNaccelerators	Core	XmNmappedWhenManaged	Core
XmNacceleratorText	XmLabel	XmNmarginBottom	XmLabel
XmNalignment	XmLabel	XmNmarginHeight	XmLabel
XmNancestorSensitive	Core	XmNmarginLeft	XmLabel
XmNbackground	Core	XmNmarginRight	XmLabel
XmNbackgroundPixmap	Core	XmNmarginTop	XmLabel
XmNborderColor	Core	XmNmarginWidth	XmLabel
XmNborderPixmap	Core	XmNmnemonicCharSet	XmLabel
XmNborderWidth	Core	XmNmnemonic	XmLabel
XmNbottomShadowColor	XmPrimitive	XmNnavigationType	XmPrimitive
XmNbottomShadowPixmap	XmPrimitive	XmNpopupHandlerCallback	XmPrimitive

Resource	Inherited From	Resource	Inherited From
XmNcolormap	Core	XmNrecomputeSize	XmLabel
XmNconvertCallback	XmPrimitive	XmNrenderTable	XmLabel
XmNdepth	Core	XmNscreen	Core
XmNdestroyCallback	Core	XmNsensitive	Core
XmNfontList	XmLabel	XmNshadowThickness	XmPrimitive
XmNforeground	XmPrimitive	XmNstringDirection	XmLabel
XmNheight	Core	XmNtoolTipString	XmPrimitive
XmNhelpCallback	XmPrimitive	XmNtopShadowColor	XmPrimitive
XmNhighlightColor	XmPrimitive	XmNtopShadowPixmap	XmPrimitive
XmNhighlightOnEnter	XmPrimitive	XmNtranslations	Core
XmNhighlightPixmap	XmPrimitive	XmNtraversalOn	XmPrimitive
XmNhighlightThickness	XmPrimitive	XmNunitType	XmPrimitive
XmNinitialResourcesPersistent	Core	XmNuserData	XmPrimitive
XmNlabelInsensitivePixmap	XmLabel	XmNwidth	Core
XmNlabelPixmap	XmLabel	XmNx	Core
XmNlabelString	XmLabel	XmNy	Core
XmNlabelType	XmLabel		

Translations

The translations for ToggleButton include those from Primitive. In addition, ToggleButtons that are not in a menu system have the following translations:

Event	Action
BTransfer Press	ProcessDrag()
BSelect Press	Arm()
MCtrl BSelect Press	ButtonTakeFocus()
BSelect Release	Select() Disarm()
KHelp	Help()
KSelect	ArmAndActivate()

For ToggleButtons that are in a menu system, translations include the menu traversal translations inherited from the Label widget, as well as the following:

Event	Action
MCtrl BSelect Press	MenuButtonTakeFocus()
BSelect Press	BtnDown()
BSelect Release	BtnUp()
KHelp	Help()
KActivate	ArmAndActivate()
KSelect	ArmAndActivate()
MAny KCancel	MenuShellPopdownOne()

Action Routines

ToggleButton defines the following action routines:

Arm()

Sets the button if it was previously unset, unsets the button if it was previously set, and invokes the callbacks specified by XmNarmCallback. Setting the button means displaying it so that it appears selected. The selected state can be shown by: Highlighting the indicator so it appears pressed in. Filling in the indicator (using the color given by XmNselectColor). Highlighting the button so it appears pressed in. (This is done only if the indicator isn't displayed). Drawing the button face using the pixmap given by XmNselectPixmap.

The unselected state can be shown by: Highlighting the indicator so it appears raised. Filling in the indicator with the background color. Highlighting the button so it appears raised. (This is done only if the indicator isn't displayed). Drawing the button face using the pixmap given by XmNlabelPixmap.

ArmAndActivate()

Sets the button if it was previously unset, unsets the button if it was previously set, and invokes the callbacks specified by XmNarmCallback (if the button isn't yet armed), XmNvalueChangedCallback, and XmNdisarmCallback. Inside a menu, this action unposts the menu hierarchy. Outside a menu, this action displays the button as selected or unselected, as described for Arm().

BtnDown()
: Unposts any menus that were posted by the parent menu of the ToggleButton, changes from keyboard traversal to mouse traversal, draws a shadow to show the ToggleButton as armed, and (assuming the button is not yet armed) invokes the callbacks specified by XmNarmCallback.

BtnUp()
: Unposts the menu hierarchy, changes the ToggleButton's state, and invokes first the callbacks specified by XmNvalueChangedCallback and then those specified by XmNdisarmCallback.

ButtonTakeFocus()
: In Motif 2.0 and later, moves the current keyboard focus to the ToggleButton, without activating the widget.

Disarm()
: Invokes the callbacks specified by XmNdisarmCallback.

Help()
: Unposts the menu hierarchy, restores the previous keyboard focus, and invokes the callbacks specified by the XmNhelpCallback resource.

MenuButtonTakeFocus()
: In Motif 2.0 and later, moves the current keyboard focus to the ToggleButton, without activating the widget.

MenuShellPopdownOne()
: Unposts the current menu and (unless the menu is a pulldown submenu) restores keyboard focus to the tab group or widget that previously had it. In a top-level pulldown menu pane attached to a menu bar, this action routine also disarms the cascade button and the menu bar.

ProcessDrag()
: In Motif 1.2, initiates a drag and drop operation using the label of the ToggleButton.

Select()
: Switches the state of the ToggleButton and invokes the callbacks specified by the resource XmNvalueChangedCallback.

Additional Behavior

ToggleButton has the following additional behavior:

<EnterWindow>
Displays the ToggleButton as armed.

<LeaveWindow>
Displays the ToggleButton as unarmed.

See Also

`XmCreateObject` (1), `XmToggleButtonGetState` (1), `XmToggleButtonSetState` (1), `XmToggleButtonSetValue(1)`, `XmToggleButtonGetValue(1)`,`Core` (2), `XmPrimitive` (2), `XmLabel` (2), `XmCheckBox` (2), `XmRadioBox` (2), `XmRowColumn` (2).

Name

XmToggleButtonGadget widget class –a button gadget that maintains a Boolean state.

Synopsis

Public Header:
<Xm/ToggleBG.h>

Class Name:
XmToggleButtonGadget

Class Hierarchy:
Object → RectObj → XmGadget → XmLabelGadget → XmToggleButtonGadget

Class Pointer:
xmToggleButtonGadgetClass

Instantiation:
widget = `XmCreateToggleButtonGadget` (parent, name,...)
or
widget = `XtCreateWidget` (name, xmToggleButtonGadgetClass,...)

Functions/Macros:
`XmCreateToggleButtonGadget`(), `XmToggleButtonGadgetGetState`(), `XmToggleButtonGadgetSetState`(), `XmToggleButtonGadgetSetValue`(), `XmIsToggleButtonGadget`()

Description

ToggleButtonGadget is the gadget variant of ToggleButton.

ToggleButtonGadget's new resources, callback resources, and callback structure are the same as those for ToggleButton.

Traits

ToggleButtonGadget holds the XmQTcareParentVisual trait, which is inherited in any derived classes, and clones the XmQTmenuSavvy trait from the LabelGadget class. In addition, the widget uses the XmQTmenuSystem and XmQTspecifyRenderTable traits.

Inherited Resources

ToggleButtonGadget inherits the following resources. The resources are listed alphabetically, along with the superclass that defines them. ToggleButtonGadget sets the default values of XmNmarginBottom[1], XmNmarginTop, XmNmarginWidth, and XmNshadowThickness dynamically. The default value of XmNborderWidth is reset to 0 by Primitive.

Resource	Inherited From	Resource	Inherited From
XmNancestorSensitive	RectObj	XmNlayoutDirection	XmGadget
XmNbackground	XmGadget	XmNnavigationType	XmGadget
XmNbackgroundPixmap	XmGadget	XmNsensitive	RectObj
XmNbottomShadowColor	XmGadget	XmNshadowThickness	XmGadget
XmNbottomShadowPixmap	XmGadget	XmNtoolTipString	XmGadget
XmNborderWidth	RectObj	XmNtopShadowColor	XmGadget
XmNdestroyCallback	Object	XmNtopShadowPixmap	XmGadget
XmNforeground	XmGadget	XmNtraversalOn	XmGadget
XmNheight	RectObj	XmNunitType	XmGadget
XmNhelpCallback	XmGadget	XmNuserData	XmGadget
XmNhighlightColor	XmGadget	XmNwidth	RectObj
XmNhighlightOnEnter	XmGadget	XmNx	RectObj
XmNhighlightPixmap	XmGadget	XmNy	RectObj
XmNhighlightThickness	XmGadget		

Behavior

As a gadget subclass, ToggleButtonGadget has no translations associated with it. However, ToggleButtonGadget behavior corresponds to the action routines of the ToggleButton widget. See the ToggleButton action routines for more information.

Event	Action
BTransfer Press	ProcessDrag()
BSelect Press	Arm() BtnDown() in a menu
BSelect Release	Select(), Disarm() BtnUp() in a menu
KActivate	ArmAndActivate()

1.Erroneously given as XmNmarginBotton in 1st and 2nd editions.

Event	Action
KSelect	ArmAndActivate()
KHelp	Help()
MAny KCancel	MenuShellPopdownOne()
MCtrl BSelect Press	ButtonTakeFocus() MenuButtonTakeFocus() in a menu

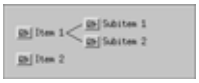
ToggleButtonGadget has additional behavior associated with <Enter> and <Leave>, which draw the shadow in the armed or unarmed state, respectively.

See Also

`XmCreateObject` (1), `XmToggleButtonGetState` (1), `XmToggleButtonSetState` (1), XmToggleButtonGadgetSetValue(1), XmToggleButtonGadgetSetValue(1), `Object` (2), `RectObj` (2), `XmCheckBox` (2), `XmGadget` (2), `XmLabelGadget` (2), `XmRadioBox` (2), `XmRowColumn` (2), `XmToggleButton` (2).

Name

XmTree – The Tree widget class



Synopsis

Public Header:
<Xm/Tree.h>

Class Name:
XmTree

Class Hierarchy:
Core → Composite → Constraint → XmManager → XmHierarchy → XmTree

Class Pointer:
xmTreeWidgetClass

Instantiation:

```
widget = XmCreateTree(parent, name,...)
or
widget = XtCreateWidget(name, xmTreeWidgetClass,...)
```

Functions/Macros:

```
XmCreateTree()
```

Availability

OpenMotif 2.2 and later. (Contributed Widget).

Description

The Tree widget is a container that shows the relationship of its children in a graphical tree-like format. Each child of the Tree widget is a node in the Tree. The parent-child relationships between these nodes are completely distinct from the widget hierarchy. The hierarchy of nodes is created by specifying the tree "parent" of each node as a constraint resource. If a node's parent is NULL then it is assumed to be a root of the tree. Although each widget can only have one parent, the Tree widget supports adding more than one "root" node to a single Tree. Note: the Tree widget assumes that is will be totally responsible for mapping and unmapping its children. Therefore no child of this widget should ever modify its mappedWhenManaged resource. If a child does modify this resource the behavior is undefined.

New Resources

The following table defines a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by

class in a **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (N/A).

Name	Class	Type	Default	Access
XmNconnectStyle	XmCConnectStyle	unsigned char	XmTreeDirect	CSG
XmNhorizontalNodeSpace	XmCDimension	Dimension	20	CSG
XmNverticalNodeSpace	XmCDimension	Dimension	2	CSG

XmNconnectStyle
: The style of the lines visually connecting parent nodes to children nodes. The valid styles are XmTreeDirect or XmTreeLadder.

XmNhorizontalNodeSpace

XmNverticalNodeSpace
: The amount of space between each node in the tree and it nearest neighbor.

Constraint Resources

XmTreeConstraint Resource Set

Name	Class	Type	Default	Access
XmNopenClosePadding	XmCOpenClosePadding	int	0	CSG
XmNlineColor	XmCForeground	Pixel	dynamic	CSG
XmNlineWidth	XmCLineWidth	int	0	CSG

XmNopenClosePadding
: The number of pixels between the folder button and the node it is associated with.

XmNlineColor
: The color of the line connecting a node to its parent. The default value for this resource is the foreground color of the Tree widget.

XmNlineWidth
: The width of a connection line between a node and its parent

The following constraint resources are inherited from the XmHierarchy widget:

XmHierarchy Constraint Resource Set

Name	Class	Type	Default	Access
XmNinsertBefore	XmCInsertBefore	Widget	NULL	S
XmNnodeState	XmCNodeState	XmHierarchyNodeState	XmOpen	CSG
XmNparentNode	XmCParentNode	Widget	NULL	S

Inherited Resources

Tree behavior and resources from the superclasses described in the following tables. For a complete description of each resource, refer to the reference page for that superclass

Resource	Inherited from	Resource	Inherited from
XmNautoClose	XmHierarchy	XmNchildren	Composite
XmNcloseFolderPixmap	XmHierarchy	XmNinsertPosition	Composite
XmNhorizontalMargin	XmHierarchy	XmNnumChildren	Composite
XmNopenFolderPixmap	XmHierarchy	XmNaccelerators	Core
XmNfigureMode	XmHierarchy	XmNancestorSensitive	Core
XmNverticalMargin	XmHierarchy	XmNbackground	Core
XmNbottomShadowColor	XmManager	XmNbackgroundPixmap	Core
XmNbottomShadowPixmap	XmManager	XmNborderColor	Core
XmNforeground	XmManager	XmNborderPixmap	Core
XmNhelpCallback	XmManager	XmNborderWidth	Core
XmNhighlightColor	XmManager	XmNcolormap	Core
XmNhighlightPixmap	XmManager	XmNdepth	Core
XmNinitialFocus	XmManager	XmNdestroyCallback	Core
XmNlayoutDirection	XmManager	XmNheight	Core
XmNnavigationType	XmManager	XmNinitialResourcesPersistent	Core
XmNpopupHandlerCallback	XmManager	XmNmappedWhenManaged	Core
XmNshadowThickness	XmManager	XmNscreen	Core
XmNstringDirection	XmManager	XmNsensitive	Core
XmNtopShadowColor	XmManager	XmNtranslations	Core
XmNtopShadowPixmap	XmManager	XmNwidth	Core

Resource	Inherited from	Resource	Inherited from
XmNtraversalOn	XmManager	XmNx	Core
XmNunitType	XmManager	XmNy	Core
XmNuserData	XmManager		

See Also

XmCreateObject(1), Composite(2), Constraint(2), XmContainer, Core(2), XmHierarchy, XmManager(2), XmOutline, XmBulletinBoard(2).

Name

XmWarningDialog –an unmanaged MessageBox as a child of a DialogShell.

Synopsis

Public Header:
<Xm/MessageB.h>

Instantiation:
widget = `XmCreateWarningDialog` (parent, name,...)

Functions/Macros:
`XmCreateWarningDialog`(), `XmMessageBoxGetChild`()

Description

An XmWarningDialog is a compound object created by a call to `XmCreateWarningDialog`() that an application can use to warn the user about a potentially hazardous action. A WarningDialog consists of a DialogShell with an unmanaged MessageBox widget as its child. The MessageBox resource XmNdialogType is set to XmDIALOG_WARNING.

A WarningDialog includes four components: a symbol, a message, three buttons, and a separator between the message and the buttons. By default, the symbol is an exclamation point. In Motif 1.2, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Cancel**, and **Help** by default.

Default Resource Values

A WarningDialog sets the following default values for MessageBox resources:

Name	Default
XmNdialogType	XmDIALOG_WARNING
XmNsymbolPixmap	xm_warning

Widget Hierarchy

When a WarningDialog is created with a specified name, the DialogShell is named name*_popup* and the MessageBox is called *name*.

See Also

`XmCreateObject` (1), `XmMessageBoxGetChild` (1), `XmDialogShell` (2), `XmMessageBox` (2).

Name

XmWorkingDialog –an unmanaged MessageBox as a child of a DialogShell.

Synopsis

Public Header:
<Xm/MessageB.h>

Instantiation:
`widget` = `XmCreateWorkingDialog` (parent, name,...)

Functions/Macros:
`XmCreateWorkingDialog`(), `XmMessageBoxGetChild`()

Description

An XmWorkingDialog is a compound object created by a call to `XmCreate-WorkingDialog`() that an application can use to warn the user that the current action is in progress, and likely to take some time. A WorkingDialog consists of a DialogShell with an unmanaged MessageBox widget as its child. The MessageBox resource XmNdialogType is set to XmDIALOG_WORKING.[1]

A WorkingDialog includes four components: a symbol, a message, three buttons, and a separator between the message and the buttons. By default, the symbol is an hourglass. In Motif 1.2, the default button labels can be localized. In the C locale, and in Motif 1.1, the PushButtons are labelled **OK**, **Cancel**, and **Help** by default.

Default Resource Values

A WorkingDialog sets the following default values for MessageBox resources:

Name	Default
XmNdialogType	XmDIALOG_WORKING
XmNsymbolPixmap	xm_working

Widget Hierarchy

When a WorkingDialog is created with a specified name, the DialogShell is named name*_popup* and the MessageBox is called *name*.

See Also

`XmCreateObject` (1), `XmMessageBoxGetChild` (1), `XmDialogShell` (2), `XmMessageBox` (2).

1.In the 1st and 2nd editions, this paragraph erroneously duplicated that of the XmWarningDialog description.