AIX Version 3 for
RISC System/6000™

Calls and Subroutines Reference: User Interface

Volume 3

SC23-21

RISC System/6000™

Calls and Subroutines Reference: User Interface

Volume 3

First Edition (March 1990)

This edition of the *AIX Calls and Subroutines Reference for IBM RISC System/6000* applies to IBM AIX Version 3 for RISC System/6000, Version 3 of IBM AIXwindows Environment/6000, IBM AIX System Network Architecture Services/6000, IBM AIX 3270 Host Connection Program/6000, IBM AIX 3278/79 Emulation/6000, IBM AIX Network Management/6000, and IBM AIX Personal Computer Simulator/6000 and to all subsequent releases of these products until otherwise indicated in new releases or technical newsletters.

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS MANUAL "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

IBM does not warrant that the contents of this publication or the accompanying source code examples, whether individually or as one or more groups, will meet your requirements or that the publication or the accompanying source code examples are error–free.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Any reference to an IBM licensed program in this publication is not intended to state or imply that you can use only IBM's licensed program. You can use any functionally equivalent program instead.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Department 997, 11400 Burnet Road, Austin, Texas 78758–3493. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks and Acknowledgements

The following trademarks and acknowledgements apply to this information:

AIX is a trademark of International Business Machines Corporation.

AIX/RT is a trademark of International Business Machines Corporation.

AIXwindows is a trademark of International Business Machines Corporation.

HP is a trademark of Hewlett Packard Inc.

HP–GL is a trademark of Hewlett–Packard Company.

IBM is a registered trademark of International Business Machines Corporation.

Operating System/2 and OS/2 are trademarks of International Business Machines Corporation.

OSF and OSF/Motif are trademarks of Open Software Foundation, Inc.

PAL is a trademark of International Business Machines Corporation.

Personal Computer AT and AT are trademarks of International Business Machines Corporation.

RISC System/6000 is a trademark of International Business Machines Corporation.

RT is a trademark of International Business Machines Corporation.

UNIX was developed and licensed by AT&T and is a registered trademark of AT&T Corporation.

Xstation Manager is a trademark of International Business Machines Corporation.

X Window System is a trademark of Massachusetts Institute of Technology.

X/OPEN is a trademark of X/OPEN Company Limited.

About This Book

This book provides information on AIXwindows classes, subroutines, and resource sets; Enhanced X–Windows subroutines, events, extensions, protocols and toolkit subroutines, and Curses and Extended Curses for use on the Advanced Interactive Executive Operating System (referred to in this text as AIX) for use on the IBM RISC System/6000.

This book is part of *AIX Calls and Subroutines Reference for IBM RISC System/6000*, SC23–2198. *AIX Calls and Subroutines Reference* is divided into the following four major sections:

- Volumes 1 and 2, *Calls and Subroutines Reference*: *Base Operating System*, contains reference information about the system calls, subroutines, functions, macros, and statements associated with AIX base operating system runtime services, communications services, and device services.
- Volumes 3 and 4, *Calls and Subroutines Reference*: *User Interface*, contain reference information about the AIXwindows widget classes, subroutines, and resource sets; the AIXwindows Desktop resource sets; the Enhanced X–Windows subroutines, macros, protocols, extensions, and events; the X–Window toolkit subroutines and macros; and the curses and extended curses subroutine libraries.
- Volume 5, *Calls and Subroutines Reference*: *Kernel Reference*, contains reference information about kernel services, device driver operations, file system operations subroutines, the configuration subsystem, the communications subsystem, the high function terminal (HFT) subsystem, the logical volume subsystem, the printer subsystem, and the SCSI subsystem.
- Volumes 6, *Calls and Subroutines Reference*: *Graphics*, contains reference information and example programs for the Graphics Library (GL) and the AIXwindows Graphics Support Library (XGSL) subroutines.

Who Should Use This Book

This book is intended for experienced programmers who understand the basic functions of the IBM RISC System/6000. To use this book effectively, you should be familiar with AIX or UNIX System V commands and subroutines, AIXwindows subroutines, and Enhanced X–Windows subroutines. If you are not already familiar with AIX or UNIX System V, refer to *AIX General Concepts and Procedures*.

How to Use This Book

Overview of Contents

This book contains the following alphabetically arranged sections on AIXwindows, Enhanced X–Windows, Curses and Extended Curses.

- **AIXwindows**
  - Classes
  - Subroutines
  - Resource Sets
  - Desktop Resource Sets
  - Window Management
- **Enhanced X–Windows**
  - Subroutines
  - Toolkit Subroutines

  - Protocols
  - Extensions
  - Events
- **Curses**
- **Extended Curses**

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, keywords, files, directories, and other items whose names are predefined by the system.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Related Publications

The following books contain information about or related to application programming interfaces:

- *AIX General Programming Concepts for IBM RISC System/6000*, Order Number SC23–2205.
- *AIX Communication Programming Concepts for IBM RISC System/6000*, Order Number SC23–2206.
- *AIX Kernel Extensions and Device Support Programming Concepts for IBM RISC System/6000*, Order Number SC23–2207.
- *AIX Files Reference for IBM RISC System/6000*, Order Number SC23–2200.
- *AIX User Interface Programming Concepts for IBM RISC System/6000*, Order Number SC23–2209.
- *IBM RISC System/6000 Problem Solving Guide*, Order Number SC23–2204.
- *XL C Language Reference for IBM AIX Version 3 for RISC System/6000*, Order Number SC09–1260.
- *XL C User's Guide for IBM AIX Version 3 for RISC System/6000*, Order Number SC09–1259.

Ordering Additional Copies of This Book

To order additional copies of this book, use Order Number SC23–2198.

Contents

AIXwindows Classes

ApplicationShell Widget Class

Purpose

The **ApplicationShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Shell.h>
```

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget
ArrowButton Gadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget

Description

The **ApplicationShell** widget class serves as the main top–level window for a client application. An application should only have more than one **ApplicationShell** if it implements multiple logical applications.

The **ApplicationShell** widget class inherits behavior and resources from the **Core**, **Composite**, **Shell**, **WMShell**, **VendorShell**, and **TopLevelShell** classes. The class pointer is **applicationShellWidgetClass**. The class name is **ApplicationShell**.

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowercase or uppercase, but including any underscores between words). The codes in the access column indicate whether 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**). The following resource set lists the resources of the **ApplicationShell** class:

- **ApplicationShell Resource Set**

Inherited Resources

The following resource sets list all of the resources inherited by the **ApplicationShell** widget class:

- **TopLevelShell Resource Set**
- **VendorShell Resource Set**
- **WMShell Resource Set**
- **Shell Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Shell.h

Related Information

The **Core** widget class, **Shell** widget class, **WMShell** widget class, **VendorShell** widget class, **TopLevelShell** widget class.

Composite Widget Class

Purpose

The **Composite** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Children

No children are supported.

Description

Composite widgets are intended to be containers for other widgets; they can have an arbitrary number of children. Their responsibilities (either implemented directly by the widget class or indirectly by the **Enhanced X–Windows** subroutines) include:

- Overall management of children from creation to destruction.
- Destruction of descendants when the **Composite** widget is destroyed.
- Physical arrangement (geometry management) of a displayable subset of managed children.
- Mapping and unmapping of a subset of the managed children. Instances of the **Composite** widgets need to specify about the order in which their children are kept. For example, an application may require a set of command buttons in some logical order grouped by function, and it may need buttons that represent file names to be kept in alphabetical order.

The **XmComposite** widget class inherits behavior and resources from the **Core** class. The class pointer is the **xmcompositeWidgetClass**. The class name is **XmComposite**.

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lowe case 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**).

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmCascadeButton** widget:

- **Composite Resource Set**
- **Core Resource Set**

The following procedure pointer in an **XmComposite** widget class instance is of the type **XtOrderProc**:

Cardinal (*XtOrderProc) (*widget*)

Widget *w*

w Specifies the widget.

The **Composite** widgets that allow clients to order their children (usually homogeneous boxes) can call their widget instance **insert_position** procedure from the class **insert_child** procedure to determine where a new child should go in its children array. A client application can apply different sorting criteria to widget instances of the composite class, passing in a different **insert_position** procedure when it creates each **Composite** widget instance.

The return value of the **insert_position** procedure indicates how many children should go before the widget. Returning *zero* indicates that the widget should go before all other children; returning **num_children** indicates that it should go after all other children. The default **insert_position** subroutine returns **num_children** and can be overridden by a specific **Composite** widget resource list or by the parameter list provided when the **Composite** widget is created.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **Core** widget class.

Constraint Widget Class

Purpose

The **Constraint** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include<Xm/Xm.h>

Children

No children are supported.

Description

The **Constraint** widget class maintains additional state data for each child. For example, client–defined constraints on the geometry of the child can be specified.

When a constrained composite widget defines **Constraint** resources, all children of the widget inherit those resources as their own. These **Constraint** resources are set and read the same way that other resources are defined for the child. This resource inheritance extends exactly one generation down; only the first–generation children of a constrained composite widget inherit the parent widget **Constraint** resources.

Because the **Constraint** resources are defined by the parent widgets and not the children, the child widgets never directly use the constraint resource data. **Constraint** resource data is instead used by the parents to attach child–specific data to children.

The **Constraint** widget class inherits behavior and resources from the **Composite** and **Core** classes. The class pointer is **constraintWidgetClass**. The class name is **Constraint**.

New Resources

The **Constraint** widget class defines no new resource sets.

Inherited Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for an resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The **Constraint** widget inherits behavior and resources from **Composite** and **Core**. The following resource set lists the resources of the **Constraint** class:

- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **Composite** widget class, **Core** widget class.

Core Widget Class

Purpose

The **Core** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Children

No children are supported.

Description

The **Core** widget class serves as the Enhanced X–Windows Toolkit base class for windowed widgets. To add support for the windowless widgets known as gadgets, three additional classes have been added above the **Core** widget in the class hierarchy. They are the **Object**, **RectObj**, and **WindowObj** classes. The **WindowObj** class is a synonym of the **Core** widget class that provides no added functionality, but was necessary for implementation reasons.

All widgets are built from the **Core** widget class. The class pointer is **widgetClass**. The class name is **Core**.

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.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 whether 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**). The following resource set lists the resources of the **Core** class:

- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **WindowObj** widget class, **Object** widget class, **RectObj** widget class.

Object Widget Class

Purpose

The **Object** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Description

The **Object** widget class is never instantiated. The sole purpose of this widget class is to act as a supporting superclass for other widget classes. The class pointer is **objectClass**. The class name is **Object**.

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference an resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for an resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **Object** widget class:

- **Object Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XtGetValues** subroutine, **XtSetValues** subroutine.

OverrideShell Widget Class

Purpose

The **OverrideShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Shell.h>
```

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
Label Widget
List Widget
PushButtonGadget Gadget
Separator Widget
Text Widget
ToggleButtonGadget Gadget
ArrowButtonGadget Gadget
CascadeButton Widget
DrawnButton Widget
LabelGadget Gadget
PushButton Widget
ScrollBar Widget
SeparatorGadget Gadget
ToggleButton Widget

Description

The **OverrideShell** widget class applies to shell windows (such as **PopupMenu** shells) that completely bypass the AIXwindows window manager.

The **OverrideShell** widget class inherits behavior and resources from the **Core**, **Composite**, and **Shell** classes. The pointer is **overrideShellWidgetClass**. The class name is **OverrideShell**.

New Resources

The **OverrideShell** widget class defines no new resources, but overrides the **XmNoverrideRedirect** and **XmNsaveUnder** resources in the **Shell** widget class.

Inherited Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference an resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.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 whether 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**). The following resource sets list all the resources inherited by the **OverrideShell** widget class:

- **Shell Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Shell.h

Related Information

The **Core** widget class, **Shell** widget class, **Composite** widget class.

RectObj Widget Class

Purpose

The **RectObj** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Children

No children are supported.

Description

The **RectObj** widget class serves as a supporting superclass for other widget classes. It is never instantiated.

The **RectObj** widget class is built from the **Object** widget class. The class pointer is **rectObjClass**. The class name is **RectObj**.

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.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 whether 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**). The following resource set lists the resources of the **RectObj** widget class.

- **RectObj Resource Set**

Inherited Resources

The **RectObj** widget class inherits behavior and an resource from the **Object** widget. The following resource set lists the inherited behavior and resource:

- **Object Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **Object** widget class.

Shell Widget Class

Purpose

The **Shell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Shell.h>
```

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButton Widget
DrawnButton Widget
FormDialog
LabelGadget Gadget
MessageDialog
PopupMenu
PushButton Widget
QuestionDialog
SelectionDialog
SeparatorGadget Gadget
ToggleButton Widget
WarningDialog
ArrowButtonGadget Gadget
BulletinBoardDialog
CascadeButtonGadget Gadget
ErrorDialog
Label Widget
List Widget
OptionMenu
PulldownMenu
PushButtonGadget Gadget
ScrollBar Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget
WorkingDialog

Description

The **Shell** widget class acts as a top–level widget (with only one managed child) that encapsulates the interaction with the AIXwindows window manager.

The **Shell** widget class inherits behavior and resources from the **Composite** and **Core** classes. The class pointer is **shellWidgetClass**. The class name is **Shell**.

New Resources

Setting the resource values for the inherited classes also sets resources for the **Shell** widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for an resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, but include any underscores between words). The codes in the access column indicate whether 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**). The following resource set lists the resources of the **Shell** widget class:

- **Shell Resource Set**

Inherited Resources

The following resource sets list all of the resources inherited by the **Shell** widget class:

- **Composite Resource Set**
- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Shell.h

Related Information

The **Composite** widget class, **Core** widget class.

TopLevelShell Widget Class

Purpose

The **TopLevelShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Shell.h>
```

Children

ApplicationShell Widget Class
ArrowButtonGadget Gadget
BulletinBoardDialog
CascadeButtonGadget Gadget
ErrorDialog
Label Widget
List Widget
OptionMenu
PopupMenu
PushButton Widget
QuestionDialog
SelectionDialog
SeparatorGadget Gadget
Text Widget
ToggleButtonGadget Gadget
VendorShell Widget Class
WMShell Widget Class
ArrowButton Widget
BulletinBoard Widget
CascadeButton Widget
DrawnButton Widget
FormDialog
LabelGadget Gadget
MessageDialog
OverrideShell Widget Class
PulldownMenu
PushButtonGadget Gadget
ScrollBar Widget
Separator Widget
Shell Widget Class
ToggleButton Widget
TransientShell Widget Class
WarningDialog
WorkingDialog

Description

The **TopLevelShell** widget class applies to normal top–level windows such as any additional top–level widgets that an application needs.

The class pointer is **topLevelShellWidgetClass**. The class name is **TopLevelShell**.

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference an resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for an resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, but include any underscores between words). The codes in the access column indicate whether 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**). The following resource set lists the resources of the **TopLevelShell** widget:

- **TopLevelShell Resource Set**

Inherited Resources

The following resource sets list all the resources inherited by the **TopLevelShell** widget class:

- **VendorShell Resource Set**
- **WMShell Resource Set**
- **Shell Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Shell.h

Related Information

The **Composite** widget class, **Core** widget class, **Shell** widget class, **WMShell** widget class, **VendorShell** widget class,

TransientShell Widget Class

Purpose

The **TransientShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Shell.h>
```

Children

ApplicationShell Widget Class
ArrowButtonGadget Gadget
CascadeButtonGadget Gadget
DrawnButton Widget
FormDialog
LabelGadget Gadget
MessageDialog
OverrideShell Widget Class
PulldownMenu
PushButtonGadget Gadget
ScrollBar Widget
Separator Widget
Shell Widget Class
ToggleButton Widget
TransientShell Widget Class
WarningDialog
WorkingDialog
ArrowButton Widget
BulletinBoard Widget
CascadeButton Widget
ErrorDialog
Label Widget
List Widget
OptionMenu
PopupMenu
PushButton Widget
QuestionDialog
SelectionDialog
SeparatorGadget Gadget
Text Widget
ToggleButtonGadgetGadget
VendorShell Widget Class
WMShell Widget Class

Description

The **TransientShell** widget class applies to shell windows that can be manipulated by the AIXwindows window manager, but are not allowed to be iconified separately. For example, **Dialog** boxes make no sense without their associated application. They are iconified by the window manager only if the main application shell is iconified.

The **TransientShell** widget class inherits behavior and resources from the **Core**, **Composite**, **Shell**, **WMShell**, and **VendorShell** classes. The class pointer is **transientShellWidgetClass**. The class name is **TransientShell**.

New Resources

The **TransientShell** widget class defines no new resources, but overrides the **XmNsaveUnder** resource in the **Shell** widget class and the **XmNtransient** resource in the **WMShell** widget class.

Inherited Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name of 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 lower case or upper case, 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**). The following resource sets contain a complete description of the resources inherited by the **TransientShell** widget class:

- **Core Resource Set**
- **Composite Resource Set**
- **Shell Resource Set**
- **VendorShell Resource Set**
- **WMShell Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Shell.h

Related Information

The **Composite** widget class, **Core** widget class, **Shell** widget class, **VendorShell** widget class, **WMShell** widget class.

VendorShell Widget Class

Purpose

The **VendorShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Shell.h>
```

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget

Description

The **VendorShell** widget class is used as a supporting superclass for all shell classes that are visible to the AIXwindows window manager and that do not have the **XmNoverrideRedirect** resource. This widget class contains the resources that maintain the AIXwindows window manager "look and feel." It also manages the AIXwindows window manager–specific communication needed by all **VendorShell** widget subclasses.

The **VendorShell** widget class inherits behavior and resources from the **Core**, **Composite**, **Shell**, and **WMShell** classes. The class pointer is **vendorShellClass**. The class name is **VendorShell**.

Subroutines

- **XmActivateProtocol**
- **XmAddProtocolCallback**
- **XmAddProtocols**
- **XmDeactivateProtocol**
- **XmAtomToName**
- **XmInternAtom**

- **XmIsMotifWMRunning**
- **XmRemoveProtocolCallback**
- **XmRemoveProtocols**
- **XmSetProtocolHooks**

New Resources

Setting the resource values for the inherited classes also sets resources 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 an resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, but include any underscores between words). The codes in the access column indicate whether 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**). The following resource sets list the resources of the **VendorShell** widget class:

- **VendorShell Resource Set**

Inherited Resources

The following superclasses contain a complete description of resources inherited by the **VendorShell** widget class:

- **WMShell Resource Set**
- **Shell Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Shell.h

Related Information

The **Composite** widget class, **Core** widget class, **Shell** widget class, **WMShell** widget class, **XmActivateProtocol** subroutine, **XmAddProtocolCallback** subroutine, **XmAddProtocols** subroutine, **XmDeactivateProtocol** subroutine, **XmAtomToName** subroutine, **XmInternAtom** subroutine, **XmIsMotifWMRunning** subroutine, **XmRemoveProtocolCallback** subroutine,**XmRemoveProtocols** subroutine, **XmSetProtocolHooks** subroutine.

WMShell Widget Class

Purpose

The **WMShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Shell.h>
```

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
Label Widget
List Widget
PushButtonGadget Gadget
Separator Widget
Text Widget
ToggleButtonGadget Gadget
ArrowButtonGadget Gadget
CascadeButton Widget
DrawnButton Widget
LabelGadget Gadget
PushButton Widget
ScrollBar Widget
SeparatorGadget Gadget
ToggleButton Widget

Description

The **WMShell** widget class serves as a top–level widget that encapsulates the interaction with the AIXwindows window manager.

The **WMShell** widget class inherits behavior and resources from the **Core**, **Composite**, and **Shell** classes. The class pointer is **wmShellWidgetClass**. The class name is **WMShell**.

New Resources

Setting the resource values for the inherited classes also sets resources 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 an resource in a **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, but include any underscores between words). The codes in the access column indicate whether 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**). The following resource set lists the resources of the **WMShell** widget class:

- **WMShell Resource Set**

Inherited Resources

The following superclasses contain a complete description of resources inherited by the **WMShell** widget class:

- **Shell Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Shell.h

Related Information

The **Core** widget class, **Composite** widget class, **Shell** widget class.

WindowObj Widget Class

Purpose

The **WindowObj** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Children

No children are supported.

Description

The **WindowObj** widget class is an internal **Enhanced X–Windows** widget class. This widget class is a synonym of the **Core** widget class that provides no added functionality but was necessary for implementation reasons.

The **WindowObj** widget class inherits behavior and resources from the **Object** and **RectObj** classes. The class pointer is **windowObjClass**. The class name is **WindowObj**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **Object** widget class, **Core** widget class, **RectObj** widget class.

XmArrowButton Widget Class

Purpose

The **ArrowButton** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ArrowB.h>

Children

No children are supported.

Description

An **ArrowButton** widget consists of a directional arrow surrounded by a border shadow. When the widget is selected, the shadow moves to give the appearance that the **ArrowButton** widget has been pressed in. When the **ArrowButton** widget is unselected, the shadow moves to give the appearance that the **ArrowButton** widget is released, or out.

The **XmArrowButton** widget class inherits behavior and resources from the **Core** and **XmPrimitive** classes. The class pointer is **xmArrowButtonWidgetClass**. The class name is **XmArrowButton**.

Subroutines

- **XmCreateArrowButton**
- **XtCreateWidget** subroutine

New Resources

Setting the resource values for the inherited classes to also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for an resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **ArrowButton** widget:

- **XmArrowButton Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **ArrowButton** widget:

- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int     reason;
        XEvent  * event;
} XmAnyCallbackStruct;
```

reason Indicates why the callback was invoked.

event Points to the **XEvent** that triggered the callback. This event will be **NULL** for the **XmNactivateCallback** if the callback was triggered when the **Primitive** resource **XmNtraversalOn** was **True** or if the callback was accessed through the **ArmAndActivate** action routine.

Behavior

<Btn1Down>: This action causes the arrow to be armed, and the shadow to be drawn in the selected state. The callbacks for **XmNarmCallback** are called.

<Btn1Up>: If the mouse button release occurs when the pointer is within the **ArrowButton** widget, the arrow shadows are redrawn in the unselected state. The callbacks for **XmNactivateCallback** are called, followed by callbacks for **XmNdisarmCallback**.

If the mouse button release occurs when the pointer is outside the **ArrowButton** widget, the callbacks for **XmNdisarmCallback** are called.

<Leave Window>: If the mouse button is pressed and the cursor leaves the widget window, the arrow shadow is redrawn in its unselected state.

<Enter Window>: If the mouse button is pressed and the cursor leaves and re–enters the widget window, the arrow shadow is drawn in the same manner as when the button was first armed.

Default Translations

<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()** **Disarm()**
<Key>Return:	**ArmAndActivate()**
<Key>Space:	**ArmAndActivate()**
<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**

Keyboard Traversal

For information on keyboard traversal, see **XmPrimitive** and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ArrowB.h

Related Information

The **Core** widget class, **XmCreateArrowButton** subroutine, **XmPrimitive** widget class, **XtCreateWidget** subroutine.

XmArrowButtonGadget Gadget Class

Purpose

The **ArrowButtonGadget** gadget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ArrowBG.h>

Children

No children are supported.

Description

An **ArrowButtonGadget** gadget consists of a directional arrow surrounded by a border shadow. When the gadget is selected, the shadow moves to give the appearance that the **ArrowButtonGadget** gadget has been pressed in. When it is unselected, the shadow moves to give the appearance that the button is released, or out.

The **ArrowButtonGadget** gadget class inherits behavior and resources from the **Object**, **RectObj**, and **XmGadget** classes. The class pointer is **xmArrowButtonGadgetClass**. The class name is **XmArrowButtonGadget**.

Subroutines

- **XmCreateArrowButtonGadget**
- **XtCreateWidget**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmArrowButtonGadget** gadget class:

- **XmArrowButtonGadget Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **ArrowButtonGadget** gadget class:

- **XmGadget Resource Set**
- **RectObj Resource Set**
- **Object Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct {
        int     reason;
        XEvent  *event;
} XmAnyCallbackStruct;
```

reason — Indicates why the callback was invoked.

event — Points to the **XEvent** that invoked the callback. This event is **NULL** for the **XmNactivateCallback** resource if the callback was triggered when the **XmNtraversalOn** resource of the **Primitive** was **True** or if the callback was accessed through the **ArmAndActivate** action routine.

Behavior

<Btn1Down>: This action causes the arrow to be armed, and the shadow to be drawn in the selected state. The callbacks for the **XmNarmCallback** resource are called.

<Btn1Up>: If the mouse button release occurs when the pointer is within the **ArrowButtonGadget** gadget, the arrow shadows are redrawn in the unselected state. The callbacks for the **XmNactivateCallback** resource are called, followed by callbacks for the **XmNdisarmCallback** resource.

If the mouse button release occurs when the pointer is outside the **ArrowButtonGadget** gadget, the callbacks for the **XmNdisarmCallback** resource are called.

<Leave Window>: If the mouse button is pressed and the cursor leaves the widget window, the arrow shadow is redrawn in its unselected state.

<Enter Window>: If the mouse button is pressed and the cursor leaves and re–enters the widget window, the arrow shadow is drawn in the same manner as when the button was first armed.

Default Translations

<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()**
	Disarm()
<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**

Keyboard Traversal

For information on keyboard traversal, see **XmGadget** and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ArrowBG.h

Related Information

The **Object** widget class, **RectObj** widget class, **XmCreateArrowButtonGadget** subroutine, **XtCreateWidget** subroutine, **XmGadget** gadget class.

XmBulletinBoard Widget Class

Purpose

The **BulletinBoard** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/BulletinB.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget

Description

A **BulletinBoard** widget is a composite widget that provides simple geometry management for children widgets. This widget does not force positioning on its children, but can be set to reject geometry requests that would result in overlapping children. The **BulletinBoard** widget is the base widget for most dialog widgets, and is also used as a general container widget.

Modal and modeless dialogs are implemented as collections of widgets including **DialogShell** widgets, **BulletinBoard** widgets (or subclass children of the shell), and various dialog components (such as buttons and labels) that are children of the **BulletinBoard** widget. The **BulletinBoard** widget defines callback routines useful for dialogs (focus, map, unmap). If its parent is a **DialogShell** widget, the **BulletinBoard** widget passes title and input mode (based on dialog style) information to the parent, which is responsible for appropriate communication with the window manager.

The **XmBulletinBoard** widget class inherits behavior and resource from the **Core**, **Composite**, **Constraint**, and **XmManager** classes.The class pointer is **xmBulletinBoardWidgetClass**. The class name is **XmBulletinBoard**.

Subroutines

The following subroutines create an instance of a **BulletinBoard** widget and return the associated widget ID:

- **XmCreateBulletinBoard**
- **XmCreateBulletinBoardDialog**

New Resources

Setting the resource values for the inherited classes also sets resource for this widget. To reference an resource by name or by class in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resource of the **XmBulletinBoard** class:

- **XmBulletinBoard Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resource inherited by the **BulletinBoard** widget:

- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback routine:

```
typedef struct
{
int         reason;
XEvent      * event;
} XmAnyCallbackStruct;
```

reason — Is set to the value that corresponds to the type of selection that invoked this callback routine.

event — Points to the **XEvent** that invoked the callback routine.

Behavior

The **BulletinBoard** widget behavior is summarized below:

<Cancel Button Activated>: When the Cancel button is pressed, the "activate" callback routines of the Cancel pushbutton are invoked.

<Default Button Activated> or **<Key>Return**: When the Default button is pressed, the "activate" callback routines of the Default pushbutton are invoked.

<Help Button Activated> or **<Key>F1:** When the help button or **Function key 1** is pressed, the callback routines for **XmNhelpCallback** are invoked.

<FocusIn>: When a **FocusIn** *event* is generated on the widget window, the callback routines for **XmNfocusCallback** are invoked.

<MapWindow>: When a **BulletinBoard** widget that is the child of a **DialogShell** widget is mapped, the callback routines for **XmNmapCallback** are invoked. When a **BulletinBoard** widget that is not the child of a **DialogShell** widget is mapped, the callback routines are not invoked.

<UnmapWindow>: When a **BulletinBoard** widget that is the child of a **DialogShell** widget is unmapped, the callback routines for **XmNunmapCallback** are invoked. When a **BulletinBoard** widget that is not the child of a **DialogShell** widget is unmapped, the callback routines are not invoked.

Default Translations

The default translations defined for **XmBulletinBoard** widgets are:

<EnterWindow>:	**Enter()**
<FocusIn>:	**FocusIn()**
<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()**
<Key>F1:	**Help()**
<Key>Return:	**Return()**
<Key>KP_Enter:	**Return()**

Default Accelerators

The default accelerator translations added to descendants of a **BulletinBoard** widget (if the parent of the **BulletinBoard** widget is a **DialogShell** widget) are:

#override

<Key>F1:	**Help()**
<Key>Return:	**Return()**
<Key>KP_Enter:	**Return()**

Keyboard Traversal

By default, if the parent of a **BulletinBoard** widget is a **DialogShell** widget, the **BulletinBoard** widget uses the Return key to activate the Default button. It installs accelerators on all descendant widgets to make this possible. These accelerators disable the normal keyboard traversal behavior of the Return key. This traversal behavior can be restored (and the Default button behavior disabled) by replacing the **BulletinBoard** widget default accelerators with an alternate set of translations which do not specify the Return action. The description of the **Manager** widget has more information on keyboard traversal.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/BulletinB.h

Related Information

The **Core** widget class, **Constraint** widget class, **Composite** widget class, **XmCreateBulletinBoard** subroutine, **XmCreateBulletinBoardDialog** subroutine, **XmDialogShell** widget class, **XmManager** widget class.

XmCascadeButton Widget Class

Purpose

The **CascadeButton** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/CascadeB.h>

Children

No children are supported.

Description

A **CascadeButton** widget links two **MenuPane** widgets or a **MenuBar** widget to a **MenuPane** widget.

This widget is used in menu systems and must have a **RowColumn** widget with its **XmNrowColumnType** resource set to the **XmMENU_BAR**, **XmMENU_POPUP**, **XmMENU_PULLDOWN**, or **XmMENU_OPTION** value.

It is the only widget that can have a **Pulldown MenuPane** attached to it as a submenu. The submenu is displayed when this widget is activated within a **MenuBar**, a **PopupMenu**, or a **PulldownMenu**. Its visuals can include a label or pixmap and a cascading indicator when it is in a **Popup** or **Pulldown MenuPane**; when it is in a **MenuBar**, its visuals are limited to a label or a pixmap.

The default behavior associated with a **CascadeButton** widget depends on the type of menuing system in which it resides. By default, controls the behavior of the **CascadeButton** widget if it resides in a **PulldownMenu** or a **MenuBar**; mouse button two controls the behavior of the **CascadeButton** widget if it resides in a **PopupMenu**. The actual mouse button used is determined by its **RowColumn** parent widget.

A **CascadeButton** widget's visuals differ from most other button widgets. When the button becomes armed, its visuals change from a two–dimensional appearance to a three–dimensional appearance, and it displays the submenu that has been attached to it. If no submenu is attached, it simply changes its visuals.

When a **CascadeButton** widget within a **Pulldown** or **Popup MenuPane** is armed as the result of the user moving the mouse pointer into the widget, it does not immediately display its submenu. Instead, it waits a short amount of time to see if the arming was temporary (i.e., the user was simply passing through the widget), or whether the user really wanted the submenu posted. This time delay is configurable through the **XmNmappingDelay** resource.

The **CascadeButton** widget provides a single mechanism for activating the widget from the keyboard. This mechanism is referred to as a keyboard mnemonic. If a mnemonic has been specified for the widget, the user can activate the **CascadeButton** widget by typing the mnemonic while the **CascadeButton** widget is visible. If the **CascadeButton** widget is in a **MenuBar**, the **meta** key must be pressed with the mnemonic. Mnemonics are typically used to interact with a menu through the keyboard interface.

If the **CascadeButton** widget is in a **Pulldown** or **Popup Menupane** and a submenu is attached, the **XmNmarginBottom**, **XmNmarginRight**, and **XmNmarginTop** resources enlarge to accommodate the **XmNcascadePixmap** resource. The **XmNmarginWidth** resource defaults to six if this resource is in a **Menubar**; otherwise, it takes the defaults of **Label**, which is two.

The **CascadeButton** widget inherits behavior and resources from **Core**, **XmPrimitive**, and **XmLabel** classes. The class pointer is **xmCascadeButtonWidgetClass**. The class name is the **XmCascadeButton**.

Subroutines

- **XmCreateCascadeButton**
- **XmCascadeButtonHighlight**
- **XmCreateMenuBar**
- **XmCreatePulldownMenu**
- **XmCreatePopupMenu**
- **XtCreateWidget**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **CascadeButton** widget:

- **XmCascadeButton Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **CascadeButton** widget:

- **XmLabel Resource Set**
- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int     reason;
        XEvent  * event;
} XmAnyCallbackStruct;
```

reason Is set to the value that corresponds to the type of selection that invoked this callback.

event Points to the **XEvent** that invoked the callback or is **NULL** if this callback was not triggered due to an **XEvent**.

Behavior

The default behavior associated with a **CascadeButton** widget depends on whether the button is part of a **PopupMenu** system, a **PulldownMenu** system or an **OptionMenu** system. The **RowColumn** widget parent determines the mouse button that is used through its **XmNrowColumnType** and **XmNwhichButton** resources.

Default PopupMenu System

Btn3Down<EnterWindow>: This action arms the **CascadeButton** widget and posts the associated submenu after a short delay.

Btn3Down<LeaveWindow>: The action that takes place depends on whether the mouse pointer has moved into the submenu associated with this **CascadeButton** widget. If the mouse pointer has moved into the submenu, this event is ignored. If not, the **CascadeButton** widget is disarmed and its submenu is unposted.

<Btn3Up>: This action posts the submenu attached to the **CascadeButton** widget and enables keyboard traversal within the menu. If the **CascadeButton** widget does not have a submenu attached, this event activates the **CascadeButton** widget and unposts the menu.

<Btn3Down>: This action disables traversal for the menu and returns the user to drag mode in which the menu is manipulated using the mouse. The submenu associated with this **CascadeButton** widget is posted.

<Key>Return: This event posts the submenu attached to the **CascadeButton** widget if keyboard traversal is enabled in the menu. If the **CascadeButton** widget does not have a submenu attached, this event activates the **CascadeButton** widget and unposts the menu.

Default MenuBar

<Btn1Down>: This event arms both the **CascadeButton** widget and the **MenuBar** and posts the associated submenu. If the menu is already active, this event disables traversal for the menu and returns the user to the mode where the menu is manipulated using the mouse.

Btn1Down<EnterWindow>: This event unposts any visible **MenuPanes** if they are associated with a different **MenuBar** entry, arms the **CascadeButton** widget, and posts the associated submenu.

Btn1Down<LeaveWindow>: This event disarms the **CascadeButton** widget if the submenu associated with it is not currently posted or if there is no submenu associated with this **CascadeButton** widget. Otherwise, this event is ignored.

<Btn1Up>: This event posts the submenu attached to the **CascadeButton** widget and enables keyboard traversal within the menu. If the **CascadeButton** widget does not have a submenu attached, this event activates the **CascadeButton** widget and unposts the menu.

<Key>Return: This event posts the submenu attached to the **CascadeButton** widget if keyboard traversal is enabled in the menu. If the **CascadeButton** widget does not have a submenu attached, the **CascadeButton** widget is activated, and the menu is unposted.

Default PulldownMenu System from a MenuBar

Btn1Down<EnterWindow>: This event arms the **CascadeButton** widget, and after a short delay, posts the associated submenu.

Btn1Down<LeaveWindow>: The event is ignored if the mouse pointer has moved into the submenu. In all other cases, the **CascadeButton** widget is disarmed and its submenu unposted.

<Btn1Up>: This event posts the submenu attached to the **CascadeButton** widget and enables keyboard traversal within the menu. If the **CascadeButton** widget does not have a submenu attached, this event selects the **CascadeButton** widget and unposts the menu.

<Btn1Down>: This event disables traversal for the menu and returns the user to the drag mode. The submenu associated with this **CascadeButton** widget is posted.

<Key>Return: This event posts the submenu attached to the **CascadeButton** widget if keyboard traversal is enabled in the menu. If the **CascadeButton** widget does not have a submenu attached, this event activates the **CascadeButton** widget and unposts the menu.

Default Option Menu System

<Btn2Down>: This event arms the **CascadeButton** widget and posts the associated submenu.

<Key>Return: This event posts the associated submenu and enables traversal within the menu.

Default Translations

Default translations for **CascadeButton** in a **MenuBar** are:

<BtnDown>:	**MenuBarSelect()**
<EnterWindow>:	**MenuBarEnter()**
<LeaveWindow>:	**MenuBarLeave()**
<BtnUp>:	**DoSelect()**
<Key>Return:	**KeySelect()**
<Key>Escape:	**CleanupMenuBar()**

Default translations for **CascadeButton** in a **Popup** or **Pulldown MenuPane** are:

<BtnDown>:	**StartDrag()**
<EnterWindow>:	**DelayedArm()**
<LeaveWindow>:	**CheckDisarm()**
<BtnUp>:	**DoSelect()**
<Key>Return:	**KeySelect()**
<Key>Escape:	**MenuShellPopdownDone()**

Default translations for **CascadeButton** in an **OptionMenu** are:

<BtnDown>:	**CheckArmAndPost()**
<Key>Return:	**KeySelect()**

Keyboard Traversal

<Unmap>:	**Unmap()**
<FocusOut>:	**FocusOut()**
<FocusIn>:	**FocusIn()**
<Key>space:	**Noop()**
<Key>Left:	**MenuTraverseLeft()**
<Key>Right:	**MenuTraverseRight()**
<Key>Up:	**MenuTraverseUp()**
<Key>Down:	**MenuTraverseDown()**
<Key>Home:	**Noop()**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/CascadeB.h

Related Information

The **Core** widget class, **XmCreateCascadeButton** subroutine, **XmCascadeButtonHighlight** subroutine, **XmCreateMenuBar** subroutine, **XmCreatePulldownMenu** subroutine, **XmCreatePopupMenu** subroutine, **XmLabel** widget class, **XmPrimitive** widget class, **XmRowColumn** widget class, **XtCreateWidget** subroutine.

XmCascadeButtonGadget Gadget Class

Purpose

The **CascadeButtonGadget** gadget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/CascadeBG.h>

Children

No children are supported.

Description

An **XmCascadeButtonGadget** gadget class links two **MenuPane** widgets or an **OptionMenu** widget to a **MenuPane** widget.

This gadget is used in menu systems and must have a **RowColumn** parent with its **XmNrowColumnType** resource set to the **XmMENU_POPUP** value, the **XmMENU_PULLDOWN** value, or the **XmMENU_OPTION** value.

This is the only gadget that can have a **Pulldown MenuPane** attached to it as a submenu. The submenu is displayed when this gadget is activated within a **PopupMenu**, a **PulldownMenu**, or a **OptionMenu**. When this gadget is in an **XmPopupMenu** widget or an **XmPulldownMenu** widget, its visuals can include a label or pixmap and a cascading indicator. When it is in an **XmOptionMenu** widget, its visuals can include only a label or a pixmap.

The default behavior associated with a **CascadeButtonGadget** gadget depends on the type of menu system in which it resides. By default, controls the behavior of the **CascadeButtonGadget** when it resides in a **PulldownMenu** or an **OptionMenu**; controls the behavior of the **CascadeButtonGadget** gadget when it resides in a **PopupMenu**. The actual mouse button used is determined by its **RowColumn** parent.

A **CascadeButtonGadget** gadget's visuals differ from the visuals of most other button widgets. When the button becomes armed, its visuals change from a two–dimensional to a three–dimensional look, and it displays the submenu that has been attached to it. If no submenu is attached, it simply changes its visuals.

When a **CascadeButtonGadget** gadget within a **PulldownMenu** or a **PopupMenu** is armed as a result of the user moving the mouse pointer into the widget, it does not immediately display its submenu. Instead, it waits a short amount of time to see if the arming was temporary (in other words, the user was simply passing through the widget), or whether the user really wanted the submenu posted. This time delay is configurable through the **XmNmappingDelay** resource.

The **CascadeButtonGadget** gadget provides a single mechanism for activating the gadget from the keyboard. This mechanism is referred to as keyboard mnemonic. If a mnemonic has been specified for the gadget, the user can activate it by simply typing the mnemonic while the **CascadeButtonGadget** gadget is visible. Mnemonics are typically used to interact with a menu through the keyboard interface.

If the **CascadeButtonGadget** gadget is in a **Pulldown** or a **Popup MenuPane** and there is a submenu attached, the **XmNmarginBottom**, **XmNmarginRight**, and **XmNmarginTop** resources enlarge to accommodate the **XmNcascadePixmap** resource.

The **XmCascadeButtonGadget** gadget class inherits behavior and resources from the **XmObject, XmRectObj, XmGadget**, and **XmLabelGadget** classes. The class pointer is **xmCascadeButtonGadgetClass**. The class name is **XmCascadeButtonGadget**.

Subroutines

- **XmCascadeButtonHighlight**
- **XmCreateCascadeButtonGadget**
- **XmCreatePulldownMenu**
- **XmCreatePopupMenu**
- **XmCreateOptionMenu**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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)**. The following resource set lists the resources of the **XmCascadeButtonGadget** gadget class:

- **XmCascadeButtonGadget Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmCascadeButtonGadget** gadget class:

- **XmLabelGadget Resource Set**
- **XmGadget Resource Set**
- **RectObj Resource Set**
- **Object Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
  int      reason;
  XEvent * event;
} XmAnyCallbackStruct;
```

reason Indicates why the callback was invoked.

event Points to the **XEvent** that triggered the callback or is **NULL** if this callback was not triggered due to an **XEvent** within a widget.

Behavior

The default behavior associated with a **CascadeButtonGadget** gadget depends on whether the button is part of a **PopupMenu** system, a **Pulldown MenuPane** in a **MenuBar**, or an **OptionMenu** system. The **RowColumn** widget determines the mouse button used through its **XmNrowColumnType** and **XmNwhichButton** resources.

Default PopupMenu System

Btn2Down <EnterWindow>: This action arms the **CascadeButtonGadget** gadget and posts the associated submenu after a short delay.

Btn2Down <LeaveWindow>: The action that takes place depends on whether the mouse pointer has moved into the submenu associated with this **CascadeButtonGadget** gadget. If the mouse pointer has moved into the submenu, this event is ignored. If not, the **CascadeButtonGadget** gadget is disarmed and its submenu unposted.

<Btn2Up>: This action posts the submenu attached to the **CascadeButtonGadget** gadget and enables keyboard traversal within the menu. If the **CascadeButtonGadget** gadget does not have a submenu attached, this event selects the **CascadeButtonGadget** gadget and unposts the menu.

<Btn2Down>: This action disables traversal for the menu and returns the user to drag mode in which the menu is manipulated using the mouse. The submenu associated with this **CascadeButtonGadget** gadget is posted.

<Key> Return: This event posts the submenu attached to the **CascadeButtonGadget** gadget if keyboard traversal is enabled in the menu. If the **CascadeButtonGadget** gadget does not have a submenu attached, this event activates the **CascadeButtonGadget** gadget and unposts the menu.

Default Pulldown MenuPane System from a MenuBar or from an OptionMenu

Btn1Down <EnterWindow>: This event arms the **CascadeButtonGadget** gadget and posts the associated submenu after a short delay.

Btn1Down <LeaveWindow>: The event is ignored if the mouse pointer has moved into the submenu. In all other cases, the **CascadeButtonGadget** gadget is disarmed and its submenu unposted

<Btn1Up>: This event posts the submenu attached to the **CascadeButtonGadget** gadget and enables keyboard traversal within the menu. If the **CascadeButtonGadget** gadget does not have a submenu attached, this event activates the **CascadeButtonGadget** gadget and unposts the menu.

<Btn1Down>: This event disables traversal for the menu and returns the user to the drag mode. The submenu associated with this **CascadeButtonGadget** gadget is posted.

<Key> Return: This event posts the submenu attached to the **CascadeButtonGadget** gadget if keyboard traversal is enabled in the menu. If the **CascadeButtonGadget** gadget does not have a submenu attached, then this event selects the **CascadeButtonGadget** gadget and unposts the menu.

Default OptionMenu System

<Btn1Down>: This event arms the **CascadeButtonGadget** gadget and posts the associated submenu.

<Key> Return: This event posts the associated submenu and enables traversal within the menu.

Keyboard Traversal

The **XmRowColumn** widget and its sections on behavior and default translations contain information on keyboard traversal.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/CascadeBG.h

Related Information

The **RectObj** widget class, **Object** widget class, **XmCascadeButtonHighlight** subroutine, **XmCreateCascadeButtonGadget** subroutine, **XmCreatePulldownMenu** subroutine, **XmCreatePopupMenu** subroutine, **XmCreateOptionMenu** subroutine, **XmGadget** widget class, **XmLabelGadget** widget class, **XmRowColumn** widget class.

XmCommand Widget Class

Purpose

The **Command** widget class.

Libraries

AIXwindows Library (**libXm.a**)

AIXwindows Library (**libIM.a**)

Syntax

#include <Xm/Command.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget

Description

A **Command** widget is a special–purpose composite widget for command entry that provides a built–in command history mechanism. The **Command** widget includes a command line text input field, a command line prompt, and a command history list region.

Note: You should be aware of the proper usage of the **XmText** widget class before using this widget class.

One additional **WorkArea** child widget can be added to the **Command** widget after creation.

As each command is entered, it is automatically added to the end of the command history list and made visible. This does not change the selected item in the list, if there is one.

Many of the new resources specified for the **Command** widget are actually **XmSelectionBox** resources that have been renamed for clarity and ease of use.

The **XmCommand** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, **XmManager**, **XmBulletinBoard**, and **XmSelectionBox** classes. The class pointer is **xmCommandWidgetClass**. The class name is **XmCommand**.

Subroutines

- **XmCreateCommand**
- **XmCommandAppendValue**
- **XmCommandError**
- **XmCommandGetChild**
- **XmCommandSetValue**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for an resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmCommandWidget** widget class:

- **XmCommand Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmCommand** widget class:

- **XmSelectionBox Resource Set**
- **XmBulletinBoard Resource Set**
- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback routine:

```
typedef struct
{
  int reason;
  XEvent * event;
  XmString value;
  int length;
} XmCommandCallbackStruct
```

reason	Indicates why the callback routine was invoked.
event	Points to the **XEvent** that triggered the callback routine.
value	Specifies the **XmString** in the **CommandArea**.
length	Specifies the size of the command in **XmString**.

Behavior

Command behavior is summarized as follows:

<Key>: When any change is made to the text edit widget, the callback routines for the **XmNcommandChangedCallback** resource are invoked.

<Key>Return: When the Return key is pressed, the callback routines for the **XmNcommandEnteredCallback** resource and the **XmNcommandChangedCallback** resource are invoked.

<Key>Up or **<Key>Down:** When the up or down key is pressed within the **Text** subwidget of the **XmCommand** subroutine, the text value is replaced with the previous or next item in the **List** subwidget. The selected item in the list is also changed to the previous or the next item. The callback routines for the **XmNcommandChangedCallback** resource are invoked.

<DoubleClick>: When an item in the **List** subwidget is double–clicked, that item is selected and added to the end of the list in one action. The callback routines for the **XmNcommandEnteredCallback** resource and the **XmNcommandChangedCallback** resource are invoked.

<Key>F1: When the **Function Key 1** is pressed, the callback routines for the **XmNhelpCallback** resource are invoked.

<FocusIn>: When a **FocusIn** *event* is generated on the widget window, the callback routines for the **XmNfocusCallback** resource are invoked.

<MapWindow>: When a **Command** widget that is the child of a **DialogShell** widget is mapped, the callback routines for the **XmNmapCallback** resource are invoked. When a **Command** widget that is not the child of a **DialogShell** widget is mapped, the callback routines are not invoked.

<UnmapWindow>: When a **Command** widget that is the child of a **DialogShell** widget is unmapped, the callback routines for the **XmNunmapCallback** resource are invoked. When a **Command** widget that is not the child of a **DialogShell** widget is unmapped, the callback routines are not invoked.

Default Translations

The **Command** widget inherits default translations from the **SelectionBox** widget.

Default Accelerators

The default accelerators added to descendants of a **BulletinBoard** widget if the parent of the **BulletinBoard** widget is a **DialogShell** are as follows:

```
#override
<Key>F1:                Help()
<Key>Return:            Return()
<Key>KP_Enter:          Return()
```

Default Text Accelerators

The default text accelerators inherited from the **SelectionBox** widget are:

```
#override
<Key>:                  UpOrDown(0)
<Key>Down:              UpOrDown(1)
<Key>F1:                Help()
<Key>Return:            Return()
<Key>KP_Enter:          Return()
```

Keyboard Traversal

For information on keyboard traversal, see **XmManager** and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Command.h

Related Information

The **XmCreateCommand** subroutine, **XmCommandAppendValue** subroutine, **XmCommandError** subroutine, **XmCommandGetChild** subroutine, **XmCommandSetValue** subroutine, **XmDialogShell** widget class, **XmSelectionBox** widget class, **XmBulletinBoard** widget class, **XmManager** widget class, **Constraint** widget class, **Composite** widget class, **Core** widget class.

XmDialogShell Widget Class

Purpose

The **DialogShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/DialogS.h>

Children

ArrowButton Widget
BulletinBoardDialog
Command Widget
ErrorDialog
FormDialog
InformationDialog
MenuShell Widget
MessageDialog
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
Separator Widget
ToggleButton Widget
WorkingDialog
BulletinBoard Widget
CascadeButton Widget
DrawingArea Widget
Form Widget
Frame Widget
Label Widget
MessageBox Widget
PanedWindow Widget
QuestionDialog
Scale Widget
ScrolledWindow Widget
SelectionDialog
Text Widget
WarningDialog

Description

Modal and modeless dialogs use the **DialogShell** widget as the **Shell** parent. The **DialogShell** widgets cannot be iconified. Instead, all secondary **DialogShell** widgets associated with an **ApplicationShell** widget are iconified and de–iconified as a group with the primary widget.

The client indirectly manipulates the **DialogShell** widget by the convenience interfaces during creation, and it can directly manipulate its **BulletinBoard** widget–derived child. Much of the functionality of the **DialogShell** widget assumes its child is a **BulletinBoard** subclass, although it can potentially stand alone.

The **DialogShell** widget inherits behavior and resources from the **Core**, **Composite**, **Shell**, **WMShell**, **VendorShell**, and **TransientShell** classes. The class pointer is **xmDialogShellWidgetClass**. The class name is **XmDialogShell**.

Subroutine

- **XmCreateDialogShell**

New Resources

The **DialogShell** widget defines no new resources, but overrides the **XmNdeleteResponse** resource in the **VendorShell** class.

Inherited Resources

Setting the resource values for the inherited classes also sets resources 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**). The following resource sets list the resources of the **DialogShell** widget:

- **VendorShell Resource Set**
- **WMShell Resource Set**
- **Shell Resource Set**
- **Core Resource Set**

File

/usr/include/Xm/DialogS.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **TransientShell** widget class, **Composite** widget class, **VendorShell** widget class, **WMShell** widget class, **Shell** widget class, **Core** widget class, **XmBulletinBoard** widget class, **XmCreateDialogShell** subroutine.

XmDrawingArea Widget Class

Purpose

The **DrawingArea** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/DrawingA.h>

Children

ArrowButton Widget
CascadeButton Widget
DrawnButton Widget
Frame Widget
MenuShell Widget
PanedWindow Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
Text Widget
BulletinBoard Widget
Command Widget
Form Widget
Label Widget
MessageBox Widget
PushButton Widget
Scale Widget
ScrolledWindow Widget
Separator Widget
ToggleButton Widget

Description

The **DrawingArea** widget is an empty widget that is easily adaptable to a variety of purposes. It does no drawing and defines no behavior except for invoking callback routines, which notify the application when graphics need to be drawn (exposure events or widget resize) and when the widget receives input from the keyboard or mouse. Client applications are responsible for defining appearance and behavior as needed in response to the **DrawingArea** widget callback routines.

The **DrawingArea** widget is also a composite widget that supports minimal geometry management for multiple widget or gadget children.

The **DrawingArea** widget inherits behavior and resources from the **Core**, **Composite**, **Constraint**, and **XmManager** widget classes. The class pointer is **xmDrawingAreaWidgetClass**. The class name is **XmDrawingArea**.

Subroutine

- **XmCreateDrawingArea**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letter (in either lower case or upper case, 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**). The following resource set lists resources of the **XmDrawingArea** widget class:

- **XmDrawingArea Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmDrawingArea** widget class:

- **XmManager Resource Set**
- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
int reason;
XEvent * event;
Window window;
}XmDrawingAreaCallbackStruct;
```

reason	Indicates why the callback was invoked.
event	Points to the **XEvent** that invoked the callback.
window	Is set to the widget window.

Behavior

The **XmDrawingArea** behavior is summarized below.

<KeyDown>, <KeyUp>, <BtnDown>, <BtnUp>: The callback routines for **XmNinputCallback** are invoked when a keyboard key or mouse button is pressed or released.

<Expose>: The callback routines for **XmNexposeCallback** are invoked when the widget receives an exposure event.

<Widget Resize>: The callback routines for **XmNresizeCallback** are invoked when the widget is resized.

Default Translations

The following are **XmDrawingArea** default translations:

<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()**
<EnterWindow>:	**Enter()**
<FocusIn>:	**FocusIn()**

Keyboard Traversal

The description of the **Manager** widget contains information on keyboard traversal.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/DrawingA.h

Related Information

The **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmCreateDrawingArea** subroutine, **XmManager** widget class.

XmDrawnButton Widget Class

Purpose

The **DrawnButton** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/DrawnB.h>

Children

No children are supported.

Description

A **DrawnButton** widget consists of an empty widget window surrounded by a shadow border. It provides a graphics area that can have **XmPushButton** input semantics.

Callback routines are defined for widget exposure and resize to allow the client application to redraw or reposition its graphics. If the **DrawnButton** widget has a highlight and shadow thickness, the application should not draw in that area. To avoid drawing in the highlight and shadow area, create the graphics context with a clipping rectangle for drawing in the widget. The clipping rectangle takes into account the size of the widget's highlight thickness and shadow.

The **DrawnButton** widget inherits behavior and resources from the **Core**, **XmPrimitive**, and **XmLabel** widget classes. The class pointer is **xmDrawnButtonWidgetClass**. The class name is **XmDrawnButton**.

Subroutines

- **XmCreateDrawnButton**
- **XtCreateWidget**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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)**. The following resource set lists the resources of the **XmDrawnButton** widget class:

- **XmDrawnButton Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmDrawnButton** widget class.

- **XmLabel Resource Set**
- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int       reason;
        XEvent    * event;
        Window    window;
} XmDrawnButtonCallbackStruct;
```

reason — Indicates why the callback was invoked.

event — Points to the **XEvent** that triggered the callback. **NULL** is returned by the *event* for the **XmNresizeCallback** resource. This event will be **NULL** for the **XmNactivateCallback** resource if the callback was triggered when the **XmPrimitive** resource **XmNtraversalOn** was **True** or if the callback was accessed through the **ArmAndActivate** action routine.

window — Is set to the window ID in which the event occurred.

Behavior

<Btn1Down>: A selection on the **DrawnButton** widget causes its shadow to be drawn in the selected state if the **XmNpushButtonEnabled** resource is set to **True**. The callbacks for the **XmNarmCallback** resource are also called.

<Btn1Up>: If **<Btn1Up>** occurs when the pointer is within the **DrawnButton**, the shadows are redrawn in the unselected state if the **XmNpushButtonEnabled** resource is set to **True**. The callbacks for the **XmNactivateCallback** resource are called, followed by callbacks for the **XmNdisarmCallback** resource.

If **<Btn1Up>** occurs when the pointer is outside the **DrawnButton**, the callbacks for the **XmNdisarmCallback** resource are called.

<Leave Window>: If the mouse button is pressed and the cursor leaves the **DrawnButton** window, the shadow is redrawn to its unselected state if the **XmNpushButtonEnabled** resource is set to **True**.

<Enter Window>: If the mouse button is pressed and the cursor reenters the **DrawnButton** window, the shadow is drawn in the same manner as when the button was first selected.

Default Translations

<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()** **Disarm()**
<Key>Return:	**ArmAndActivate()**
<Key>space:	**ArmAndActivate()**
<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**

Keyboard Traversal

Button assignments are: Left Button is Button 1; Left Button AND Right Button are Button 2; and Right Button is Button 3.

For information on keyboard traversal, refer to the **Primitive** widget and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/DrawnB.h

Related Information

The **Core** widget class, **XmCreateDrawnButton** subroutine, **XmLabel** widget class, **XmPrimitive** widget class, **XmPushButton** widget class, **XmSeparator** widget class, **XtCreateWidget** subroutine.

XmFileSelectionBox Widget Class

Purpose

The **FileSelectionBox** widget class.

Libraries

AIXwindows Library (**libXm.a**)

AIXwindows Library (**libIM.a**)

Syntax

#include <Xm/FileSB.h>

Children

ArrowButton Widget
CascadeButton Widget
DrawingArea Widget
Frame Widget
MenuShell Widget
PanedWindow Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
Text Widget
BulletinBoard Widget
Command Widget
Form Widget
Label Widget
MessageBox Widget
PushButton Widget
Scale Widget
ScrolledWindow Widget
Separator Widget
ToggleButton Widget

Description

A **FileSelectionBox** widget traverses through directories, views the files in them, and then selects a file.

Note: You should be aware of the proper usage of the **XmText** widget class before using this widget class.

A **FileSelectionBox** widget has four main areas:

- A directory mask that includes a **filter label** and a **directory mask** input field used to specify the directory that is to be examined.
- A scrollable list of file names.
- A text input field for directly typing in a file name.
- A group of **PushButton** widgets, labeled **Filter**, **OK**, **Cancel**, and **Help**.

One additional **WorkArea** child may be added to the **FileSelectionBox** widget after creation.

The user can select a file by scrolling through the list of file names and selecting the desired file or by entering the file name directly into the text edit area. Selecting a file from the list causes that file name to appear in the file selection text edit area.

The user may select a new file as many times as desired. The client application is not notified until the user selects the **OK PushButton** widget or presses the return key while the selection text edit area has the keyboard focus.

The **FileSelectionBox** widget initiates a file search when any of the following occurs:

- The **XtSetValues** subroutine is used to change the directory mask.
- The user activates the **Filter PushButton** widget.
- The client application calls the **XmFileSelectionDoSearch** subroutine.
- The user presses the Return key while the directory mask input field has the keyboard focus.

This can be useful when an application creates a new file and wants to incorporate it into the file list.

The **XmFileSelectionBox** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, **XmManager**, **XmBulletinBoard**, and **XmSelectionBox** classes. The class pointer is **xmFileSelectionBoxWidgetClass**. The class name is **XmFileSelectionBox**.

Subroutines

- **XmCreateFileSelectionBox**
- **XmFileSelectionBoxGetChild**
- **XmFileSelectionDoSearch**
- **XmCreateFileSelectionDialog**
- **XtCreateWidget**

New Resources

Setting the resource values also sets the resource values for this widget. To reference a resource 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 lower case or upper case, 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**). The following resource set lists the resources of the **XmFileSelectionBox** widget class:

- **XmFileSelectionBox Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmFileSelectionBox** widget class:

- **XmSelectionBox Resource Set**
- **XmBulletinBoard Resource Set**
- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
```

```
        int         reason;
        XEvent      event;
        XmString    value;
        int         length;
        XmString    mask;
        int         mask_length;
}XmFileSelectionBoxCallbackStruct;
```

reason	Indicates why the callback was invoked.
event	Points to the **XEvent** that triggered the callback.
value	Specifies the value of the current **XmNdirSpec**.
length	Specifies the number of bytes of the structure pointed to by *value*.
mask	Specifies the current value of **XmNdirMask**.
mask_length	Specifies the number of bytes of the structure pointed to by *mask*.

Behavior

The **XmFileSelectionBox** widget class inherits behavior from the **XmSelectionBox** widget class and the **XmBulletinBoard** widget class; below is an addition to that behavior:

<Apply Button Activated>:
A new file search begins when the apply button is activated.

Default Translations

The **XmFileSelectionBox** widget class inherits the **XmSelectionBox** widget class default translations.

Default Accelerators

The following are the default accelerator translations added to descendants of a **BulletinBoard** widget if the parent of the **BulletinBoard** widget is a **DialogShell** widget:

```
#override
<Key>F1:          Help()
<Key>Return:      Return()
<Key>KP_Enter:    Return()
```

Default Text Accelerators

The following are the default text accelerators inherited from the **XmSelectionBox** widget class:

```
#override
<Key>Up:          UpOrDown(0)
<Key>Down:        UpOrDown(1)
<Key>F1:          Help()
<Key>Return:      Return()
<Key>KP_Enter:    Return()
```

Keyboard Traversal

The **XmManager** widget class and its sections on behavior and default translations contain information on keyboard traversal.

File

/usr/include/Xm/FileSB.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmBulletinBoard** widget class, **XmCreateFileSelectionBox** subroutine, **XmCreateFileSelectionDialog** subroutine, **XmFileSelectionBoxGetChild** subroutine, **XmFileSelectionDoSearch** subroutine, **XmManager** widget class, **XmSelectionBox** widget class, **XtCreateWidget** subroutine.

XmForm Widget Class

Purpose

The **Form** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Form.h>

Children

ArrowButton Widget
CascadeButton Widget
DrawingArea Widget
Label Widget
MessageBox Widget
PushButton Widget
Scale Widget
ScrolledWindow Widget
Separator Widget
ToggleButton Widget
BulletinBoard Widget
Command Widget
Frame Widget
MenuShell Widget
PanedWindow Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
Text Widget

Description

A **Form** widget is a container widget with no input semantics of its own. Constraints are placed on children of the **Form** widget to define attachments for each of the four sides of each child widget. These attachments can be to the **Form** widget, to another child widget or gadget, to a relative position within the **Form** widget, or to the initial position of the child. The attachments determine the layout behavior of the **Form** widget when resizing occurs.

The **Form** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, **XmManager**, and **XmBulletinBoard** classes. The class pointer is **xmFormWidgetClass**. The class name is **XmForm**.

Subroutines

- **XmCreateForm**
- **XmCreateFormDialog**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource sets list the resources of the **XmForm** widget class:

- **XmForm Resource Set**

- **XmForm Constraint Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmForm** widget class:

- **XmBulletinBoard Resource Set**
- **XmManager Resource Set**
- **Core Resource Set**
- **Composite Resource Set**

Behavior

The **XmForm** widget class inherits the **XmBulletinBoard** widget class behavior.

Default Translations

The **XmForm** widget class inherits the **XmBulletinBoard** widget class default translations.

Default Accelerators

The default accelerator translations added to descendants of a **BulletinBoard** widget if the parent of the **BulletinBoard** widget is a **DialogShell** widget are:

```
#override
<Key>F1:          Help()
<Key>Return:      Return()
<Key>KP_Enter:    Return()
```

Keyboard Traversal

For information on keyboard traversal, refer to the **XmManager** widget class and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Form.h

Related Information

The **XmBulletinBoard** widget class, **Core** widget class, **Composite** widget class, **Constraint** widget class, **XmCreateForm** subroutine, **XmCreateFormDialog** subroutine, **XmManager** widget class.

XmFrame Widget Class

Purpose

The **Frame** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Frame.h>

Children

ArrowButton Widget
CascadeButton Widget
DrawingArea Widget
Label Widget
MessageBox Widget
PushButton Widget
Scale Widget
ScrolledWindow Widget
Separator Widget
ToggleButton Widget
BulletinBoard Widget
Command Widget
Form Widget
MenuShell Widget
PanedWindow Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
Text Widget

Description

A **Frame** widget is a very simple manager used to enclose a single child widget in a border drawn by the **Frame** widget. It uses the **XmManager** widget class resources for border drawing and performs geometry management such that its size always matches the size of the child widget plus the margins defined for it.

The **Frame** widget is most often used to enclose other managers when the manager is supposed to have the same border appearance as the primitive widgets. The **Frame** widget can also be used to enclose primitive widgets that do not support the same type of border drawing. This provides visual consistency when applications are developed using diverse widget sets.

If the **Frame** widget's parent is a **Shell** widget, the **XmNshadowType** resource is set by default to **XmSHADOW_OUT**, and the **Manager** widget's **XmNshadowThickness** resource is set to 1.

The **XmFrame** widget class inherits behavior and resources from the **Core** and **XmManager** classes. The class pointer is **xmFrameWidgetClass**. The class name is **XmFrame**.

Subroutine

- **XmCreateFrame**

New Resources

Setting the resource values for the inherited classes also sets resources for the **XmFrame** widget class. To reference a resource in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmFrame** widget class:

- **XmFrame Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmFrame** widget class:

- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Default Translations

<EnterWindow>:	**Enter()**
<FocusIn>:	**FocusIn()**
<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Frame.h

Related Information

The **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmCreateFrame** subroutine, **XmManager** widget class.

XmGadget Gadget Class

Purpose

The **Gadget** gadget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Children

ArrowButtonGadget Gadget
CascadeButtonGadget Gadget
LabelGadget Gadget
PushButtonGadget Gadget
SeparatorGadget Gadget
ToggleButtonGadget Gadget

Description

The **XmGadget** gadget class is used as a supporting superclass for other gadget classes. This gadget handles shadow border drawing and highlighting, traversal activation and deactivation, and various lists of callback routines needed by gadgets.

The color and pixmap resources defined by the **XmManager** widget class are directly used by gadgets. If the **XtSetValues** subroutine is used to change one of the resources for a **Manager** widget, all of the gadget children within the manager also change.

The **Gadget** gadget is built from the **RectObj** widget class. The class pointer is **xmGadgetClass**. The class name is **XmGadget**.

New Resources

Setting the resource values for the inherited classes also sets resources 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 lower case of upper case, but include any underscores between words). The codes in the access column indicate if the given resource is set at creation time (**C**), set by using **XtSetValues** (S), retrieved by using **XtGetValues** (G), or is not applicable (**N/A**). The following resource set lists the resources of the **XmGadget** widget class:

- **XmGadget Resource Set**

Inherited Resources

The following resource sets contain a complete description of resources inherited by the **XmGadget** widget class:

- **RectObj Resource Set**
- **Object Resource Set**

Behavior

Gadget widgets cannot have translations associated with them. Because of this, gadget behavior is determined by the **Manager** widget into which the **Gadget** gadget is placed. The

following types of events are caught by a **Manager** widget and forwarded to a **Gadget** gadget:

- **ButtonPress**
- **ButtonRelease**
- **EnterNotify**
- **LeaveNotify**
- **FocusIn**
- **FocusOut**
- **MotionNotify**

The **XmManager** widget class defines the translations supported by all **Manager** widgets.

File

/usr/include/Xm/Xm.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **Object** widget class, **RectObj** widget class, **XmManager** widget class.

XmLabel Widget Class

Purpose

The **Label** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Label.h>

Children

ArrowButton Gadget
CascadeButtonGadget Gadget
Label Widget
PushButtonGadget Gadget
ToggleButtonGadget Gadget
CascadeButton Widget
DrawnButton Widget
PushButton Widget
ToggleButton Widget

Description

The **XmLabel** widget is never instantiated; it is used as a superclass for other button widgets, such as **XmPushButton** and **XmToggleButton**. The **Label** widget does not accept any button or key input, and the **help** callback routine is the only callback routine defined. **XmLabel** widgets also receive enter and leave events.

A **Label** widget can contain either text or a pixmap. The **Label** text should be a compound string.The text can be multidirectional, multiline, and/or multifont. When a **Label** widget is insensitive, its text is stippled, or the user supplied insensitive pixmap is displayed.

The **XmLabel** widget class supports both accelerators and mnemonics primarily for use in the **Label** subclass widgets that are contained in menus. Mnemonics are available in a menu system when the button is visible. Accelerators in a menu system are accessible even when the button is not visible. The **Label** widget displays the mnemonic by underlining the first matching character in the text string. The accelerator is displayed as a text string to the right of the label text or pixmap.

The **Label** widget consists of many margin fields surrounding the text or pixmap. These margin fields are resources that can be set by the user, but the **Label** widget subclasses also modify some of these fields. The subclasses tend to modify the **XmNmarginLeft**, **XmNmarginRight**, **XmNmarginTop**, and **XmNmarginBottom** resources and leave the **XmNmarginWidth** and **XmNmarginHeight** resources as set by the application.

The **XmLabel** widget class inherits behavior and resources from **Core** and **XmPrimitive** widget classes. The class pointer is **xmLabelWidgetClass**. The class name is **XmLabel**.

Subroutines

- **XmCreateLabel**
- **XmFontListCreate**
- **XmStringCreate**
- **XmStringCreateLtoR**

- **XtCreateWidget**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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)**. The following resource set lists the resources of the **Label** class:

- **XmLabel Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **Label** widget:

- **XmPrimitive Resource Set**
- **XmCore Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int     reason;
        XEvent  * event;
} XmAnyCallbackStruct
```

reason Indicates why the callback was invoked. For this callback, *reason* is set to **XmCR_HELP**.

event Points to the **XEvent** that triggered the callback.

Behavior Default Translations

<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**
<Unmap>:	**Unmap()**
FocusOut>:	**FocusOut()**
<FocusIn>:	**FocusIn()**
<Key>space:	**Noop()**
<Key>Left:	**MenuTraverseLeft()**
<Key>Right:	**MenuTraverseRight()**
<Key>Up:	**MenuTraverseUp()**
<Key>Down:	**MenuTraverseDown()**
<Key>Home:	**Noop()**

Keyboard Traversal

For information on keyboard traversal, refer to the **XmPrimitive** widget class and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Label.h

Related Information

The **Core** widget class, **XmPrimitive** widget, **XmCreateLabel** subroutine, **XmFontListCreate** subroutine, **XmStringCreate** subroutine, **XmStringCreateLtoR** subroutine, **XtCreateWidget** subroutine.

XmLabelGadget Gadget Class

Purpose

The **LabelGadget** gadget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/LabelG.h>

Children

CascadeButtonGadget Gadget
PushButtonGadget Gadget
ToggleButtonGadget Gadget

Description

The **XmLabelGadget** gadget class is never instantiated; it is used as a superclass for other button gadgets, such as a **PushButtonGadget** gadget and a **ToggleButtonGadget** gadget. The **LabelGadget** gadget does not accept any button or key input, and the **help** callback routine is the only callback routine defined. The **LabelGadget** gadget also receives enter and leave events.

A **LabelGadget** gadget can contain either text or a pixmap. **LabelGadget** text is a compound string. Refer to the **XmString** widget class for more information on compound strings. The text can be multidirectional, multiline, and/or multifont. When a **LabelGadget** gadget is insensitive, its text is stippled, or the user supplied insensitive pixmap is displayed.

A **LabelGadget** gadget supports both accelerators and mnemonics primarily for use in the **LabelGadget** subclass gadgets that are contained in menus. Mnemonics are available in a menu system when the button is visible. Accelerators in a menu system are accessible even when the button is not visible. The **XmLabelGadget** gadget displays the mnemonic by underlining the first matching character in the text string. The accelerator is displayed as a text string to the right of the label text or pixmap.

A **LabelGadget** gadget consists of many margin fields surrounding the text or pixmap. These margin fields are resources that can be set by the user, but the **LabelGadget** subclasses also modify some of these fields. The subclasses tend to modify the margin **XmNmarginLeft**, **XmNmarginRight**, **XmNmarginTop** and **XmNmarginBottom** resources and leave the **XmNmarginWidth** and **XmNmarginHeight** resources as set by the client application.

The **LabelGadget** gadget class inherits behavior and resources from the **Object**, **RectObj** and **XmGadget** classes. The class pointer is **xmLabelGadgetClass**. The class name is **XmLabelGadget**.

Subroutines

- **XmCreateLabelGadget**
- **XmFontListCreate**
- **XmStringCreate**
- **XmStringCreateLtoR**

- **XtCreateWidget**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmLabelGadget** widget class:

- **XmLabelGadget Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmLabelGadget** widget class:

- **XmGadget Resource Set**
- **Object Resource Set**
- **RectObj Resource Set**

Keyboard Traversal

For information on keyboard traversal, refer to the **XmGadget** widget class and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/LabelG.h

Related Information

The **Object** widget class, **RectObj** widget class, **XmCreateLabelGadget** subroutine, **XmFontListCreate** subroutine, **XmGadget** widget class, **XmStringCreate** subroutine, **XmStringCreateLtoR** subroutine, **XtCreateWidget** subroutine.

XmList Widget Class

Purpose

The **List** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

Children

No children are supported.

Description

A **List** widget allows a user to select one or more items from a group of choices. Items are selected from the list in a variety of ways, using both the pointer and the keyboard.

The **List** widget operates on an array of strings that are defined by the client application. Each string becomes an item in the **List** widget, with the first string becoming the item in position one, the second string becoming the item in position two, and so on.

The visual size of the **List** widget is set by specifying the number of items that are visible. If the ability to scroll through a large set of choices is desired, use the **XmCreateScrolledList** convenience subroutine.

To select items, move the pointer or cursor to the desired item and press the mouse button or the key defined as select. There are several styles of selection behavior, and they all highlight the selected item or items by displaying them in inverse colors. An appropriate callback routine is invoked to notify the application of the user's choice. The application then takes whatever action is required for the specified selection.

The **XmList** widget class inherits behavior and resources from **Core** and **XmPrimitive** classes. The class pointer is **xmListWidgetClass**. The class name is **XmList**.

Subroutines

- **XmCreateList**
- **XmCreateScrolledList**
- **XmListAddItem**
- **XmListAddItemUnselected**
- **XmListDeleteItem**
- **XmListDeletePos**
- **XmListDeselectAllItems**
- **XmListDeselectItem**
- **XmListItemExists**
- **XmListSelectItem**

- **XmListSelectPos**
- **XmListSetBottomItem**
- **XmListSetBottomPos**
- **XmListSetHorizPos**
- **XmListSetItem**
- **XmListSetPos**
- **XtCreateWidget**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource sets list the resources of the **XmList** widget class:

- **XmList Resource Set**
- **XmScrolledList Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmList** widget class:

- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The **XmList** widget class defines a new callback structure. The client application must first look at the reason field and only use the structure members that are valid for that particular reason, because not all of the fields are relevant for every possible reason. The callback structure is defined as follows:

```
typedef struct
{
        int             reason;
        XEvent          * event;
        XmString        item;
        int             item_length;
        int             item_position;
        XmString        * selected_items;
        int             selected_item_count;
        int             selection_type;
} XmListCallbackStruct;
```

reason Indicates why the callback was invoked.

event	Points to the **XEvent** that invoked the callback. It can be **NULL**.
item	Is the item selected by this action. *selected_items* points to a temporary storage space that is reused after the callback is finished. Therefore, if an application needs to save the selected list, it should copy the list into its own data space.
item_length	Is the length of the item when the selection action occurred.
item_position	Is the position in the **List** widget of the selected item.
selected_items	Points to the list of items selected at the time of the *event* that caused the callback. *selected_items* points to a temporary storage space that is reused after the callback is finished. Therefore, if an application needs to save the selected list, it should copy the list into its own data space.
selected_items_count	Is the number of items in the *selected_items* list.
selection_type	Indicates that the most recent extended selection was either the initial selection (**XmINITIAL**), a modification of an existing selection (**XmMODIFICATION**), or an additional non–contiguous selection (**XmADDITION**).

The following table describes the reasons for which the individual callback structure fields are valid:

Reason Valid Fields

XmCR_SINGLE_SELECT	*reason, event, Item, item_length, item_position*
XmCR_DEFAULT_ACTION	*reason, event, Item, item_length, item_position*
XmCR_BROWSE_SELECT	*reason, event, Item, item_length, item_position*
XmCR_MULTIPLE_SELECT	*reason, event, Item,selected_items, selected_item_count*
XmCR_EXTENDED_SELECT	*reason, event, Item, selected_items, selected_item_count, selection_type*

Behavior

The **XmList** widget class provides several methods for selecting its items. The general selection model is as follows:

The user moves the pointer to the item that is desired to be selected, either by using the mouse to move the pointer over the desired item, or, in keyboard traversal mode, moving the active highlight to the desired item with the up and down arrow keys. The item is selected by clicking the select button on the mouse (usually the left mouse button), or by pressing the select key on the keyboard (usually defined to be the Space key). Each of the selection modes provides some variation of the above behavior.

Note that the keyboard selection interface is only active when traversal is enabled for the **List** widget.

The selection mode is set by the **XmNselectionPolicy** resource and is modified by the **XmNautomaticSelection** resource. The behavior of the various modes are defined below:

XmSINGLE_SELECT (Single Selection): Move the mouse pointer or keyboard highlight until it is over the desired item and press the select button or key. The item inverts its foreground and background colors to indicate that it is to be the selected object. Any

previously selected items are unselected (returned to their normal visual state). When the button or key is released, the **XmNsingleSelectionCallback** resource is invoked.

XmBROWSE_SELECT (Browse Selection): When using the mouse, press the select button; the item under the pointer is highlighted. While the button is held down, drag the selection by moving the pointer. When the select button is released, the object under the pointer becomes the selected item and the **XmNbrowseSelectionCallback** resource is invoked.

If the **XmNautomaticSelection** resource is **True**, the **XmNbrowseSelectionCallback** resource invokes when the select button is pressed. For each subsequent item entered while the select button is held down, the callback is invoked when the pointer moves into the item. No selection callback is invoked when the button is released.

When selecting through the keyboard and the **XmNautomaticSelection** resource is **False**, browse selection is no different from single–selection mode. However, when the **XmNautomaticSelection** resource is **True**, the callback is invoked for each element that is selected. Both the keyboard highlight and the selection highlight move as the user moves through the list.

XmMULTIPLE_SELECT (Multiple Selection): Move the mouse pointer or keyboard highlight until it is over the desired item and press the select button or key. The item inverts its foreground and background colors to indicate that it is a selected object. Any previously selected items are not affected by this action. When the button or key is released, the **XmNmultipleSelectionCallback** resource is invoked. To unselect an item in this mode, move to a selected item and press the select button or key. The **XmNmultipleSelectionCallback** resource invokes with the updated selection list.

XmEXTENDED_SELECT (Extended Selection): This mode selects a contiguous range of objects with one action. Press the select button on the first item of the range. This begins a new selection process, which deselects any previous selection in the list. That item inverts to show its inclusion in the selection. While depressing the button, drag the cursor through other items in the **List**. As the pointer moves through the list, all items between the initial item and the item currently under the pointer are inverted to show that they are included in the selection. When the button is released, the **XmNextendedSelectionCallback** resource is invoked and contains a list of all selected items. The *selection_type* field is set to the **XmINITIAL** value.

Modify a selection by pressing and holding the shift key, moving to the new endpoint, and pressing the select button. The items between the initial start point and the new end point are selected. The rest of the selection process proceeds as above. Any previous selections are not unselected. When the select button is released, the **XmNextendedSelectionCallback** resource is invoked and contains a list of all selected items, both new and previous. The *selection_type* field is set to the **XmMODIFICATION** value.

Items can be added to or deleted from a selected range by using the CTRL key. To add an additional range to an existing selection, move to the first item of the new group, press and hold the CTRL key, and then press the Select button. The item under the pointer inverts; any previous selections are unaffected. This item becomes the initial item for the new selection range. If the pointer is dragged through additional items while the CTRL key and select button are held down, those items invert as described above. When the Select button is released, the **XmNextendedSelectionCallback** resource is invoked and contains a list of all selected items, both new and previous. The *selection_type* field is set to the **XmADDITION** value.

To delete an item or a range of items from an existing selection, move to the first item to be deselected, press and hold the CTRL key, and then press the select button. The item under the pointer returns to its normal visual state to indicate that it is no longer in the selection. This item becomes the initial item for the range to be deselected. If the pointer is dragged through additional selected items while the CTRL key and select button are held down, those items are deselected. Any other selection are unaffected. When the select button is released, the **XmNextendedSelectionCallback** resource is invoked and contains a list of all remaining selected items, both new and previous. The *selection_type* field is set to the **XmADDITION** value.

A range of items can also be deselected by setting the initial item for the range as described above, then moving to the end of the range, and pressing the select button while holding the Shift key down. All items between the two endpoints are deselected. When the button is released, the **XmNextendedSelectionCallback** resource is issued as described above.

If the **XmNautomaticSelection** resource is set to **True**, the **XmNextendedSelectionCallback** resource is invoked when the select button is pressed. For each subsequent item the user selects or deselects, the callback is invoked when the pointer is moved into the item. The *selection_type* field is set to reflect the current action. No selection callback is invoke when the button is released.

Keyboard selection in extended selection mode is accomplished by moving the keyboard highlight to the start of the desired range and pressing the select key. The selection callback is invoked with a *selection_type* value of the **XmINITIAL** value. Then, using the arrow keys, move the keyboard highlight to the end of the range, depress the Shift key, and press the Select key. This invokes the **XmNextendedSelectionCallback** resource with a value of the **XmMODIFICATION** value. Select additional ranges by moving to the beginning of a range, pressing the select key while depressing the CTRL key, and then moving to the end of the range and pressing the select key while holding the Shift key. Erase previously selected elements by moving to them and pressing the select key while holding down the CTRL key. In all cases, callbacks are issued as described above.

When using the keyboard with the **XmNautomaticSelection** resource set to **True**, the **XmNextendedSelectionCallback** resource is invoked when the select button is pressed. For each subsequent item the user selects, the callback is invoked when the pointer is moved into the item if there are modifier keys in use. For example, start the selection by pressing the select key, and then extend it by using the arrow keys while holding down the Shift key. The *selection_type* field is set to reflect the current action. There is no selection callback invoked when the button is released.

XmDEFAULT_ACTION (Double Click): If an object is clicked twice within the interval defined by the **XmNdoubleClickInterval** resource, the **List** widget interprets that as a double click and invokes the **XmNdefaultActionCallback** resource. The item inverts to indicate its selection.

Default Translations

The following are the default translations for the **List** widget:

Button1<Motion>:	**ListButtonMotion()**
Shift Ctrl ~Meta<Btn1Down>:	**ListShiftCtrlSelect()**
Shift Ctrl ~Meta<Btn1Up>:	**ListShiftCtrlUnSelect()**
Shift Ctrl ~Meta<KeyDown>space:	**ListKbdShiftCtrlSelect()**
Shift Ctrl ~Meta<KeyUp>space:	**ListKbdShiftCtrlUnSelect()**
Shift Ctrl ~Meta<KeyDown>Select:	**ListKbdShiftCtrlSelect()**
Shift Ctrl ~Meta<KeyUp>Select:	**ListKbdShiftCtrlUnSelect()**
Shift ~Ctrl ~Meta<Btn1Down>:	**ListShiftSelect()**

Shift ~Ctrl ~Meta<Btn1Up>:	**ListShiftUnSelect()**
Shift ~Ctrl ~Meta<KeyDown>space:	**ListKbdShiftSelect()**
Shift ~Ctrl ~Meta<KeyUp>space:	**ListKbdShiftUnSelect()**
Shift ~Ctrl ~Meta<KeyDown>Select:	**ListKbdShiftSelect()**
Shift ~Ctrl ~Meta<KeyUp>Select:	**ListKbdShiftUnSelect()**
Ctrl ~Shift ~Meta<Btn1Down>Down:	**ListCtrlSelect()**
Ctrl ~Shift ~Meta<Btn1Up>:	**ListCtrlUnselect()**
Ctrl ~Shift ~Meta<KeyDown>space:	**ListKbdCtrlSelect()**
Ctrl ~Shift ~Meta<KeyUp>space:	**ListKdbCtrlUnselect()**
Ctrl ~Shift ~Meta<KeyDown>Select:	**ListKbdCtrlSelect()**
Ctrl ~Shift ~Meta<KeyUp>Select:	**ListKbdCtrlUnselect()**
~Shift ~Ctrl ~Meta<Btn1Down>;	**ListElementSelect()**
~Shift ~Ctrl ~Meta<Btn1Up>:	**ListElementUnSelect()**
~Shift ~Ctrl ~Meta<KeyDown>space:	**ListKbdSelect()**
~Shift ~Ctrl ~Meta<KeyUp>space:	**ListKbdUnSelect()**
~Shift ~Ctrl ~Meta<KeyDown>Select:	**ListKbdSelect()**
~Shift ~Ctrl ~Meta<KeyUp>Select:	**ListKbdUnSelect()**
Shift Ctrl ~Meta<Key>Up:	**ListShiftCtrlPrevElement()**
Shift Ctrl ~Meta<Key>Down:	**ListShiftCtrlNextElement()**
Shift ~Ctrl ~Meta<Key>Up:	**ListShiftPrevElement()**
Shift ~Ctrl ~Meta<Key>Down:	**ListShiftNextElement()**
~Shift Ctrl ~Meta<Key>Up:	**ListCtrlPrevElement()**
~Shift Ctrl ~Meta<Key>Down:	**ListCtrlNextElement()**
~Shift ~Ctrl ~Meta<Key>Up:	**ListPrevElement()**
~Shift ~Ctrl ~Meta<Key>Down:	**ListNextElement()**
<Enter>:	**ListEnter()**
<Leave>:	**ListLeave()**
<FocusIn>:	**ListFocusIn()**
<FocusOut>:	**ListFocusOut()**
<Unmap>:	**PrimitiveUnmap()**
Shift<Key>Tab:	**PrimitivePrevTabGroup()**
<Key>Tab:	**PrimitiveNextTabGroup()**
<Key>Home:	**PrimitiveTraverseHome()**

Keyboard Traversal

For those actions not inherited from the **XmPrimitive** widget class, keyboard traversal is described in the behavior section shown previously in this widget.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/List.h

Related Information

The **Core** widget class, **XmPrimitive** widget class, **XmCreateList** subroutine, **XmCreateScrolledList** subroutine, **XmFontListCreate** subroutine, **XmListAddItem** subroutine, **XmListAddItemUnselected** subroutine, **XmListDeleteItem** subroutine, **XmListDeletePos** subroutine, **XmListDeselectItem** subroutine, **XmListDeselectAllItems** subroutine, **XmListSelectItem** subroutine, **XmListSetHorizPos** subroutine, **XmListSetItem** subroutine, **XmListSetPos** subroutine, **XmListSetBottomItem** subroutine, **XmListSetBottomPos** subroutine, **XmListSelectPos** subroutine, **XmListDeselectPos** subroutine, **XmListItemExists** subroutine, **XmStringCreate** subroutine, **XtCreateWidget** subroutine.

XmMainWindow Widget Class

Purpose

The **MainWindow** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MainW.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DrawingArea Widget
Form Widget
Label Widget
MenuShell Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawnButton Widget
Frame Widget
LabelGadget Gadget
MessageBox Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget

Description

The **XmMainWindow** widget class provides a standard layout for the primary window of a client application. This layout includes a **MenuBar** widget, a **CommandWindow** widget, a work region, and **ScrollBar** widgets. Any or all of these areas are optional. The work region and **ScrollBar** widgets in the **MainWindow** widget behave identically to the work region and **ScrollBar** widgets in the **ScrolledWindow** widget. The user can think of the **MainWindow** widget as an extended **ScrolledWindow** widget with an optional **MenuBar** widget and optional a **CommandWindow** widget.

In a fully–loaded **MainWindow** widget, the **MenuBar** widget spans the top of the window horizontally. The **CommandWindow** widget spans the **MainWindow** widget horizontally just below the **MenuBar** widget, and the work region lies below the **CommandWindow** widget. The space remaining below the **CommandWindow** widget, if any, is managed in a manner identical to the **ScrolledWindow** widget. The behavior of a **ScrolledWindow** widget can be controlled by the **ScrolledWindow** resources. To create a **MainWindow** widget, create a **MenuBar** widget, a **CommandWindow** widget, a horizontal **ScrollBar** widget, and a vertical **ScrollBar** widget to use as the work region, and then call the **MainWindowSetArea** subroutines with those widget IDs.

The **MainWindow** widget can also create two **XmSeparator** widgets that provide a visual separation of the **MainWindow** widget's three components.

The **MainWindow** widget inherits behavior and resources from **Core**, **Composite**, **Constraint**, **XmManager**, and **XmScrolledWindow** classes. The class pointer is **xmMainWindowWidgetClass.** The class name is **XmMainWindow.**

Subroutines

- **XmCreateMainWindow** subroutine
- **XmMainWindowSep1** subroutine
- **XmMainWindowSep2** subroutine
- **XmMainWindowSetAreas** subroutine

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmMainWindow** widget class:

- **XmMainWindow Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmMainWindow** widget class:

- **XmScrolledWindow Resource Set**
- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Behavior

The **XmMainWindow** widget class inherits behavior from the **XmScrolledWindow** widget class.

Keyboard Traversal

For information on keyboard traversal, refer to the **XmManager** widget class and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MainW.h

Related Information

The **Constraint** widget class, **Composite** widget class, **Core** widget class, **XmCreateMainWindow** subroutine, **XmMainWindowSetAreas** subroutine, **XmMainWindowSep1** subroutine, **XmMainWindowSep2** subroutine, **XmManager** widget class, **XmScrolledWindow** widget class.

XmManager Widget Class

Purpose

The **Manager** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
MessageBox Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget

Description

The **XmManager** widget class is used as a supporting superclass for other widget classes. This widget class supports the visual resources, graphics contexts, and traversal resources necessary for the graphics and traversal mechanisms.

The **XmManager** widget class is built from the **Core**, **Composite**, and **Constraint** classes. The class pointer is **xmManagerWidgetClass**. The class name is **XmManager**.

New Resources

The **XmManager** widget defines a set of widget resources that are used to specify data. Setting the resource values for the inherited classes also sets resources 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 lower or upper case, 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**). The following superclass contains a complete description of resources defined by the **XmManager** widget class:

- **XmManagerResource Set**

Dynamic Color Defaults

The foreground, background, top shadow, and bottom shadow resources are dynamically defaulted. If no color data is specified, the colors are automatically generated. On a monochrome system, a black and white color scheme is generated. On a color system, a set of four colors are generated which display the correct shading for the three–dimensional visuals.

If the background is the only color specified for a widget, the top shadow, bottom shadow, and foreground colors are generated to give the three–dimensional appearance. The color generation works best with non–saturated colors. Using pure red, green, or blue yields poor results.

Colors are generated at creation only. Resetting the background through **XtSetValues** does not regenerate the other colors.

Inherited Resources

The following superclass resource sets contain a complete description of the resources inherited by the **XmManager** widget class:

- **Composite Resource Set**
- **Core Resource Set**

Behavior

The following set of translations are used by **Manager** widgets that have **Gadget** children. Since **Gadgets** cannot have translations associated with them, it is the responsibility of the **Manager** widget to intercept the events of interest and pass them to the appropriate **Gadget** child.

Shift <Key> Tab:

Moves the focus to the first item contained within the previous tab group. If the beginning of the tab group list is reached, it wraps to the end of the tab group list.

<Key> Tab or **<Key> F6**:

Moves the focus to the first item contained within the next tab group. If the current tab group is the last entry in the tab group list, it wraps to the beginning of the tab group list.

<Key> Up or **<Key> Left**:

Moves the keyboard focus to the previous **Manager** widget or gadget within the current tab group. The previous widget or gadget is the one that is the previous entry in the tab group's list of children. Wrapping occurs, if necessary.

<Key> Down or **<Key> Right**:

Moves the keyboard focus to the next **Manager** widget or gadget within the current tab group. The next widget or gadget is the one that is the next entry in the tab group's list of children. Wrapping occurs, if necessary.

<Key> Home:

Moves the keyboard focus to the first **Manager** widget or gadget in the current tab group.

Default Translations

The following are translations used by all **Manager** widgets:

<EnterWindow>:	**ManagerEnter()**
<FocusOut>:	**ManagerFocusOut()**
<FocusIn>:	**ManagerFocusIn()**

The following are the translations necessary to provide gadget event processing:

<Key> space:	**ManagerGadgetSelect()**
<Key> Return:	**ManagerGadgetSelect()**
Shift <Key> Tab:	**ManagerGadgetPrevTabGroup()**
<Key> Tab:	**ManagerGadgetNextTabGroup()**
<Key> F6:	**ManagerGadgetNextTabGroup()**
<Key> Up:	**ManagerGadgetTraversePrev()**
<Key> Down:	**ManagerGadgetTraverseNext()**
<Key> Left:	**ManagerGadgetTraversePrev()**
<Key> Right:	**ManagerGadgetTraverseNext()**
<Key> Home:	**ManagerGadgetTraverseHome()**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmGadget** gadget class.

XmMenuShell Widget Class

Purpose

The **MenuShell** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MenuShell.h>

Children

ArrowButton Widget
CascadeButton Widget
MenuBar
PopupMenu
PushButton Widget
RowColumn Widget
SeparatorGadget Gadget
ToggleButtonGadget Gadget
ArrowButtonGadget Gadget
CascadeButtonGadget Gadget
OptionMenu
PulldownMenu
PushButtonGadget Gadget
Separator Widget
ToggleButton Widget

Description

The **MenuShell** widget is a custom **OverrideShell** widget. An **OverrideShell** widget bypasses the window manager when displaying itself. It is designed specifically to contain **Popup** or **Pulldown MenuPanes**.

This widget is rarely encountered because the convenience subroutines **XmCreatePopupMenu** or **XmCreatePulldownMenu** are generally used to create a **Popup** or **Pulldown MenuPane**. The convenience subroutines automatically create a **MenuShell** widget as the parent of the **MenuPane**. However, if the convenience subroutines are not used, the required **MenuShell** widget must be created. In this case, it is important to note that the parent of the **MenuShell** widget depends on the type of menu system being built.

- If the **MenuShell** widget is for the top–level **Popup MenuPane**, the **MenuShell** widget must be created as a child of the widget from which the **MenuShell** widget is popped up.
- If the **MenuShell** widget is for a **MenuPane** that is pulled down from a **Popup** or another **Pulldown MenuPane**, the **MenuShell** widget must be created as a child of the **Popup** or **Pulldown MenuPane**'s parent **MenuShell**.
- If the **MenuShell** widget is for a **MenuPane** that is pulled down from a **MenuBar**, the **MenuShell** widget must be created as a child of the **MenuBar**.
- If the **MenuShell** widget is for a **Pulldown MenuPane** in an **OptionMenu**, the **MenuShell** widget must have the same parent as the **OptionMenu**.

The **XmMenuShell** widget class inherits resources from **Core**, **Composite**, and **OverrideShell** classes. The class pointer is **xmMenuShellWidgetClass**. The class name is **XmMenuShell**.

Subroutines

- **XmCreateMenuShell**
- **XmCreatePopupMenu**
- **XmCreatePulldownMenu**

New Resources

The **XmMenuShell** defines no new resources, but overrides the **XmNallowShellResize** resource in the **Shell** widget class.

Inherited Resources

Setting the resource values for the inherited classes also sets resources 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 lower or upper case, 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**). The following resource sets contain a complete description of resources inherited by the **XmMenuShell** widget class:

- **Shell Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Behavior

The specific mouse button that is used depends upon the resources **XmNrowColumnType** and **XmNwhichButton** in the menu's top–level **RowColumn** widget

Default PopupMenu System

<Btn2Down>: If this event has not already been processed by another menu component, keyboard traversal is disabled for the menus and the user is returned to drag mode.

<Btn2Up>: If this event has not already been processed by another menu component, all visible **MenuPanes** are unposted.

<Key> Escape: If this event has not already been processed by another menu component, all visible **MenuPanes** are unposted.

Default PulldownMenu System or OptionMenu System

<Btn1Down>: If this event has not already been processed by another menu component, keyboard traversal is disabled for the menus and the user is returned to drag mode.

<Btn1Up>: If this event has not already been processed by another menu component, all visible **MenuPanes** are unposted.

<Key> Escape: If this event has not already been processed by another menu component, all visible **MenuPanes** are unposted.

Default Translations

The default translations for the **MenuShell** widget are:

<BtnDown>:	**ClearTraversal()**
<Key> Escape:	**MenuShellPopdownDone()**
<BtnUp>:	**MenuShellPopdownDone()**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MenuShell.h

Related Information

The **Composite** widget class, **Core** widget class, **OverrideShell** widget class, **Shell** widget class, **XmCreateMenuShell** subroutine, **XmCreatePopupMenu** subroutine, **XmCreatePulldownMenu** subroutine, **XmRowColumn** widget class.

XmMessageBox Widget Class

Purpose

The **MessageBox** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
MessageBox Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget

Description

The **XmMessageBox** widget class is a dialog class used for creating simple message dialogs. Convenience dialogs based on the **MessageBox** widget are provided for several common interaction tasks including giving information, asking questions, and notifying about errors.

A **MessageBox** widget dialog is transient in nature; it is displayed for the duration of a single interaction. The **XmMessageBox** widget class is a subclass of the **XmBulletinBoard** widget class and depends on it for much of its general dialog behavior.

A **MessageBox** widget can contain a message symbol, a message, and up to three standard default **PushButtons**: **OK**, **Cancel**, and **Help**. It is laid out with the symbol in the top left, the message in the top and center–to–right side, and the **PushButtons** on the bottom. The Help button is positioned to the far right of the other **PushButtons**. Default symbols and button labels for the **XmMessageBox** widget convenience dialogs are localizable.

The Button label defaults are easily modified by including the new values in any of the app_defaults file locations supported by **Enhanced X–Windows** toolkit. Changing the defaults for the **MessageBox** widget symbols is more complicated, since the **Enhanced X–Windows** toolkit does not support specification of pixmaps by name in resource files.

At initialization, the **MessageBox** widget looks for a bitmaps subdirectory that includes the following bitmap files:

- **xm_error**
- **xm_information**
- **xm_question**
- **xm_working**
- **xm_warning**

A description of what paths are searched for these files is contained in **XmGetPixmap** subroutine.

The **XmMessageBox** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, **XmManager**, and **XmBulletinBoard** classes. The class pointer is **xmMessageBoxWidgetClass**. The class name is **XmMessageBox**.

Subroutines

- **XmCreateErrorDialog**
- **XmCreateInformationDialog**
- **XmCreateMessageBox**
- **XmCreateMessageDialog**
- **XmCreateQuestionDialog**
- **XmCreateWarningDialog**
- **XmCreateWorkingDialog**
- **XmMessageBoxGetChild**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmMessageBox** widget class:

- **XmMessageBox Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmMessageBox** widget class:

- **XmBulletinBoard Resource Set**
- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int     reason;
        XEvent  * event;
} XmAnyCallbackStruct;
```

reason Is set to the value that corresponds to the type of selection that invoked this callback.

event Points to the **XEvent** that invoked the callback.

Behavior

Following is a summary of the behavior of the **MessageBox** widget:

<OK Button Activated>: When the **OK PushButton** is activated, the callbacks for the **XmNokCallback** resource are called.

<Cancel Button Activated>: When the cancel **PushButton** is activated, the callbacks for the **XmNcancelCallback** resource are called.

<Help Button Activated> or **<Key>F1:** When the help button or **subroutine key** 1 is pressed, the callbacks for the **XmNhelpCallback** resource are called.

<Default Button Activated>: When the default button is pressed, the activate callbacks of the default **PushButton** are called.

<FocusIn>: When a **FocusIn** event is generated on the widget window, the callbacks for the **XmNfocusCallback** resource are called.

<MapWindow>: When a **MapWindow** event is generated on the widget window, the callbacks for the **XmNmapCallback** resource are called.

<UnmapWindow>: When a **UnmapWindow** event is generated on the widget window, the callbacks for the **XmNunmapCallback** resource are called.

Default Accelerators

The default accelerator translations added to descendants of a **BulletinBoard** widget if the parent of the **BulletinBoard** widget is a **DialogShell** widget are:

```
#override
<Key>F1:          Help()
<Key>Return:      Return()
<Key>KP_Enter:    Return()
```

Keyboard Traversal

Information on keyboard traversal is contained in the **Manager** widget and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MessageB.h

Related Information

The **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmBulletinBoard** widget class, **XmCreateErrorDialog** subroutine, **XmCreateInformationDialog** subroutine, **XmCreateMessageDialog** subroutine, **XmCreateQuestionDialog** subroutine, **XmCreateWarningDialog** subroutine, **XmCreateWorkingDialog** subroutine, **XmManager** widget class, **XmMessageBoxGetChild** subroutine.

XmPanedWindow Widget Class

Purpose

The **PanedWindow** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/PanedW.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DrawingArea Widget
Form Widget
Label Widget
MenuShell Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawnButton Widget
Frame Widget
LabelGadget Gadget
MessageBox Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget

Description

A **PanedWindow** widget is a composite widget that lays out child widgets in a vertically–tiled format. Child widgets appear in top–to–bottom fashion: the first child is inserted at the top of the **PanedWindow** widget and the last child is inserted at the bottom. The **PanedWindow** widget grows to match the width of its widest child and all other children are forced to this width. The height of the **PanedWindow** widget is equal to the sum of the heights of all its child widgets, the spacing between them, and the size of the top and bottom margins.

A mouse can be used to adjust the size of the individual panes. To facilitate this adjustment, a pane control sash is created for most children. The sash appears as a square box positioned on the bottom of the pane it controls.

The **PanedWindow** widget is a constraint widget, which means that it creates and manages a set of constraints for each child. A minimum and maximum size can be specified for each pane. The **PanedWindow** widget does not allow a pane to be resized below its minimum size nor beyond its maximum size. When the minimum size of a pane is equal to its maximum size, no control sash is presented for that pane or for the lowest pane.

Subroutine

- **XmCreatePanedWindow**

New Resources

Setting the resource values for the inherited classes also sets resources 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 lower case or upper case, 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**). The following resource sets list the resources of the **XmPanedWindow** widget class:

- **XmPanedWindow Resource Set**
- **XmPanedWindow Constraint Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmPanedWindow** widget class.

- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Behavior

Shift<Btn1Down>: (in sash): Activates the interactive placement of the pane's borders. Changes the pointer cursor from a crosshair to an upward pointing arrow to indicate that the upper pane is adjusted (usually the pane to which the sash is attached). All panes below the sash that can be adjusted, are adjusted.

<Btn1Down>: (in sash): Activates the interactive placement of the pane's borders. Changes the pointer cursor from a crosshair to a double–headed arrow to indicate that the pane to be adjusted is the pane to which the sash is attached. Also indicates that the first pane below the sash can be adjusted. Unlike pane adjustment using **ShiftBtn1Down** or **CTRLBtn1Down**, only two panes is affected. If one of the panes reaches its minimum or maximum size, adjustment stops, and the next adjustable pane is not identified.

CTRL <Btn1Down>: (in sash): Activates the interactive placement of the pane's borders. Changes the pointer cursor from a crosshair to a downward–pointing arrow to indicate that the lower pane will be adjusted (usually the pane below the pane to which the sash is attached). All panes above the sash that can be adjusted, are adjusted.

Shift Button1 <PtrMoved>: If the mouse button is pressed while the pointer is within the sash, the motion events draw a series of track lines to illustrate what the heights of the panes would be if the Commit action were invoked. This action determines which pane below the upper pane can be adjusted, and then makes the appropriate adjustments.

Button1 <PtrMoved>: If the mouse button is pressed while the pointer is within the sash, the motion events draw a series of track lines to illustrate what the heights of the panes would be if the Commit action were invoked. This action adjusts as needed (and as possible) the upper and lower panes selected when the **Btn1Down** action was invoked.

CTRL Button1 <PtrMoved>: If the mouse button is pressed while the pointer is within the sash, the motion events draw a series of track lines to illustrate what the heights of the panes would be if the Commit action were invoked. This action determines which pane above the lower pane can be adjusted, and then makes the appropriate adjustments.

Any <BtnUp>: Commits to any action taken since the interactive placement was activated. The sashes and the pane boundaries are moved to the committed positions of the panes.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/PanedW.h

Related Information

The **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmManager** widget class, **XmCreatePanedWindow** subroutine.

XmPrimitive Widget Class

Purpose

The **Primitive** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Children

No children are supported.

Description

The **XmPrimitive** widget class is used as a supporting superclass for other widget classes. This widget handles border drawing and highlighting, traversal activation and deactivation, and various callback lists needed by the **Primitive** widgets.

The **XmPrimitive** widget class inherits behavior and resources from the **Core** widget class. The class pointer is **xmPrimitiveWidgetClass**. The class name is **XmPrimitive**.

New Resources

Setting the resource values for the inherited classes also sets resources 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**). The following resource set lists the resources of the **XmPrimitive** widget class:

- **XmPrimitive Resource Set**

Inherited Resources

The following resource set contains a complete description of resource inherited by **XmPrimitive**.

- **Core Resource Set**

Behavior

Shift<Key>Tab: Moves the focus to the first item contained within the previous tab group. If the beginning of the tab group list is reached, it wraps to the end of the tab group list.

<Key>Tab or **<Key>F6:** Moves the focus to the first item contained within the next tab group. If the current tab group is the last entry in the tab group list, it wraps to the beginning of the tab group list.

<Key>Up or **<Key>Left:** Moves the keyboard focus to the previous **Primitive** widget or gadget within the current tab group. The previous widget or gadget is the one that is the previous entry in the tab group's list of children. Wrapping occurs, if necessary.

<Key>Down or **<Key>Right:** Moves the Keyboard focus to the next **Primitive** widget or gadget within the current tab group. The previous widget or gadget is the one that is the next entry in the tab group's list of children. Wrapping occurs, if necessary.

<Key>Home: Moves the keyboard focus to the first **Primitive** widget or gadget in the current tab group.

Default Translations

The following are the default translations for the **Primitive** widget:

<FocusIn>:	**PrimitiveFocusIn()**
<FocusOut>:	**PrimitiveFocusOut()**
<Unmap>:	**PrimitiveUnmap()**
Shift<Key>Tab:	**PrimitivePrevTabGroup()**
<Key>Tab:	**PrimitiveNextTabGroup()**
<Key>F6:	**PrimitiveNextTabGroup()**
<Key>Up:	**PrimitiveTraversePrev()**
<Key>Down:	**PrimitiveTraverseNext()**
<Key>Left:	**PrimitiveTraversePrev()**
<Key>Right:	**PrimitiveTraverseNext()**
<Key>Home:	**PrimitiveTraverseHome()**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **Core** widget class.

XmPushButton Widget Class

Purpose

The **PushButton** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/PushB.h>

Children

No children are supported.

Description

The **PushButton** widget issues commands within an application. It consists of a text label or pixmap surrounded by a border shadow. When the **PushButton** widget is selected, the shadow moves to give the appearance that it has been pressed in. When the **PushButton** widget is unselected, the shadow moves to give the appearance that it is out.

The behavior of the **PushButton** widget differs, depending on the active mouse button. The active mouse button may be determined by the parent widget. Normally, mouse button one is used to arm and activate the **PushButton** widget. However, if the **PushButton** widget resides within a menu, the mouse button used is determined by the **RowColumn** widget resources **XmNrowColumnType** and **XmNwhichButton**.

Thickness for a second shadow can be specified by using the **XmNshowAsDefault** resource. If it has a nonzero value, the **Label** widget resources **XmNmarginLeft, XmNmarginRight, XmNmarginTop,** and **XmNmarginBottom** can be modified to accommodate the second shadow.

The **XmPushButton** widget class inherits behavior and resources from the **Core**, **XmPrimitive**, and **XmLabel** classes. The class pointer is **xmPushButtonWidgetClass**. The class name is **XmPushButton**.

Subroutines

- **XmCreatePushButton**
- **XtCreateWidget**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources for the **PushButton** widget class:

- **XmPushButton Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmPushButton** widget class:

- **XmLabel Resource Set**
- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
int         reason;
XEvent      * event;
} XmAnyCallbackStruct;
```

reason — Is set to the value that corresponds to the type of selection that invoked this callback.

event — Points to the **XEvent** that invoked the callback. This event is **NULL** for the **XmNactivateCallback** if the callback was triggered when **Primitive's** resource **XmNtraversalOn** was **True** or if the callback was accessed through the **ArmAndActivate** action routine.

Behavior

The **PushButton** widget is associated with the default behavior unless it is part of a menu system. In a menu system, the **RowColumn** parent determines which mouse button is used.

Default Behavior

<Btn1Down>: This action causes the **PushButton** to be armed. The shadow is drawn in the armed state, and the button is filled with the color specified by **XmNarmColor** if **XmNfillOnArm** is set to True. The callbacks for **XmNarmCallback** are also called.

<Btn1Up>: (in button): This action redraws the shadow in the unarmed state. The background color will revert to the unarmed color if **XmNfillOnArm** is set to **True**. The callbacks for **XmNactivateCallback** are called, followed by callbacks for **XmNdisarmCallback**.

(outside of button): This action causes the callbacks for **XmNdisarmCallback** to be called.

<Leave Window>: If the button is pressed and the cursor leaves the widget window, the shadow is redrawn in its unarmed state, and the background color reverts to the unarmed color if **XmNfillOnArm** is set to **True**.

<Enter Window>: If the button is pressed and the cursor leaves and reenters the widget window, the shadow is drawn in the armed state, and the button is filled with the color specified by **XmNarmColor** if **XmNfillOnArm** is set to **True**.

Default PopupMenu System and OptionMenu System

<Btn2Down>: This action disables keyboard traversal for the menu and returns the user to drag mode, which is the mode in which the menu is manipulated by using the mouse. The shadow is drawn in the armed state, and the callbacks for **XmNarmCallback** are called.

<Btn2Up>: This action causes the **PushButton** widget to be activated and the menu to be unposted. The callbacks for **XmNactivateCallback** are called, followed by callbacks for **XmNdisarmCallback**.

<Leave Window>: If button two is pressed and the cursor leaves the widget window, the **PushButton** widget is redrawn with no shadow. The callbacks for **XmNdisarmCallback** are called. If keyboard traversal is enabled in the menu, then this event is ignored.

<Enter Window>: If button two is pressed and the cursor enters the widget window, the shadow is drawn in the armed state. The callbacks for **XmNarmCallback** are called. If keyboard traversal is enabled in the menu, this event is ignored.

<Key>Return: If keyboard traversal is enabled in the menu, this event causes the **PushButton** widget to be activated and the menu to be unposted. The callbacks for **XmNactivateCallback** are called, followed by callbacks for **XmNdisarmCallback**.

Default Pulldown Menu System

<Btn1Down>: This action disables keyboard traversal for the menu and returns the user to drag mode, which is the mode in which the menu is manipulated by using the mouse. The shadow is drawn in the armed state, and the callbacks for **XmNarmCallback** are called.

<Btn1Up>: This action causes the **PushButton** widget to be activated and the menu to be unposted. The callbacks for **XmNactivateCallback** are called, followed by callbacks for **XmNdisarmCallback**.

<Leave Window>: If mouse button one is pressed and the cursor leaves the widget window, the **PushButton** widget is redrawn with no shadow. The callbacks for **XmNdisarmCallback** are called. If keyboard traversal is enabled in the menu, this event is ignored.

<Enter Window>: If mouse button one is pressed and the cursor enters the widget window, the shadow is drawn in the armed state. The callbacks for **XmNarmCallback** are called. If keyboard traversal is enabled in the menu, this event is ignored.

<Key>Return: If keyboard traversal is enabled in the menu, this event causes the PushButton to be activated and the menu to be unposted. The callbacks for **XmNactivateCallback** are called, followed by callbacks for **XmNdisarmCallback**.

Default Translations

The default translations for the **PushButton** widget when not in a menu system are:

<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()**
	Disarm()
<Key>Return:	**ArmAndActivate()**
<Key>space:	**ArmAndActivate()**
<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**

The default translations for the **PushButton** widget when in a menu system are:

<BtnDown>:	**BtnDown()**
<BtnUp>:	**BtnUp()**
<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**
<Key>Return:	**KeySelect()**
<Key>Escape:	**MenuShellPopdownDone()**

Keyboard Traversal

Information on keyboard traversal when not in a menu system is contained in the **Primitive** widget and its sections on behavior and default translations. When in a menu system, the following keyboard traversal translations are defined:

<Unmap>:	**Unmap()**
<FocusOut>:	**FocusOut()**
<FocusIn>:	**FocusIn()**
<Key>space:	**Noop()**
<Key>Left:	**MenuTraverseLeft()**
<Key>Right:	**MenuTraverseRight()**
<Key>Up:	**MenuTraverseUp()**
<Key>Down:	**MenuTraverseDown()**
<Key>Home:	**Noop()**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/PushB.h

Related Information

The **Core** widget class, **XmCreatePushButton** subroutine, **XmLabel** widget class, **XmPrimitive** widget class, **XmRowColumn** widget class, **XtCreateWidget** subroutine.

XmPushButtonGadget Gadget

Purpose

The **PushButtonGadget** gadget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/PushBG.h>

Children

No children are supported.

Description

A **PushButtonGadget** gadget issues commands within a client application. This widget consists of a text label or icon surrounded by a border shadow. When the **PushButtonGadget** gadget is selected, the shadow moves to give the appearance that the **PushButtonGadget** gadget has been pressed in. When the **PushButtonGadget** gadget is deselected, the shadow moves to give the appearance that the **PushButtonGadget** gadget is out.

The behavior of the **PushButtonGadget** gadget differs, depending on the active mouse button. The active mouse button is determined by the parent widget. Normally, is used to arm and activate the **PushButtonGadget** gadget. However, if the **PushButtonGadget** gadget resides within a menu, mouse button use is determined by two **RowColumn** widget resources: **XmNrowColumnType** and **XmNwhichButton**.

Thickness for a second shadow can be specified by using the **XmNshowAsDefault** resource. If the resource has a nonzero value, the **Label** widget resources **XmNmarginLeft**, **XmNmarginRight**, **XmNmarginTop**, and **XmNmarginBottom** can be modified to accommodate the second shadow.

The **XmPushButtonGadget** gadget class inherits behavior and resources from the **Object**, **RectObj**, **XmGadget**, and **XmLabelGadget** classes. The class pointer is **xmPushButtonGadgetClass**. The class name is **XmPushButtonGadget**.

Subroutine

- **XmCreatePushButtonGadget**

New Resources

Setting the resource values for **Object**, **RectObj**, **XmGadget**, and **XmLabelGadget** classes also sets resources for the **XmPushButtonGadget** gadget class. To reference a resource in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.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**). The following resource set lists the resources of the **XmPushButtonGadget** widget class:

- **XmPushButtonGadget Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmPushButtonGadget** gadget class:

- **XmLabelGadget Resource Set**
- **XmGadget Resource Set**
- **RectObj Resource Set**
- **Object Resource Set**

Callback Information

```
typedef struct
{
        int     reason;
        XEvent  * event;
} XmAnyCallbackStruct;
```

reason Is set to the value that corresponds to the type of selection that invoked this callback routine.

event Points to the **XEvent** that invoked the callback routine. This event is **NULL** for the **XmNactivate** resource if the callback routine was triggered when the **Primitive** resource **XmNtraversalOn** was **True** or if the callback routine was accessed through the **ArmAndActivate** action routine.

Behavior

The **PushButtonGadget** gadget is associated with the default behavior unless it is part of a **Popup** menu system, a **Pulldown** menu system, or an **OptionMenu** system. In each menu system, the **RowColumn** parent determines which mouse button is used.

Default Behavior

<Btn1Down>: This action causes the **PushButtonGadget** gadget to be armed. The shadow is drawn in the armed state, and the button is filled with the color specified by the **XmNarmColor** resource if the **XmNfillOnArm** resource is set to **True**. The callback routines for the **XmNarmCallback** resource are also invoked.

<Btn1Up> (in button): This action redraws the shadow in the unarmed state. The background color reverts to the unarmed color if the **XmNfillOnArm** resource is set to **True**. The callback routines for the **XmNactivateCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Btn1Up> (outside of button): This action causes the callback routines for the **XmNdisarmCallback** resource to be invoked.

<Leave Window>: If the button is pressed and the cursor leaves the widget window, the shadow is redrawn in its unarmed state, and the background color reverts to the unarmed color if the **XmNfillOnArm** resource is set to **True**.

<Enter Window>: If the button is pressed and the cursor leaves and reenters the widget window, the shadow is drawn in the armed state, and the button is filled with the color specified by the **XmNarmColor** resource if the **XmNfillOnArm** resource is set to **True**.

Default PopupMenu System and OptionMenu System

<Btn2Down>: This action disables keyboard traversal for the menu and returns the user to drag mode (the mode in which the menu is manipulated by using the mouse). The shadow is drawn in the unarmed state, and the callback routines for the **XmNarmCallback** resource are invoked.

<Btn2Up>: This action causes the **PushButtonGadget** gadget to be activated and the menu to be unposted. The callback routines for the **XmNactivateCallback** are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<LeaveWindow>: If is pressed and the cursor leaves the widget window, the **PushButtonGadget** gadget is redrawn without a shadow. The callback routines for the **XmNdisarmCallback** resource are invoked. If keyboard traversal is enabled in the menu, this event is ignored.

<Enter Window>: If is pressed and the cursor enters the widget window, the shadow is drawn in the armed state. The callbacks for the **XmNarmCallback** resource are called. If keyboard traversal is enabled in the menu, this event is ignored.

<Key> Return: If keyboard traversal is enabled in the menu, this event causes the **PushButtonGadget** gadget to be activated and the menu to be unposted. The callback routines for the **XmNactivateCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

Default Pulldown Menu System

<Btn1Down>: This action disables keyboard traversal for the menu and returns the user to drag mode (the mode in which the menu is manipulated by using the mouse). The shadow is drawn in the armed state, and the callback routines for the **XmNarmCallback** resource are invoked.

<Btn1Up>: This action causes the **PushButtonGadget** gadget to be activated and the menu to be unposted. The callback routines for the **XmNactivateCallback** are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Leave Window>: If is pressed and the cursor leaves the widget window, the **PushButtonGadget** gadget is redrawn without a shadow. The callback routines for the **XmNdisarmCallback** resource are invoked. If keyboard traversal is enabled in the menu, this event is ignored.

<Enter Window>: If is pressed and the cursor enters the widget window, the shadow is drawn in the armed state. The callback routines for the **XmNarmCallback** resource are invoked. If keyboard traversal is enabled in the menu, this event is ignored.

<Key> Window: If keyboard traversal is enabled in the menu, this event causes the **PushButtonGadget** gadget to be activated and the menu to be unposted. The callback routines for the **XmNactivateCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

Keyboard Traversal

The **XmGadget** gadget class and its sections on behavior and default translations contain information on keyboard traversal.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/PushBG.h

Related Information

The **XmCreatePushButtonGadget** subroutine, **XmLabel** widget class, **XmLabelGadget** gadget class, **XmGadget** gadget class, **RectObj** widget class, **Object** widget class, **XmRowColumn** widget class.

XmRowColumn Widget Class

Purpose

The **RowColumn** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Children

ArrowButton Widget
CascadeButton Widget
DrawingArea Widget
Form Widget
Label Widget
MessageBox Widget
PushButton Widget
ScrollBar Widget
SelectionBox Widget
Text Widget
BulletinBoard Widget
Command Widget
DrawnButton Widget
Frame Widget
MenuShell Widget
PanedWindow Widget
Scale Widget
ScrolledWindow Widget
Separator Widget
ToggleButton Widget

Description

The **RowColumn** widget is a general–purpose **RowColumn** widget manager capable of containing any widget type as a child. It requires no special knowledge about how its children function and provides nothing beyond support for several different layout styles. If it is configured as a menu, it expects only certain children widgets, and it configures to a particular layout. The menus supported are: **MenuBar**, **Pulldown** or **Popup MenuPanes**, and **OptionMenu**.

The type of menu system is controlled by how the client application has set the various layout resources. It can be configured to lay out its children in either rows or columns. In addition, the application can specify whether the children should be packed tightly together (not into organized rows and columns), or whether each child should be placed in an identically–sized box (producing a symmetical look), or whether specific layout should be done (the current *x* and *y* positions of the children control their location).

In addition, the client application has control over the spacing that occurs between each row and column and the margin spacing present between the edges of the **RowColumn** widget and any children that are placed against it.

In most cases, the **RowColumn** widget has no three–dimensional visuals associated with it; if an application wishes to have a three–dimensional shadow placed around this widget, it can create the **RowColumn** widget as a child of a **Frame** widget.

The **XmRowColumn** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, and **XmManager** classes. The class pointer is **xmRowColumnWidgetClass**. The class name is **XmRowColumn**.

Subroutines

- **XmCreateRowColumn**
- **XmGetMenuCursor**
- **XmMenuPosition**
- **XmOptionButtonGadget**
- **XmOptionLabelGadget**
- **XmSetMenuCursor**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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)**. The following resource sets list the resources of the **XmRowColumn** widget class:

- **XmRowColumn Resource Set**
- **XmRowColumn Special Menu Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **RowColumn** widget:

- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int       reason;
        XEvent    * event;
        Widget    widget;
        char      * data;
        char      * callbackstruct;
} XmRowColumnCallbackStruct;
```

reason Indicates why the callback routine was invoked.

event Points to the **XEvent** that invoked the callback routine.

The following fields apply only when the callback reason is **XmCR_ACTIVATE**; for all other callback reasons, these fields are set to **NULL**. The **XmCR_ACTIVATE** callback reason is only generated when the application has supplied an entry callback routine, which overrides any activation callback routines registered with the individual **RowColumn** widget items.

widget	Is set to the widget ID of the **RowColumn** widget item that has been activated.
data	Contains the client–data value supplied by the client application when the **RowColumn** widget item activation callback routine was registered.
callbackstruct	Points to the callback structure generated by the **RowColumn** widget item activation callback.

Behavior

The behavior associated with a **RowColumn** widget depends on its type (such as **MenuBar** and **Popup MenuPane**) and the type of menu system in which it resides (**Pulldown**, **Popup**, or **Option**). The specific mouse button depends on the **XmNwhichButton** resource.

Default MenuBar

<Btn1Down>: If the button event occurs within one of the **MenuBar** buttons, the **MenuBar** is armed (if not already armed) and the submenu associated with the selected button is posted. Mouse provides access to the **MenuPanes** attached to the **MenuBar**.

If the button event does not occur within one of the **MenuBar** buttons and if the **MenuBar** is already armed, it is disarmed, and any visible **MenuPanes** are unposted; if the **MenuBar** is not already armed, nothing occurs.

<Btn1Up>: If the **MenuBar** is armed, this event unposts all visible **MenuPanes** and then disarms the menubar.

Default OptionMenu

<Btn1Down>: When this event occurs within the selection area, the **Pulldown MenuPane** is posted. If this event occurs outside of the selection area and the **MenuPane** is already posted, the **Pulldown MenuPane** is unposted.

<Btn1Up>: When this event occurs while the **Pulldown MenuPane** is posted, then it is unposted.

<Return>: If this key is pressed while the focus is set to the selection area, then the **Pulldown MenuPane** is posted.

Default Pulldown MenuPane from a Popup MenuPane

<Btn3Down>: When this event occurs, the menu system disables traversal mode, and re–enters drag mode. Depending upon where the button–down event occurs, certain portions of the visible set of **MenuPanes** are unposted.

<Btn3Up>: When this event occurs within a gadget child of the **MenuPane**, the indicated child is activated. If the child is not a **CascadeButton** (widget or gadget), this also results in all visible **MenuPanes** being unposted. If the child is a **CascadeButton** (widget or gadget), the associated submenu is posted and traversal is enabled. When this event occurs outside of a gadget child, all visible **MenuPanes** are unposted.

<Return>: If this key is pressed while the focus is set to a gadget child of the **MenuPane**, the indicated child is activated. If the child is not a **CascadeButton** (widget or gadget), then this also results in all visible **MenuPanes** being unposted. If the child is a **CascadeButton** (widget or gadget), this results in the associated submenu being posted and traversal being enabled.

<Escape>: This event unposts all visible **MenuPanes**.

<Right>: If the current focus item is a **CascadeButtonGadget**, then this posts the associated **Pulldown MenuPane** and highlights the first accessible item within the **Pulldown MenuPane**.

<Left>: If this occurs within a **MenuPane** that is a submenu of another **MenuPane**, this causes the last **MenuPane** to be unposted and the focus to move to the previous **MenuPane**.

<Up>: This moves the focus to the previous menu item; the previous menu item is defined as the widget created prior to the one that currently has the focus. Wrapping occurs, if necessary.

<Down>: This moves the focus to the next menu item; the next menu item is defined as the widget created after the one that currently has the focus. Wrapping occurs, if necessary.

Default Pulldown MenuPane from a MenuBar or from an OptionMenu

<Btn1Down>: When this event occurs, the menu system disables traversal mode and re–enters drag mode. Depending upon where the button down event occurs, certain portions of the visible set of **MenuPanes** are unposted.

<Btn1Up>: When this event occurs within a gadget child of the **MenuPane**, the indicated child is activated. If the child is not a **CascadeButton** (widget or gadget), this also results in all visible **MenuPanes** being unposted. If the child is a **CascadeButton** (widget or gadget), this is results in the associated submenu being posted and traversal being enabled. When this event occurs outside of a gadget child, then all visible **MenuPanes** are unposted.

<Return>: If this key is pressed while the focus is set to a gadget child of the **MenuPane**, then the indicated child is activated. If the child is not a **CascadeButton** (widget or gadget), this also results in all visible **MenuPanes** being unposted. If the child is a **CascadeButton** (widget or gadget), then this results in the associated submenu being posted and traversal being enabled.

<Escape>: This event unposts all visible **MenuPanes**.

<Right>: If the current focus item is a **CascadeButtonGadget**, this posts the associated **Pulldown MenuPane** and highlights the first accessible item within the **Pulldown MenuPane**. If the current focus item is not a **CascadeButton**, then the visible set of **MenuPanes** are unposted, and the top level **Pulldown MenuPane** associated with the next **MenuBar** is posted.

<Left>: If this occurs within a **MenuPane** that is a submenu of another **MenuPane**, this event causes the last **MenuPane** to be unposted and the focus to move to the previous **MenuPane**. If this occurs within a **MenuPane** that is connected directly to the **MenuBar**, then the visible set of **MenuPanes** are unposted, and the top level **Pulldown MenuPane** associated with the previous menubar item is posted.

<Up>: This moves the focus to the previous menu item; the previous menu item is defined as the widget created prior to the one that currently has the focus. Wrapping occurs, if necessary.

<Down>: This moves the focus to the next menu item; the next menu item is defined as the widget created after the one that currently has the focus. Wrapping occurs, if necessary.

<Btn1Down>: If the button press occurred in a gadget child, it is dispatched to it.

<Btn1Up>: If the button press occurred in a gadget child, it is dispatched to it.

Default Translations

For an **OptionMenu**, the default translations are:

<BtnDown>:	**PopupBtnDown()**
<BtnUp>:	**PopupBtnUp()**
<Key>Return:	**MenuGadgetReturn()**

For a **Popup MenuPane**, the default translations are:

<BtnDown>:	**PopupBtnDown()**
<BtnUp>:	**PopupBtnUp()**
<Key>Return:	**MenuGadgetReturn()**
<Key>Escape:	**MenuGadgetEscape()**
<Unmap>:	**MenuUnmap()**
<FocusIn>:	**MenuFocusIn()**
<FocusOut>:	**MenuFocusOut()**
<EnterWindow>:	**MenuEnter()**
<Key>Left:	**MenuGadgetTraverseLeft()**
<Key>Right:	**MenuGadgetTraverseRight()**
<Key>Up:	**MenuGadgetTraverseUp()**
<Key>Down:	**MenuGadgetTraverseDown()**

For a **Pulldown MenuPane**, the default translations are:

<BtnDown>:	**PulldownBtnDown()**
<BtnUp>:	**PulldownBtnUp()**
<Key>Return:	**MenuGadgetReturn()**
<Key>Escape:	**MenuGadgetEscape()**
<Unmap>:	**MenuUnmap()**
<FocusIn>:	**MenuFocusIn()**
<FocusOut>:	**MenuFocusOut()**
<EnterWindow>:	**MenuEnter()**
<Key>Left:	**MenuGadgetTraverseLeft()**
<Key>Right:	**MenuGadgetTraverseRight()**
<Key>Up:	**MenuGadgetTraverseUp()**
<Key>Down:	**MenuGadgetTraverseDown()**

For a **MenuBar**, the default translations are:

<BtnDown>:	**MenuBarBtnDown()**
<BtnUp>:	**MenuBarBtnUp()**
<Unmap>:	**MenuUnmap()**
<FocusIn>:	**MenuFocusIn()**
<FocusOut>:	**MenuFocusOut()**
<EnterWindow>:	**MenuEnter()**

For a **WorkArea**, the default translations are:

<Btn1Down>:	**WorkAreaBtnDown()**
<Btn1Up>:	**WorkAreaBtnUp()**

Keyboard Traversal

The description of the **XmManager** widget class contains information on keyboard traversal in a **WorkArea**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h

Related Information

The **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmCreateMenuBar** subroutine, **XmCreateOptionMenu** subroutine, **XmCreatePopupMenu** subroutine, **XmCreatePulldownMenu** subroutine, **XmCreateRadioBox** subroutine, **XmGetMenuCursor** subroutine, **XmLabel** widget class, **XmManager** widget class, **XmSetMenuCursor** subroutine, **XmMenuPosition** subroutine, **XmOptionButtonGadget** subroutine, **XmOptionLabelGadget** subroutine, **XmUpdateDisplay** subroutine, **XmMenuPosition** subroutine.

XmScale Widget Class

Purpose

The **Scale** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include<Xm/Scale.h>

Children

No children are supported.

Description

A **Scale** widget class is used by a client application to select a value from within a range of values, and it allows the user to input or modify a value from the same range.

A **Scale** widget class has an elongated rectangular region similar to a **ScrollBar** widget. A slider inside this region indicates the current value along the **Scale**. The user can also modify the **Scale** value by moving the slider within the rectangular region of the **Scale** widget. A **Scale** widget can also include a label set outside the **Scale** widget region. The label set can be used to indicate the relative value at various positions along the scale.

A **Scale** widget can be either input/output or output only. An input/output **Scale** widget value can be set by the client application and modified through use of the slider. An output only **Scale** widget is used strictly as an indicator of current value and cannot be modified interactively. The **Core** resource **XmNsensitive** specifies whether the **Scale** widget value can be modified interactively.

The **XmScale** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, and **XmManager** classes. The class pointer is **xmScaleWidgetClass**. The class name is **XmScale**.

Subroutines

- **XmCreateScale**
- **XmScaleGetValue**
- **XmScaleSetValue**

New Resources

Setting the resource values for inherited classes also sets resources for the **XmScale** widget. To reference a resource in a **.Xdefaults** file, remove the first three letters (**XmN**) from the resource name 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 lower case or upper case, 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**). The following resource set lists the resources of the **XmScale** class:

- **XmScale Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **Scale** widget:

- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int         reason;
        XEvent      * event;
        int         value;
} XmScaleCallbackStruct;
```

reason Indicates why the callback was invoked.

event Points to the **XEvent** that invoked the callback.

value Is the new slider location value.

Behavior

<Btn1Down>: Activates the interactive dragging of the slider if the button is pressed anywhere inside of the scale rectangle, including the slider.

Button1<PtrMoved>: Moves the slider to the new position and calls the callback routines for **XmNdragCallback** if the button press occurs within the slider.

<Btn1Up>: Calls the callback routines for **XmNvalueChangedCallback** if the button press occurs within the scale rectangle, and if the slider position was changed.

Default Translations

Button assignments are: Left Button is Button 1; Left Button AND Right Button are Button 2; and Right Button is Button 3.

<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()**
<EnterWindow>:	**Enter()**
<FocusIn>:	**FocusIn()**

Keyboard Traversal

Information on keyboard traversal is contained in the **XmManager** widget class and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Scale.h

Related Information

The **Constraint** widget class, **Composite** widget class, **Core** widget class, **XmCreateScale** subroutine, **XmManager** widget class, **XmScaleGetValue** subroutine, **XmScaleSetValue** subroutine.

XmScrollBar Widget Class

Purpose

A **ScrollBar** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ScrollBar.h>

Children

No children are supported.

Description

A **ScrollBar** widget allows the user to view data that is too large to be displayed at one time. **ScrollBar** widgets are usually located beside or within the widget that contains the data to be viewed. When the user interacts with the **ScrollBar** widget, the data within the other widget scrolls.

A **ScrollBar** widget consists of two arrows, located at opposite ends of a rectangle called the scroll region. A smaller rectangle, called the slider, is placed within the scroll region. Data is scrolled by selecting on either arrow or the scroll region, or by dragging the slider. When an arrow is selected, the slider within the scroll region is moved in the direction of the arrow by an amount supplied by the client application. If the mouse button is held down, the slider continues to move at a constant rate.

The ratio of the slider size and the scroll region size corresponds to the relationship between the size of the visible data and the total size of the data. For example, if ten percent of the data is visible, the slider occupies ten percent of the scroll region. This provides the user with a visual clue to the size of the invisible data.

The **XmScrollBar** widget class inherits behavior and resources from the **Core** and **XmPrimitive** classes. The class pointer is **xmScrollBarWidgetClass**. The class name is **XmScrollBar**.

Subroutines

- **XmCreateScrollBar**
- **XmScrollBarGetValues**
- **XmScrollBarSetValues**
- **XtCreateWidget**

New Resources

Setting the resource values for inherited classes also sets resources for the **XmScrollBar** widget class. To reference a resource in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **ScrollBar** class:

- **XmScrollBar Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmScrollBar** widget class:

- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback routine:

```
typedef struct
{
int         reason;
XEvent      * event;
int         value;
int         pixel;
} XmScrollBarCallbackStruct;
```

reason — Indicates why the callback routine was invoked.

event — Points to the **XEvent** that invoked the callback routine.

value — Contains the new slider location value.

pixel — Is used only for **XmNtoTopCallback** and **XmNtoBottomCallback**. For horizontal ScrollBars, it contains the x coordinate of where the mouse button selection occurred. For vertical ScrollBars, it contains the y coordinate.

Behavior

<Btn1Down>(in arrow): Moves the slider one increment (or decrement) in the direction of the arrow and invokes the callback routines for the **XmNincrementCallback** resource or the **XmNdecrementCallback** resource. The **XmNvalueChangedCallbacks** is called if the **XmNincrementCallbacks** or **XmNdecrementCallbacks** resource is empty.

<Btn1Down>(in scroll region): Moves the slider one page increment or page decrement depending on the which side of the slider is selected and invokes the callback routines for **XmNpageIncrementCallback** or **XmNpageDecrementCallback**. The **XmNvalueChangedCallbacks** is called if the **XmNpageIncrementCallbacks** or **XmNpageDecrementCallbacks** resource is empty.

<Btn1Down>(in slider): Activates the interactive dragging of the slider. If the button is held down in either the arrows or scroll region longer than the **XmNinitialDelay** resource, the slider is moved again by the same increment and the same callback routines are called. After the initial delay has been used, the time delay changes to the time defined by the resource **XmNrepeatDelay**.

Button1<PtrMoved>: If the button press occurs within the slider, the subsequent motion events move the slider to the new position and the callback routines for **XmNdragCallback** are invoked.

<Btn1Up>: If the button press occurred within the slider and the slider position was changed, the callback routines for **XmNvalueChangedCallback** are invoked.

Shift<Btn1Down>: This mouse button press in the top arrow button causes the callback routines for **XmNtoTopCallback** to be called.

Shift<Btn1Down>: This mouse button press in the bottom arrow button causes the callback routines for **XmNtoBottomCallback** to be called.

<Key>Up: For vertical ScrollBars, pressing the up arrow cursor key decrements the slider one unit and calls **XmNdecrementCallback**. The **XmNvalueChangedCallbacks** is called if the **XmNdecrementCallbacks** resource is empty.

<Key>Down: For vertical ScrollBars, pressing the down arrow cursor key increments the slider one unit and calls **XmNincrementCallback**. The **XmNvalueChangedCallbacks** is called if the **XmNincrementCallbacks** resource is empty.

<Key>Left: For horizontal ScrollBars, pressing the left arrow cursor key decrements the slider one unit and calls **XmNdecrementCallback**. The **XmNvalueChangedCallbacks** is called if the **XmNdecrementCallbacks** resource is empty.

<Key>Right: For horizontal ScrollBars, pressing the right arrow cursor key increments the slider one unit and calls **XmNincrementCallback**. The **XmNvalueChangedCallbacks** is called if the **XmNincrementCallbacks** resource is empty.

Default Translations

~Shift ~Ctrl ~Meta ~Alt <Btn1Down>:	**Select( )**
~Shift ~Ctrl ~Meta ~Alt <Btn1Up>:	**Release( )**
~Shift ~Ctrl ~Meta ~Alt Button1<PtrMoved>:	**Moved( )**
Shift ~Ctrl ~Meta ~Alt <Btn1Down>:	**GoToTop( )**
Shift ~Ctrl ~Meta ~Alt <Btn1Down>:	**GoToBottom( )**
~Shift ~Ctrl ~Meta ~Alt <Key>Up:	**UpOrLeft(0)**
~Shift ~Ctrl ~Meta ~Alt <Key>Down:	**DownOrRight(0)**
~Shift ~Ctrl ~Meta ~Alt <Key>Left:	**UpOrLeft(1)**
~Shift ~Ctrl ~Meta ~Alt <Key>Right:	**DownOrRight(1)**
<EnterWindow>:	**Enter( )**
<LeaveWindow>:	**Leave( )**

Keyboard Traversal

When the **XtNtraversalOn** resource is set to **True** at create time or during a call to **XtSetValues**, the **XmManager** widget superclass automatically augments the **Manager** widget translations to support keyboard traversal. The description of the **XmManager** widget class contains a complete description of these translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ScrollBar.h

Related Information

The **Core** widget class, **XmCreateScrollBar** subroutine, **XmPrimitive** widget, **XmScrollBarGetValues** subroutine, **XmScrollBarSetValues** subroutine, **XtCreateWidget** subroutine.

XmScrolledWindow Widget

Purpose

The **ScrolledWindow** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ScrolledW.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
MessageBox Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget

Description

A **ScrolledWindow** widget combines one or more **ScrollBar** widgets and a viewing area to implement a visible window onto some other (usually larger) data display. The visible part of the window can be scrolled through the larger display by the use of the **ScrollBars**.

To use a **ScrolledWindow** widget, a client application creates a **ScrolledWindow** widget, any needed **ScrollBar** widgets, and a widget capable of displaying any desired data as the work area of the **ScrolledWindow** widget. The **ScrolledWindow** widget positions the work–area widget and display the scroll bars if so requested. When the user performs some action on the **ScrollBar** widget, the application is notified through the normal **ScrollBar** callback interface.

The **ScrolledWindow** widget can be configured to operate in an automatic manner; all scrolling and display actions are performed automatically with no need for application program involvement. It can also be configured to provide a minimal support framework in which the application is responsible for processing all user input and making all visual changes to the displayed data in response to that input.

When the **ScrolledWindow** widget is performing automatic scrolling, it creates a clipping window. Conceptually, this window becomes the view port through which the user examines the larger underlying data area. The application simply creates the desired data, and then makes that data the work area of the **ScrolledWindow** widget. When the user moves the

slider to change the displayed data, the workspace is moved under the viewing area so that a new portion of the data becomes visible.

There are instances where it is impractical for an application to create a large data space and simply display it through a small clipping window. An example of this is a text editor in which there would be an undesirable amount of overhead involved with creating a single data area that consisted of a large file. The application would want to use the concept of a **ScrolledWindow** widget (a small viewport onto some larger data), but would want to be notified when the user scrolled the viewport so it could bring in more data from storage and update the display area. For these cases the **ScrolledWindow** widget can be configured so that it provides only visual layout support. No clipping window is created, and the application must maintain the data displayed in the work area, as well as respond to user input on the **ScrollBars**.

The **XmScrolledWindow** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, and **XmManager** classes. The class pointer is **xmScrolledWindowWidgetClass**. The class name is **XmScrolledWindow**.

Subroutines

- **XmCreateScrolledWindow**
- **XmScrolledWindowSetAreas**
- **XmCreateScrolledList**
- **XmCreateScrolledText**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmScrolledWindow** class:

- **XmScrolledWindow Resource Set**

Inherited Resources

The following resource sets contain a complete description of the superclass resources inherited by the **XmScrolledWindow** widget class:

- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The **XmScrolledWindow** widget class defines no new callback structures. The client application must use the **ScrollBar** widget callback routines to be notified of user input.

Behavior

The **XmScrolledWindow** widget class makes extensive use of the **XtQueryGeometry** functionality to facilitate geometry communication among application levels. In the

XmAPPLICATION_DEFINED scrolling policy, the **WorkWindow** query procedure is called by the **ScrolledWindow** widget whenever the **ScrolledWindow** widget is going to change its size. The widget calculates the largest possible workspace area and passes this size to the WorkWindow query procedure. The query procedure can then examine this new size and determine if any changes are necessary, such as managing or unmanaging a ScrollBar. The query procedure performs the required actions and then returns to the **ScrolledWindow** widget. The **ScrolledWindow** widget examines the scroll bars to see which (if any) are managed, and then allocates a portion of the visible space for them, and resizes the WorkWindow to fit in the rest of the space.

When the scrolling policy is the **XmCONSTANT** value, the **XmScrolledWindow** widget can be queried to return the optimal size for a given dimension. The optimal size is defined to be the size that would just enclose the WorkWindow. By using this mechanism, a client application can size the **ScrolledWindow** widget so that it only needs to display a ScrollBar for one dimension. When the **ScrolledWindow** widget's query procedure is called via **XtQueryGeometry**, the request is examined to see if the width or height has been specified. If so, the routine uses the given dimension as the basis for its calculations. It determines the minimum value for the other dimension that encloses the WorkWindow, fills in the appropriate elements in the reply structure, and returns to the calling program. Occasionally, using the specified width or height and the other minimum dimension results in neither scroll bar appearing. When this happens, the query procedure sets both the width and height fields, indicating that in this situation the ideal size causes a change in both dimensions. If the calling application sets both the width and height fields, the **ScrolledWindow** widget determines the minimum size for both dimensions and return those values in the reply structure.

Keyboard Traversal

The **XmManager** widget class and its sections on behavior and default translations contain information on keyboard traversal.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ScrolledW.h

Related Information

The **Constraint** widget class, **Composite** widget class, **Core** widget class, **XmManager** widget, **XmCreateScrolledWindow** subroutine, **XmScrolledWindowSetAreas** subroutine.

XmSelectionBox Widget Class

Purpose

The **SelectionBox** widget class.

Libraries

AIXwindows Library (**libXm.a**)

AIXwindows Library (**libIM.a**)

Syntax

#include <Xm/SelectioB.h>

Children

ArrowButton Widget
BulletinBoard Widget
CascadeButtonGadget Gadget
DialogShell Widget
DrawnButton Widget
Form Widget
Label Widget
List Widget
MenuShell Widget
PanedWindow Widget
PushButtonGadget Gadget
Scale Widget
ScrolledWindow Widget
Separator Widget
Text Widget
ToggleButtonGadget Gadget
ArrowButtonGadget Gadget
CascadeButton Widget
Command Widget
DrawingArea Widget
FileSelectionBox Widget
Frame Widget
LabelGadget Gadget
MainWindow Widget
MessageBox Widget
PushButton Widget
RowColumn Widget
ScrollBar Widget
SelectionBox Widget
SeparatorGadget Gadget
ToggleButton Widget

Description

The **SelectionBox** widget is a general dialog widget that permits the selection of one item out of a list of items. A **SelectionBox** widget includes the following:

- A scrolling list of alternatives
- An editable text field for the selected alternative
- Labels for the list and text field
- Three buttons

Note: You should be aware of the proper usage of the **XmText** widget class before using this widget class.

The default button labels are **OK**, **Cancel**, and **Help**. An **Apply** button is created unmanaged and can be explicitly managed as needed. One additional **WorkArea** child can be added to the **SelectionBox** widget after creation.

The user can select an item in two ways:

- By scrolling through the list of items and selecting the desired item
- By entering the item name directly into the text edit area

Selecting an item from the list causes that item name to appear in the selection text edit area.

New items may be selected as many times as desired. The item is not actually selected until the user selects the **OK PushButton**.

The **XmSelectionBox** widget class inherits behavior and resources from the **Core**, **Composite**, **Constraint**, **XmManager** and **XmBulletinBoard** classes. The class pointer is **xmSelectionBoxWidgetClass**. The class name is **XmSelectionBox**.

Subroutines

- **XmCreateSelectionBox**
- **XmCreateSelectionDialog**
- **XmCreatePromptDialog**
- **XmSelectionBoxGetChild**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmSelectionBox** class:

- **XmSelectionBox Resource Set**

Inherited Resource

The following resource sets contain a complete description of the resources inherited by the **SelectionBox** widget:

- **XmBulletinBoard Resource Set**
- **XmManager Resource Set**
- **Composite Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int             reason;
        XEvent          * event;
        XmString value;
        int             length;
} XmSelectionBoxCallbackStruct;
```

reason Indicates why the callback routine was invoked.

event Points to the **XEvent** that triggered the callback routine.

value Indicates the **XmString** value selected by the user from the **SelectionBox** widget list or entered into the **SelectionBox** widget text field.

length Indicates the size in bytes of the **XmString** value.

Behavior

The following is a summary of the behavior of the **SelectionBox** widget:

<Ok Button Activated>: When the **OK** button is activated, **XmNokCallback** is invoked. The callback reason is **XmCR_OK**. An invalid selection that does not match any items in the list triggers the callback routine for **XmNnoMatchCallback**, but only if **XmNmustMatch** is also **True**. The callback reason is **XmCR_NO_MATCH**.

<Apply Button Activated>: When the apply button is activated, **XmNokCallback** is invoked. The callback reason is **XmCR_APPLY**. An invalid selection that does not match any items in the list triggers the callback routines for **XmNnoMatchCallback**, but only if **XmNmustMatch** is also **True**. The callback reason is **XmCR_NO_MATCH.**

<Cancel Button Activated>: When the **Cancel** button is activated, the callback **XmNcancelCallback** is called. The callback reason is the **XmCR_CANCEL** value.

<Help Button Activated> or **<Key>F1:** When the help button or **subroutine key 1** is pressed, the callback routines for **XmNhelpCallback** are invoked.

<Default Button Activated> or **<Key>Return**: When the default button or return key is pressed, a callback routine for one of the following resources is invoked: **XmNokCallback**, **XmNapplyCallback**, **XmNcancelCallback**, or **XmNhelpCallback**.

<Key>Up or <Key>Down: When the up or down key is pressed within the **Text** subwidget of the **SelectionBox** widget, the text value is replaced with the previous or next item in the **List** subwidget.

<FocusIn>: When a **FocusIn** *Event* is generated on the widget window, the callback routines for **XmNfocusCallback** are called.

<MapWindow>: When a **SelectionBox** widget that is the child of a **DialogShell** widget is mapped, the callback routines for **XmNmapCallback** are invoked. When a **SelectionBox** widget that is not the child of a **DialogShell** is mapped, the callback routines are not invoked.

<UnmapWindow>: When a **SelectionBox** widget that is the child of a **DialogShell** widget is unmapped, the callback routines for the **XmNunmapCallback** resource are invoked.

When a **SelectionBox** widget that is not the child of a **DialogShell** widget is unmapped, the callback routines are not invoked.

Default Translations

The default translations defined for the **SelectionBox** widgets are:

<EnterWindow>:	**Enter()**
<FocusIn>:	**FocusIn()**
<Btn1Down>:	**Arm()**
<Btn1Up>:	**Activate()**
<Key>F1:	**Help()**
<Key>Return:	**Return()**
<Key>KP_Enter:	**Return()**

Default Accelerators

The following are the default accelerator translations added to the descendents of an **SelectionBox** widget:

#override

<Key>F1:	**Help( )**
<Key>Return:	**Return( )**
<Key>KP_Enter:	**Return( )**

Default Text Accelerators

The following are the default accelerators added to the **Text** widget child of the **SelectionBox** widget:

#override

<Key>Up:	**UpOrDown(0)**
<Key>Down:	**UpOrDown(1)**
<Key>F1:	**Help( )**
<Key>Return:	**Return( )**
<Key>KP_Enter:	**Return( )**

Keyboard Traversal

The description of the **XmManager** widget contains additional information on keyboard traversal.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/SelectioB.h

Related Information

The **XmBulletinBoard** widget class, **Composite** widget class, **Constraint** widget class, **Core** widget class, **XmManager** widget class, **XmCreateSelectionBox** subroutine, **XmCreateSelectionDialog** subroutine, **XmCreatePromptDialog** subroutine, **XmSelectionBoxGetChild** subroutine.

XmSeparator Widget Class

Purpose

The **Separator** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Separator.h>

Children

No children are supported.

Description

A **Separator** widget is a primitive widget that separates items in a display. Several different line–drawing styles are provided as well as horizontal or vertical orientation.

The **Separator** widget line drawing is automatically centered within the height of the gadget for a horizontal orientation and centered within the width of the widget for a vertical orientation. An **XtSetValues** subroutine with a new **XmNseparatorType** resource resizes the widget to its minimal height (for horizontal orientation) or its minimal width (for vertical orientation) unless height or width is explicitly set in the **XtSetValues** call.

The **Separator** widget does not draw shadows. The **Primitive** resource **XmNshadowThickness** is used for the **Separator** widget thickness when **XmNshadowType** is **XmSHADOW_ETCHED_IN** or **XmSHADOW_ETCHED_OUT**.

The **Separator** widget does not highlight and allows no traversing. The **Primitive** widget resource **XmNtraversalOn** is forced to **False**.

The **XmNseparatorType** of **XmNO_LINE** provides an escape for client applications requiring a different style of drawing. A pixmap the height of the widget can be created and used as the background pixmap by building an argument list using the **XmNbackgroundPixmap** resource type as defined by **Core**. Whenever the widget is redrawn, its background is displayed containing the desired separator drawing.

The **XmSeparator** widget class inherits behavior and resources from the **Core** and **XmPrimitive** classes. The class pointer is **xmSeparatorWidgetClass**. The class name is **XmSeparator**.

Subroutines

- **XmCreateSeparator**
- **XtCreateWidget**
- **XtSetValues**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file,

remove the **XmN** or **XmC** prefix and use the remaining letters (in either lower or upper case, 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**). The following resource set lists the resources of the **XmSeparator** widget class:

- **XmSeparator Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmSeparator** widget class:

- **XmPrimitive Resource Set**
- **Core Resource Set**

Keyboard Traversal

The description of the **XmPrimitive** widget class and its sections on behavior and default translations contain additional information on keyboard traversal.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Separator.h

Related Information

The **Core** widget class, **XmCreateSeparator** subroutine, **XmPrimitive** widget, **XtCreateWidget** subroutine, **XtSetValues** subroutine.

XmSeparatorGadget Gadget Class

Purpose

The **SeparatorGadget** gadget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/SeparatoG.h>

Children

No children are supported.

Description

A **SeparatorGadget** gadget separates items in a display. Several line–drawing styles are provided, as well as horizontal or vertical orientation.

Lines drawn within the **SeparatorGadget** widget are automatically centered within the height of the gadget for a horizontal orientation and centered within the width of the gadget for a vertical orientation. An **XtSetValues** subroutine with a new **XmNseparatorType** resource resizes the widget to its minimal height (for horizontal orientation) or its minimal width (for vertical orientation) unless height or width is explicitly set in the **XtSetValues** call.

The **SeparatorGadget** gadget does not draw shadows. The **Gadget** resource **XmNshadowThickness** is used for the **SeparatorGadget** widget thickness when the **XmNshadowType** resource is the **XmSHADOW_ETCHED_IN** value or the **XmSHADOW_ETCHED_OUT** value.

The **SeparatorGadget** gadget does not highlight and allows no traversing. The **Gadget** resource **XmNtraversalOn** is forced to **False**.

The **XmSeparatorGadget** gadget class inherits behavior and resources from the **Object**, **RectObj**, and **XmGadget** classes. The class pointer is **xmSeparatorGadgetClass**. The class name is **XmSeparatorGadget**.

Subroutine

- **XmCreateSeparatorGadget**

New Resources

Setting the resource values for the inherited classes also sets resources for this gadget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **SeparatorGadget** class:

- **XmSeparatorGadget Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmSeparatorGadget** gadget class:

- **XmGadget Resource Set**
- **RectObj Resource Set**
- **Object Resource Set**

Keyboard Traversal

The description for the **XmGadget** gadget class contains information on behavior.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/SeparatoG.h

Related Information

The **XmCreateSeparatorGadget** subroutine, **XmGadget** widget, **Object** widget class, **RectObj** widget class, **XtCreateWidget** subroutine, **XtSetValues** subroutine.

XmText Widget Class

Purpose

A **Text** widget class.

Libraries

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

Children

No children are supported.

Description

The **Text** widget provides a single and multiline text editor for customizing user interfaces and programmatic interfaces. This widget can be used for single–line string entry, forms entry with verification procedures, and full–window editing. This widget provides client applications with a consistent editing system for textual data. The screen textual data adjusts to the client application's requirements.

The **Text** widget provides separate callback routine lists to verify movement of the insert cursor, modification of the text, and changes in input focus. Each of these callback routines provides the verification subroutine with the widget instance, the event that resulted in the callback, and a data structure specific to the verification type. From this information the subroutine can verify that the application considers this to be a legitimate state change and can signal the widget whether or not it should continue with the action.

The user interface tailors a new set of translations. The default translations provide key bindings for insert cursor movement, deletion, insertion, and selection of text.

The **Text** widget supports the selection of regions of text. Selection is based on the Interclient Communication Conventions (ICCC) selection model. The primary text selection is supported.

The **Text** widget controls the data structures for drawing the text on the screen and defines the subroutines that manipulate that data. A separate component, called the **Source**, provides textual data storage and a set of subroutines for querying and changing that data.

```
setlocale (LC_CTYPE, "Fr_FR.pc850")
XmCreateText( .....)
```

This would create a **Text** widget supporting a french keyboard, providing you are using a font that is capable of displaying french characters. If a locale, specified through setlocale, cannot be recognized, the default keymap, En_US.pc850 is used. If En_US.pc850 is not available for any reason, only ASCII characters are supported. For further details regarding keymaps, refer to the AIX Input Method Overview.

The **XmText** widget class inherits behavior and resources from the **Core** and **XmPrimitive** classes. The class pointer is **xmTextWidgetClass**. The class name is **XmText**.

Subroutines

- **XmCreateScrolledText**

- **XmCreateText**
- **XmFontListCreate**
- **XmTextClearSelection**
- **XmTextGetEditable**
- **XmTextGetMaxLength**
- **XmTextGetSelection**
- **XmTextGetString**
- **XmTextReplace**
- **XmTextSetEditable**
- **XmTextSetMaxLength**
- **XmTextSetSelection**
- **XmTextSetString**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource sets list the resources of the **XmText** widget class:

- **XmText Resource Set**
- **XmText Input Resource Set**
- **XmText Output Resource Set**
- **XmTextScrolledText Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmText** widget class:

- **XmPrimitive Resource Set**
- **Core Resource Set**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int         reason;
        XEvent      * event;
}XmAnyCallbackStruct;
```

reason Indicates why the callback was invoked.

event Points to the **XEvent** that triggered the callback.

The **Text** widget defines a new callback structure for use with verification callback routines. Note that not all of the fields are relevant for every callback reason. The client application must first look at the reason field and use only the structure members that are valid for the particular reason. The following structure is returned with **XmNlosingFocusCallbacks**, **XmNmodifyVerifyCallbacks**, and **XmNmotionVerifyCallbacks** resources.

```
typedef struct
{
        int             reason;
        XEvent          * event;
        Boolean         doit;
        XmTextPosition  currInsert, newInsert;
        XmTextPosition  startPos, endPos;
        XmTextBlock     text;
} XmTextVerifyCallbackStruct, *XmTextVerifyPtr;
```

reason Indicates why the callback routine was invoked.

event Points to the **XEvent** that invoked the callback routine.

doit Indicates whether that action that invoked the callback routine will be performed. Setting the *doit* parameter to False negates the action.

currInsert Indicates the current position of the insert cursor.

newInsert Indicates the position at which the user attempts to position the insert cursor.

startPos Indicates the starting position of the text to modify. If the callback is not a modify verification callback, this value is the same as the *currInsert* parameter.

endPos Indicates the ending position of the text to modify. If no text is replaced or deleted, then the value is the same as the *startPos* parameter. If the callback is not a modify verification callback, this value is the same as the *currInsert* parameter.

text Points to a structure of type **XmTextBlockRec**. This structure holds the text information to be inserted.

```
typedef struct
{
        char            * ptr;
        int             length;
        XmTextFormat    format;
}XmTextBlockRec, *XmTextBlock;
```

ptr Points to the text to be inserted.

length Specifies the length of the text to be inserted.

format Specifies the format of the text (such as **FMT8BIT**).

The following table describes the reasons for which the individual verification callback structure fields are valid:

Reason	Valid Fields
XmCR_LOSING_FOCUS	*reason, event, doit, currInsert, newInsert, startPos, endPos, text*
XmCR_MODIFYING_TEXT_VALUE	*reason, event, doit, currInsert, newInsert*
XmCR_MOVING_TEXT_CURSOR	*reason, event, doit, currInsert, newInsert*

Behavior

The behavior for the **Text** widget is determined by the **XmNeditMode** resource. Depending on how this resource is set, some of the key bindings perform different actions. The possible values for **XmNeditMode** are **XmSINGLE_LINE_EDIT** and **XmMULTI_LINE_EDIT**. The following describes the key bindings for these edit modes.

Default Behavior (Single–line Text Edit)

<Btn1Down>: This key binding performs the action defined in the selection array depending on the number of multiple mouse button clicks. The default selection array ordering is one click to move the insertion cursor position, two clicks to select a word, and three clicks to select a line of text, and four clicks to select a page of text.

It also begins text selection. Primary selected text that was previously selected becomes unselected.

Button1 <PtrMoved>: Text is selected in the direction of the pointer cursor movement. While the pointer cursor is moved along the text, the text is selected from the point the was pressed to the present position of the pointer cursor. Moving the pointer cursor back over previously selected text while is pressed deselects the text. Primary selected text is shown visibly by inverted text.

<Btn1Up>: The selected text becomes the primary selection (i.e., the selection is committed).

Shift <Btn1Down>: The end points of the selection move to the point where the pointer cursor is located when the shifted is pressed. If the pointer cursor is located at a position where text is already selected, the text following this position becomes unselected.

<Btn2Up>: The text is copied from the primary selection to the insertion point located at the insert cursor.

CTRL <Btn2Up>: The text is copied and cut from the primary selection and is pasted to the insertion point located at the insert cursor.

<Key> Right: The insert cursor moves one character to the right.

Shift <Key> Right: The text character to the right of the insert cursor is selected and inverted (such as primary selection). If there is already selected text to the right of the insert cursor, this text becomes unselected one character at a time.

Ctrl <Key> Right: The insert cursor moves to the end of the line.

<Key> Left: The insert cursor moves one character to the left.

Shift <Key> Left: The text character to the left of the insert cursor is selected and inverted. If there is already selected text to the left of the insert cursor, this text becomes unselected one character at a time.

Ctrl <Key> Left: The insert cursor moves to the beginning of the line.

<Key> Backspace: The character of text immediately preceding the insert cursor is deleted.

<Key> Delete or **<Key>DeleteChar** (HP keyboard): The character of text immediately following the insert cursor is deleted.

Any <Key>: This key binding inserts the associated character into the text of the **Text** widget.

<Key> Return: Calls the callback routines for **XmNactivateCallback**.

Multiline Text Edit

Button1 <PtrMoved>: Text is selected in the direction of the pointer cursor movement. While the pointer cursor is moved along the text, the text is selected from the point the was pressed to the present position of the pointer cursor. Moving the cursor over several lines selects text to the end of each line, up to the position of the pointer cursor on the current line. Moving the pointer cursor back over previously selected text while is pressed deselects the text.

<Key> Up: The insert cursor moves to the line directly above the current line on which the insert cursor resides.

<Key> Down: The insert cursor moves to the line directly below the current line on which the insert cursor resides.

<Key> Return: Inserts a new line at the point where the insert cursor is positioned. Calls the callbacks for **XmNactivateCallback**.

Default Translations

Default translations for the **Text** widget are:

Shift<Key>Tab:	**prev–tab–group()**
<Key>Tab:	**next–tab–group()**
<Key>Up:	**traverse–prev()**
<Key>Down:	**traverse–next()**
<Key>Home:	**traverse–home()**
Ctrl<Key>Right:	**end–of–line()**
Shift<Key>Right:	**key–select(right)**
<Key>Right:	**forward–character()**
Ctrl<Key>Left:	**beginning–of–line()**
Shift<Key>Left:	**key–select(left)**
<Key>Left:	**backward–character()**
Shift<Key>BackSpace:	**delete–previous–word()**
<Key>BackSpace:	**delete–previous–character()**
<Key>Return:	**activate()**
<Key>:	**self–insert()**
Shift<Btn1Down>:	**extend–start()**
<Btn1Down>:	**grab–focus()**
Button1<PtrMoved>:	**extend–adjust()**
<Btn1Up>:	**extend–end()**
Ctrl<Btn2Up>:	**move–to()**
<Btn2Up>:	**copy–to()**
<LeaveWindow>:	**leave()**
<FocusIn>:	**focusIn()**
<FocusOut>:	**focusOut()**
<Unmap>:	**unmap()**

The following default translations are for an HP keyboard:

Shift<Key>DeleteChar:	**delete–next–word()**
<Key>DeleteChar:	**delete–next–character()**

The following default translations are for a DIGITAL keyboard:

Shift<Key>Delete:	**delete–previous–word()**
<Key>Delete:	**delete–previous–character()**
Shift<Key>Linefeed:	**delete–next–word()**
<Key>Linefeed:	**delete–next–character()**
Shift<Key>F13:	**delete–next–word()**
<Key>F13:	**delete–next–character()**

The following default translations are for keyboards other than HP's or DIGITAL's :

Shift<Key>Delete:	**delete–next–word()**
<Key>Delete:	**delete–next–character()**

The following default translations override the above default translations when using Multiline Text Edit:

<Key>Tab:	**self–insert()**
<Key>Up:	**previous–line()**
<Key>Down:	**next–line()**
<Key>Home:	**beginning–of–file()**
<Key>Return:	**newline()**

When changing from Multiline Text Edit to Single–line Text Edit, the following default translations override the Multiline Text Edit default translations.

<Key>Tab:	**next–tab–group()**
<Key>Up:	**traverse–prev()**
<Key>Down:	**traverse–next()**
<Key>Home:	**traverse–home()**
<Key>Return:	**activate()**

Keyboard Traversal

Multiline Text Edit differs from standard traversal in the following manner:

Up or Down Arrow — moves the insert cursor between lines.

Tab — inserts a tab.

Home — moves the insert cursor to the first position (top) of the file.

Return — adds a new line.

Both Single–line and Multiline Text Edit differs from standard traversal in the following manner:

Right and Left Arrows — moves the insert cursor to the right or to the left.

Information on keyboard traversal is contained in the **XmPrimitive** widget class and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **Core** widget class, **XmCreateScrolledText** subroutine, **XmCreateText** subroutine, **XmFontListCreate** subroutine, **XmPrimitive** widget, **XmTextClearSelection**subroutine, **XmTextGetEditable** subroutine, **XmTextGetMaxLength** subroutine, **XmTextGetSelection** subroutine, **XmTextGetString** subroutine, **XmTextReplace** subroutine, **XmTextSetEditable** subroutine, **XmTextSetMaxLength** subroutine, **XmTextSetSelection** subroutine, **XmTextSetString** subroutine, **setlocale** subroutine.

XmToggleButton Widget Class

Purpose

The **ToggleButton** widget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleB.h>

Children

No children are supported.

Description

A **ToggleButton** widget sets non–transitory state data within an application. Usually this widget consists of an indicator (square or diamond) with either text or a pixmap to its right. However, it can also consist of just text or a pixmap without the indicator.

The toggle graphics display a **1–of–many** or **N–of–many** selection state. When a toggle indicator is displayed, a square indicator shows an **N–of–many** selection state and a diamond indicator shows a **1–of–many** selection state.

The **ToggleButton** widget implies a selected or unselected state. In the case of a label and an indicator, an empty indicator (square or diamond shaped) indicates that the **ToggleButton** widget is unselected, and a filled indicator shows that it is selected. In the case of a pixmap toggle, different pixmaps are used to display the selected/unselected states.

Normally, is used to arm and activate the button. However, if the **ToggleButton** widget resides within a **RowColumn** manager, the mouse button used is determined by the **RowColumn** resources **XmNrowColumnType** and **XmNwhichButton**.

The **XmToggleButton** widget class inherits behavior and resources from the **Core**, **XmPrimitive**, and **XmLabel** classes. The class pointer is **xmToggleButtonWidgetClass**. The class name is **XmToggleButton**.

Subroutines

- **XmCreateToggleButton**
- **XmToggleButtonGetState**
- **XmToggleButtonSetState**

New Resources

Setting the resource values for the inherited classes also sets resources for this widget. To reference a resource in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmToggleButton** class:

- **XmToggleButton Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmToggleButton** widget class:

- **XmLabel Resource Set**
- **XmPrimitive Resource Set**
- **Core Resource Set**

Keyboard Traversal

When in menu system, the following translations are added to the **ToggleButton** widget:

<Unmap>:	**Unmap()**
<FocusOut>:	**FocusOut()**
<FocusIn>:	**FocusIn()**
<Key> space:	**Noop()**
<Key> Left:	**MenuTraverseLeft()**
<Key> Right:	**MenuTraverseRight()**
<Key> Up:	**MenuTraverseUp()**
<Key> Down:	**MenuTraverseDown()**
<Key> Home:	**Noop()**

Callback Information

The following structure is returned with each callback:

```
typedef struct
{
        int             reason;
        XEvent          * event;
        Boolean         set;
} XmToggleButtonCallbackStruct;
```

reason — Is set to the value that corresponds to the type of selection that invoked this callback.

event — Points to the **XEvent** that invoked the callback.

set — Reflects the current state of the **ToggleButton** widget when the callback occurred, either **True** (selected) or **False** (unselected).

Behavior

The **ToggleButton** widget is associated with the default behavior unless it is part of a PopupMenu System, a Pulldown MenuPane in a MenuBar, or an OptionMenu System. In a menu system, the **RowColumn** widget parent determines which mouse button is used.

Default Behavior

<Btn1Down>:
(if unset): This action arms the **ToggleButton** widget. The indicator shadow is drawn so that the button looks depressed, and the indicator fills with the color specified in **XmNselectColor**. The callbacks for **XmNarmCallback** are also called.

(if set): This action arms the **ToggleButton** widget. The indicator shadow is drawn so that the button looks raised, and the indicator fills with the background color. The callbacks for **XmNarmCallback** are also called.

<Btn1Up>:
(In Button):
(if unset): This action selects the **ToggleButton** widget. Visually, it appears the same as when it is armed. The callbacks for **XmNvalueChangedCallback** are called, followed by callbacks for **XmNdisarmCallback**.
(if set): This action unselects the **ToggleButton** widget. Visually, it appears the same as when it is armed. The callbacks for **XmNvalueChangedCallback** are called, followed by callbacks for **XmNdisarmCallback**.
(Outside Of Button): If the button release occurs outside of the **ToggleButton** widget, the callbacks for **XmNdisarmCallback** are called.

<Leave Window>:
If the button is pressed and the cursor leaves the widget, it visually reverts to its previous unpressed state.

<Enter Window>:
If the button is pressed and the cursor leaves and reenters the widget, it visually appears the same as when the button was first armed.

Default PopupMenu System and OptionMenu System

<Btn2Down>:
This action disables keyboard traversal for the menu and returns the user to drag mode, which is the mode in which the menu is manipulated by using the mouse. This action also causes the **ToggleButton** widget to be armed. A shadow is drawn around the **ToggleButton** widget. The callbacks for **XmNarmCallback** are also called.

<Btn2Up>:
(if unset): This action selects the **ToggleButton** widget. The indicator shadow is drawn so that it looks depressed, and the indicator fills with the color specified in **XmNselectColor**. The menu is then unposted and the callbacks for **XmNvalueChangedCallback** are called, followed by callbacks for **XmNdisarmCallback**.
(if set): This action unselects the **ToggleButton** widget. The indicator shadow is drawn so that it looks raised, and the indicator fills with the background color. The menu is then unposted and the callbacks for **XmNvalueChangedCallback** are called, followed by callbacks for **XmNdisarmCallback**.

<Leave Window>:
Pressing and moving the cursor out of the widget window erases the shadow around the **ToggleButton** widget. This event is ignored if keyboard traversal is enabled in the menu.

<Enter Window>:
Pressing and moving the cursor into the widget window draws a shadow around the **ToggleButton** widget. This event is ignored if keyboard traversal is enabled in the menu.

Default Pulldown Menu System

<Btn1Down>:
This action disables keyboard traversal for the menu and returns the user to drag mode (the mode in which the menu is manipulated using the mouse). This action also arms the **ToggleButton** widget. A shadow is drawn around the **ToggleButton** widget. The callbacks for **XmNarmCallback** are also called.

<Btn1Up>:
(if unset): This action selects the **ToggleButton** widget. The indicator shadow is drawn so that it looks depressed, and the indicator fills with the color specified in **XmNselectColor**. The menu then unposts, and the callbacks for **XmNvalueChangedCallback** are called, followed by callbacks for **XmNdisarmCallback**.
(if set): This action unselects the **ToggleButton** widget. The indicator shadow is drawn so that it looks raised, and the indicator fills with the background color. The menu then unposts, and the callbacks for **XmNvalueChangedCallback** are called, followed by callbacks for **XmNdisarmCallback**.

<Leave Window>:
Pressing and moving the cursor out of the widget window erases the shadow around the **ToggleButton** widget. This event is ignored if keyboard traversal is enabled in the menu.

<Enter Window>:
Pressing and moving the cursor into the widget window draws a shadow around the **ToggleButton** widget. This event is ignored if keyboard traversal is enabled in the menu.

<Key>Return:
This event sets or unsets the **ToggleButton** widget if keyboard traversal is enabled in the menu.
(if set): The **ToggleButton** widget gets unset. The indicator shadow is drawn so that it looks raised, and the indicator fills with the background color.
(if unset): The **ToggleButton** widget gets set. The indicator shadow is drawn so that it looks depressed, and the indicator fills with the color specified in **XmNselectColor**.
For both set and unset cases, the menu then unposts, and the callbacks for **XmNvalueChangedCallback** are called, followed by callbacks for **XmNdisarmCallback**.

Default Translation

The default translations for **ToggleButton** widget when not in a menu system are:

<Btn1Down>:	**Arm()**
<Btn1Up>:	**Select()**
	Disarm()
<Key>Return:	**ArmAndActivate()**
<Key>space:	**ArmAndActivate()**
<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**

The default translations for **ToggleButton** widget when in a menu system are:

<BtnDown>:	**BtnDown()**
<BtnUp>:	**BtnUp()**
<EnterWindow>:	**Enter()**
<LeaveWindow>:	**Leave()**
<Key>Return:	**KeySelect()**
<Key>Escape:	**MenuShellPopdownDone()**

Keyboard Traversal

When in a menu system, the following translations are added to the **ToggleButton** widget.

<Unmap>:	**Unmap()**
<FocusOut>:	**FocusOut()**
<FocusIn>:	**FocusIn()**
<Key>Space:	**Noop()**
<Key>Left:	**MenuTraverseLeft()**
<Key>Right:	**MenuTraverseRight()**

<Key>Up:	**MenuTraverseUp()**
<Key>Down:	**MenuTraverseDown()**
<Key>Home:	**Noop()**

More information on keyboard traversal when not in a menu is contained in **XmPrimitive** and its sections on behavior and default translations.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleB.h

Related Information

The **Core** widget class, **XmCreateToggleButton** subroutine, **XmLabel** widget class, **XmPrimitive** widget class, **XmRowColumn** widget class, **XmToggleButtonGetState** subroutine, **XmToggleButtonSetState** subroutine.

XmToggleButtonGadget Gadget Class

Purpose

The **ToggleButtonGadget** gadget class.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleBG.h>

Children

No children are supported.

Description

A **ToggleButtonGadget** gadget sets non–transitory state data within a client application. Usually this widget consists of a square or diamond shaped indicator with either text or a pixmap to its right. However, it can also consist of just text or a pixmap without the indicator.

The toggle graphics display a **1–of–many** or **N–of–many** selection state. When a toggle indicator is displayed, a square shows a **N–of–many** selection state and a diamond indicator shows a **1–of–many** selection state.

The **ToggleButtonGadget** gadget implies a selected or unselected state. In the case of a label and an indicator, an empty square or diamond shaped indicator indicates that **ToggleButtonGadget** gadget is unselected, and a filled indicator shows that it is selected. Different pixmaps are used to display the selected/unselected states of a pixmap toggle.

Normally, is used to arm and activate the button. However, if the **ToggleButtonGadget** gadget resides within a menu, then mouse button use is determined by two **RowColumn** widget resources: **XmNrowColumnType** and **XmNwhichButton.**

The **XmToggleButtonGadget** gadget class inherits behavior and resources from the **Object, RectObj, XmGadget,** and **XmLabelGadget** classes. The class pointer is **xmToggleButtonGadgetClass**. The class name is **XmToggleButtonGadget.**

Subroutines

- **XmCreateToggleButtonGadget**
- **XmToggleButtonGadgetGetState**
- **XmToggleButtonGadgetSetState**

New Resources

Setting the resource values for the inherited classes also sets resources for this gadget. To reference a resource by name or by class in an **.Xdefaults** file, remove the **XmN** or **XmC** prefix and use the remaining letters. To specify one of the defined values for a resource in an **.Xdefaults** file, remove the **Xm** prefix and use the remaining letters (in either lower case or upper case, 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**). The following resource set lists the resources of the **XmToggleButtonGadget** widget class:

- **XmToggleButtonGadget Resource Set**

Inherited Resources

The following resource sets contain a complete description of the resources inherited by the **XmToggleButtonGadget** gadget class:

- **XmLabelGadget Resource Set**
- **XmGadget Resource Set**
- **XmRectObj Resource Set**

Callback Information

The following structure is returned with each callback routine:

```
typedef struct
{
        int             reason:
        XEvent          * event:
        Boolean         set;
} XmToggleButtonCallbackStruct;
```

reason Indicates why the callback routine was invoked.

event Points to the **XEvent** that triggered the callback routine.

set Reflects the **ToggleButtonGadget** widget current state when the callback occurred, either **True** (selected) or **False** (unselected).

Behavior

The **ToggleButtonGadget** gadget is associated with the default behavior unless it is part of a menu system. In a menu system, the **RowColumn** parent widget determines which mouse button is used.

Default Behavior

<Btn1Down>(if unset): This action arms the **ToggleButtonGadget** gadget. The indicator shadow is drawn so that the button looks depressed, and the indicator fills with the color specified in the **XmNselectColor** resource. The callback routines for the **XmNarmCallback** resource are also invoked.

<Btn1Down>(if set): This action arms the **ToggleButtonGadget** gadget. The indicator shadow is drawn so that the button looks raised, and the indicator fills with the background color. The callback routines for the **XmNarmCallback** resource are also invoked.

<Btn1Up>(In Button)(if unset): This action selects the **ToggleButtonGadget** gadget. Visually, it appears the same as when it is armed. The callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Btn1Up>(In Button)(if set): This action unselects the **ToggleButtonGadget** gadget. Visually, it appears the same as when it is armed. The callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Leave Window>: If the button is pressed and the cursor leaves the gadget, it visually reverts to its previous unpressed state.

<Enter Window>: If the button is pressed and the cursor leaves and reenters the gadget, it visually appears the same as when the button was first armed.

Default PopupMenu System

<Btn2Down>: This action disables keyboard traversal for the menu and returns the user to drag mode (the mode in which the menu is manipulated by using the mouse). This action also causes the **ToggleButtonGadget** gadget to be armed. A shadow is drawn around the **ToggleButtonGadget** gadget. The callback routines for the **XmNarmCallback** resource are also invoked.

<Btn2Up> (if unset): This action selects the **ToggleButtonGadget** gadget. The indicator shadow is drawn so that it looks depressed, and the indicator fills with the color specified in the **XmNselectColor** resource. The menu is then unposted and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Btn2Up> (if set): This action deselects the **ToggleButtonGadget** gadget. The indicator shadow is drawn so that it looks raised, and the indicator fills with the color. The menu is then unposted and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Leave Window>: Pressing and moving the cursor out of the widget window erases the shadow around the **ToggleButtonGadget** gadget. This event is ignored if keyboard traversal is enabled in the menu.

<Enter Window>: Pressing and moving the cursor out of the widget window draws a shadow around the **ToggleButtonGadget** gadget. This event is ignored if keyboard traversal is enabled in the menu.

<Key>Return: If keyboard traversal is enabled in the menu, this event sets or unsets the **ToggleButtonGadget** widget.

<Key>Return (if unset): The **ToggleButtonGadget** gadget gets set. The indicator shadow is drawn so that it looks depressed, and the indicator fills with the color specified in the **XmNselectColor** resource. The menu is then unposted and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Key>Return(if set): The **ToggleButtonGadget** gadget gets unset. The indicator shadow is drawn so that it looks raised, and the indicator fills with the background color. The menu is then unposted and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

Default PulldownMenu System and OptionMenu System

<Btn1Down>: This action disables keyboard traversal for the menu and returns the user to drag mode (the mode in which the menu is manipulated using the mouse). This action also arms the **ToggleButtonGadget** gadget. A shadow is drawn around the **ToggleButtonGadget** gadget. The callback routines for the **XmNarmCallback** resource are also invoked.

<Btn1Up>(if unset): This action selects the **ToggleButtonGadget** gadget. The indicator shadow is drawn so that it looks depressed, and the indicator fills with the color specified in the **XmNselectColor** resource. The menu is then unposted and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Btn1Up>(if set): This action deselects the **ToggleButtonGadget** gadget. The indicator shadow is drawn so that it looks raised, and the indicator fills with the background color. The menu then unposts, and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Leave Window>: Pressing and moving the cursor out of the widget window erases the shadow around the **ToggleButtonGadget** gadget. This event is ignored if keyboard traversal is enabled in the menu.

<Enter Window>: Pressing and moving the cursor out of the widget window draws a shadow around the **ToggleButtonGadget** gadget. This event is ignored if keyboard traversal is enabled in the menu.

<Key>Return: This event sets or unsets the **ToggleButtonGadget** gadget if keyboard traversal is enabled in the menu.

<Key>Return (if unset): The **ToggleButtonGadget** gadget gets set. The indicator shadow is drawn so that it looks depressed, and the indicator fills with the color specified in the **XmNselectColor** resource. The menu is then unposted and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

<Key>Return (if set): The **ToggleButtonGadget** gadget gets unset. The indicator shadow is drawn so that it looks raised, and the indicator fills with the background color. The menu is then unposted and the callback routines for the **XmNvalueChangedCallback** resource are invoked, followed by callback routines for the **XmNdisarmCallback** resource.

Keyboard Traversal

The description for the **XmGadget** gadget class contains information on keyboard traversal when the widget is not in a menu system. When the **ToggleButtonGadget** gadget is in a menu system, the keyboard traversal translations are defined by the **RowColumn** parent widget.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleBG.h

Related Information

The **XmCreateToggleButtonGadget** subroutine, **XmToggleButtonGadgetGetState** subroutine, **XmToggleButtonGadgetSetState** subroutine, **XmLabelGadget** gadget class, **XmGadget** gadget class, **RectObj** widget class, **Object** widget class.

AIXwindows Subroutines

XmActivateProtocol Subroutine

Purpose

A **VendorShell** subroutine that activates a protocol.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Protocols.h>

void XmActivateProtocol(Shell, Property, Protocol)
Widget Shell;
Atom Property;
Atom Protocol;
```

Description

The **XmActivateProtocol** subroutine is a **VendorShell** subroutine that activates a protocol. This subroutine updates the handlers and the *Property* parameter, but only if the *Shell* parameter is realized. It is sometimes useful to allow a protocol's state information (such as callback lists) to persist, even though the client may choose to resign temporarily from the interaction. Persistence of a protocol's state information is supported by allowing a *Protocol* parameter to be in one of two states: active or inactive. If the *Protocol* parameter is active and the *Shell* parameter is realized, the *Property* parameter contains the *Protocol* **Atom**. If the *Protocol* parameter is inactive, the **Atom** is not present in the *Property* parameter.

Parameters

Shell	Specifies the widget with which the protocol property is associated.
Property	Specifies the protocol property.
Protocol	Specifies the protocol **Atom** (or an **int** cast to **Atom**).

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Protocols.h

Related Information

The **VendorShell** widget class, **XmInternAtom** subroutine.

XmAddProtocolCallback Subroutine

Purpose

A **VendorShell** subroutine that adds client callbacks for a protocol.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <X11/Protocols.h>

void XmAddProtocolCallback(*Shell, Property, Protocol, Callback, Closure***)**

Widget *Shell*;
Atom *Property*;
Atom *Protocol*;
XtCallbackProc *Callback*;
caddr_t *Closure*;

Description

The **XmAddProtocolCallback** subroutine is a **VendorShell** subroutine that adds client callback routines for a protocol. This subroutine determines whether or not the protocol is registered; if the protocol is not registered, this subroutine calls the **XmAddProtocols** subroutine. It then adds the callback routine to the internal list. These callback routines will be called when the corresponding client message is received.

Parameters

Shell	Specifies the widget with which the protocol property is associated.
Property	Specifies the protocol property.
Protocol	Specifies the protocol **Atom** (or an **int** cast to **Atom**).
Callback	Specifies the procedure to call when a protocol message is received.
Closure	Specifies the client data to be passed to the callback when it is invoked.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Protocols.h

Related Information

The **VendorShell** widget class, **XmAddProtocols** subroutine, **XmInternAtom** subroutine.

XmAddProtocols Subroutine

Purpose

A **VendorShell** subroutine that adds the protocols to the protocol manager and allocates the internal tables.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <X11/Protocols.h>

void XmAddProtocols(*Shell*, *Property*, *Protocols*,
*NumberProtocols***)**
Widget *Shell*;
Atom *Property*;
Atom * *Protocols*;
Cardinal *NumberProtocols*;

Description

The **XmAddProtocols** subroutine is a convenience interface that adds the protocols to the protocol manager and allocates the internal tables.

Parameters

Shell	Specifies the widget with which the protocol property is associated.
Property	Specifies the protocol property.
Protocols	Specifies the protocol **Atoms** (or **ints** cast to **Atom**).
NumberProtocols	Specifies the number of elements in the *Protocols* parameter.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Protocols.h

Related Information

The **VendorShell** widget class, **XmInternAtom** subroutine.

XmAddTabGroup Subroutine

Purpose

A subroutine that adds a **Manager** widget or a **Primitive** widget to the list of tab groups.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmAddTabGroup(*TabGroup***)**
Widget *TabGroup*;

Description

The **XmAddTabGroup** subroutine adds a **Manager** widget or **Primitive** widget to the list of tab groups associated with a particular widget hierarchy. Each instance of the **List** widget, multiline **Text** edit widget, **OptionMenu** widget, and **ScrollBar** widget must be placed within its own tab groups; do not place other widgets in these groups. This allows the arrow keys to function in their normal fashion within these widgets.

When the keyboard is used to traverse through a widget hierarchy, **Primitive** widgets and **Manager** widgets are grouped together into tab groups. Any **Manager** widget or **Primitive** widget can be a tab group. Within a tab group, the focus is moved to the next widget within the tab group through use of the arrow keys. To move to another tab group, enter either the Tab, or <Shift>Tab keys.

Parameter

TabGroup Specifies the **Manager** or **Primitive** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmManager** widget class, **XmPrimitive** widget class, **XmRemoveTabGroup** subroutine.

XmAtomToName Subroutine

Purpose

Returns the string representation for an atom.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <X11/AtomMgr.h>

String XmAtomToName(*Display*, *Atom***)**
Display * *Display*;
Atom * *Atom*;

Description

The **XmAtomToName** subroutine returns the string representation for an atom. It mirrors the **Xlib** library interfaces for atom management, but provides client side caching. When and where caching is provided in **Xlib** library, the routines become pseudonyms for the **Xlib** library routines.

Parameters

Display	Specifies the connection to the X Server.
Atom	Specifies the atom for the property name you want returned.

Return Value

Returns a string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/AtomMgr.h

XmCascadeButtonHighlight Subroutine

Purpose

A **CascadeButton** and **CascadeButtonGadget** subroutine that sets the highlight state.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/CascadeB.h>
#include <Xm/CascadeBG.h>

void XmCascadeButtonHighlight (*CascadeButton*, *Highlight*)
Widget *CascadeButton*;
Boolean *Highlight*;

Description

The **XmCascadeButtonHighlight** subroutine is a **CascadeButton** and **CascadeButtonGadget** subroutine that sets the highlight state. This subroutine either draws or erases the shadow highlight around the **CascadeButton** or the **CascadeButtonGadget** subroutine.

Parameters

CascadeButton	Specifies the **CascadeButton** or the **CascadeButtonGadget** subroutine to be highlighted or unhighlighted.
Highlight	Specifies whether to highlight (**True**) or to unhighlight (**False**).

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/CascadeB.h

/usr/include/Xm/CascadeBG.h

Related Information

The **XmCascadeButton** widget class, **XmCascadeButtonGadget** widget class.

XmClipboardCancelCopy Subroutine

Purpose

A clipboard subroutine that cancels a copy to the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

void XmClipboardCancelCopy(*Display*, *Window*,
ItemIdentification)
Display * *Display*;
Window *Window*;
long *ItemIdentification*;

Description

The **XmClipboardCancelCopy** subroutine cancels the clipboard copy subroutine that is in progress and frees up temporary storage. When a copy is to be performed, the **XmClipboardStartCopy** subroutine allocates temporary storage for the clipboard data. The **XmClipboardCopy** subroutine fills in the appropriate data into the temporary storage. The **XmClipboardEndCopy** subroutine copies the data to the clipboard structure and frees up the temporary storage structures. If the **XmClipboardCancelCopy** subroutine is called, the **XmClipboardEndCopy** subroutine does not have to be called. A call to the **XmClipboardCancelCopy** subroutine is valid only after a call to the **XmClipboardStartCopy** subroutine and before a call to the **XmClipboardEndCopy** subroutine.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or **XtDisplay** subroutine.
Window	Specifies a widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each clipboard subroutine it calls.
ItemIdentification	Specifies the number assigned to this data item. This number was returned by a previous call to the **XmClipboardStartCopy** subroutine.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCopy** subroutine, **XmClipboardEndCopy** subroutine, **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardCopy Subroutine

Purpose

A clipboard subroutine that copies a data item to temporary storage for later copying to the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardCopy(*Display*, *Window*, *ItemIdentification*,
FormatName, *Buffer*, *Length*,
PrivateIdentification,
DataIdentification)

Display * *Display*;
Window *Window*;
long *ItemIdentification*;
char * *FormatName*;
char * *Buffer*;
unsigned long *Length*;
int *PrivateIdentification*;
int * *DataIdentification*;

Description

The **XmClipboardCopy** subroutine copies a data item to temporary storage. The data item is moved from temporary storage to the clipboard data structure when a call to the **XmClipboardEndCopy** subroutine is made. Additional calls to the **XmClipboardCopy** subroutine before a call to the **XmClipboardEndCopy** subroutine adds additional data item formats to the same data item or append data to an existing format. Formats are described in the ICCC manual as targets.

NOTE: Do not call the **XmClipboardCopy** subroutine before a call to the **XmClipboardStartCopy** subroutine has been made. The latter subroutine allocates temporary storage required by the **XmClipboardCopy** subroutine.

If the *Buffer* parameter is **NULL**, the data is considered to be passed by name. When data that has been passed by name is later requested by another application, the application that owns the data receives a callback with a request for the data. The application that owns the data must then transfer the data to the clipboard with the **XmClipboardCopyByName** subroutine. When a data item that was passed by name is deleted from the clipboard, the application that owns the data receives a callback stating that the data is no longer needed.

For information on the callback routine, see the callback parameter description for the **XmClipboardStartCopy** subroutine.

Parameters

Display — Specifies a pointer to the **Display** structure that was returned in a previous call to the **XOpenDisplay** subroutine or **XtDisplay** subroutine.

Window	Specifies the widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each clipboard subroutine it calls.
ItemIdentification	Specifies the number assigned to this data item. The number was returned by a previous call to the **XmClipboardStartCopy** subroutine.
FormatName	Specifies the name of the format in which the data item is stored on the clipboard. Format is known as target in the ICCC manual.
Buffer	Specifies the buffer from which the clipboard copies the data.
Length	Specifies the length of the data being copied to the clipboard.
PrivateIdentification	Specifies the private data that the application wants to store with the data item.
DataIdentification	Specifies an identifying number assigned to the data item that uniquely identifies the data item and the format. This parameter is required only for data that is passed by name.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or wants to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCopyByName** subroutine, **XmClipboardEndCopy** subroutine, **XmClipboardStartCopy** subroutine.

XmClipboardCopyByName Subroutine

Purpose

A clipboard subroutine that copies a data item passed by name.

Library

AIXwindows Library (**libXm.a**)

Syntax

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

int XmClipboardCopyByName(Display, Window,
                          DataIdentification, Buffer,
                          Length, PrivateIdentification)
Display * Display;
Window Window;
int DataIdentification;
char * Buffer;
unsigned long Length;
int PrivateIdentification;
```

Description

The **XmClipboardCopyByName** subroutine copies the actual data for a data item that was previously passed by name to the clipboard. Data is considered to be passed by name when a call to the **XmClipboardCopy** subroutine is made with a null buffer parameter. Additional calls to this subroutine append new data to the existing data. When making additional calls to this subroutine, the clipboard should be locked to ensure the integrity of the clipboard data. To lock the clipboard, use the **XmClipboardLock** subroutine. Unlock the clipboard when the copying is completed; to unlock the clipboard, use the **XmClipboardUnlock** subroutine.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies the widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each of the clipboard subroutines that it calls.
DataIdentification	Specifies an identifying number assigned to the data item that uniquely identifies the data item and the format. This number was assigned by the **XmClipboardCopy** subroutine to the data item.
Buffer	Specifies the buffer from which the clipboard copies the data.
Length	Specifies the number of bytes in the data item.

PrivateIdentification Specifies the private data that the application wants to store with the data item.

Return Values

ClipboardSuccess The subroutine is successful.

ClipboardLocked The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCopy** subroutine, **XmClipboardLock** subroutine, **XmClipboardStartCopy** subroutine, **XmClipboardUnlock** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardEndCopy Subroutine

Purpose

A clipboard subroutine that ends a copy to the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardEndCopy(*Display*, *Window*,
ItemIdentification)
Display * *Display*;
Window *Window*;
long *ItemIdentification*;

Description

The **XmClipboardEndCopy** subroutine locks the clipboard from access by other applications, places data in the clipboard data structure, and unlocks the clipboard. Data items copied to the clipboard by the **XmClipboardCopy** subroutine are not actually entered in the clipboard data structure until the call to the **XmClipboardEndCopy** subroutine.

This subroutine also frees up temporary storage that was allocated by the **XmClipboardStartCopy** subroutine, which must be called before the **XmClipboardEndCopy** subroutine. The latter subroutine should not be called if the **XmClipboardCancelCopy** subroutine has been called.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or **XtDisplay** subroutine.
Window	Specifies the widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each clipboard subroutine it calls.
ItemIdentification	Specifies the number assigned to this data item. This number was returned by a previous call to the **XmClipboardStartCopy** subroutine.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or wants to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCancelCopy** subroutine, **XmClipboardCopy** subroutine, **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardEndRetrieve Subroutine

Purpose

A clipboard subroutine that ends a copy from the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardEndRetrieve(*Display*, *Window***)**
Display * *Display*;
Window *Window*;

Description

The **XmClipboardEndRetrieve** subroutine suspends the incremental copying of data from the clipboard. This subroutine notifies the clipboard routines that the application has finished copying an item to the clipboard. Until this subroutine is called, data items can be retrieved incrementally from the clipboard by calling the **XmClipboardRetrieve** subroutine. If the application calls the **XmClipboardStartRetrieve** subroutine, it must call the **XmClipboardEndRetrieve** subroutine. If data is not being copied incrementally, the **XmClipboardStartRetrieve** subroutine and the **XmClipboardEndRetrieve** subroutine do not need to be called.

Parameters

Display Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.

Window Specifies the widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each clipboard subroutines it calls.

Return Values

ClipboardSuccess The subroutine is successful.

ClipboardLocked The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or wants to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardRetrieve** subroutine, **XmClipboardStartCopy** subroutine, **XmClipboardStartRetrieve** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardInquireCount Subroutine

Purpose

A clipboard subroutine that returns the number of data item formats.

Library

AIXwindows Library (**libXm.a**)

Syntax

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

int XmClipboardInquireCount(Display, Window, Count,
                            MaximumFormatNameLength)
Display * Display;
Window Window;
int * Count;
int * MaximumFormatNameLength
```

Description

The **XmClipboardInquireCount** subroutine returns the number of data item formats available for the data item in the clipboard. This subroutine also returns the maximum name length for all formats in which the data item is stored.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies a widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using **XtWindow**. The same application instance should pass the same window ID to each clipboard subroutine it calls.
Count	Returns the number of data item formats available for the data item in the clipboard. If no formats are available, this parameter equals zero. The count includes the format that was passed by name.
MaximumFormatNameLength	Specifies the maximum length of all format names for the data item in the clipboard.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ClipboardNoData	The subroutine could not find data on the clipboard corresponding to the format requested. This can occur because the clipboard is empty; there is data on the clipboard but not in the requested format; or the data in the requested format was passed by name and is no longer available.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardInquireFormat Subroutine

Purpose

A clipboard subroutine that returns a specified format name.

Library

AIXwindows Library (**libXm.a**)

Syntax

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

int XmClipboardInquireFormat(Display, Window, Index,
                             FormatNameBuffer,
                             BufferLength, CopiedLength)
Display * Display;
Window Window;
int Index;
char * FormatNameBuffer;
unsigned long BufferLength;
unsigned long * CopiedLength;
```

Description

The **XmClipboardInquireFormat** subroutine returns a specified format name for the next paste item in the clipboard. If the name must be truncated, the subroutine returns a warning status.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or **XtDisplay** subroutine.
Window	Specifies a widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each clipboard subroutine it calls.
Index	Specifies which of the ordered format names to obtain. If this index ***i*** is greater than the number of formats for the data item, this subroutine returns a zero in the *CopiedLength* parameter.
FormatNameBuffer	Specifies the buffer that receives the format name.
BufferLength	Specifies the number of bytes in the format name buffer.
CopiedLength	Specifies the number of bytes in the string copied to the buffer. If this parameter equals zero, there is no *Nth* format for the next data item.

Return Values

ClipboardSuccess	The subroutine is successful.

ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.
ClipboardTruncate	The data returned is truncated because the user did not provide a buffer that was large enough to hold the data.
ClipboardNoData	The subroutine could not find data on the clipboard corresponding to the format requested. This could occur because the clipboard is empty; there is data on the clipboard but not in the requested format; or the data in the requested format was passed by name and is no longer available.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardInquireLength Subroutine

Purpose

A clipboard subroutine that returns the length of the stored data.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardInquireLength(*Display, Window, FormatName, Length***)**
Display * *Display*;
Window *Window*;
char * *FormatName*;
unsigned long * *Length*;

Description

The **XmClipboardInquireLength** subroutine returns the length of the data stored under a specified format name for the next paste clipboard data item. If no data is found for the specified format, or if there is no item on the clipboard, this subroutine returns a value of zero.

Any format passed by name is assumed to have the *Length* parameter passed in a call to the **XmClipboardCopy** subroutine, even though the data has not yet been transferred to the clipboard in that format.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies a widget window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each of the clipboard subroutines it calls.
FormatName	Specifies the name of the format for the next paste item.
Length	Specifies the length of the next data item in the specified format. This parameter equals zero if no data is found for the specified format, or if there is no item on the clipboard.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to

	ask if the user wants to keep trying or to give up on the operation.
ClipboardNoData	The subroutine could not find data on the clipboard corresponding to the format requested. This could occur because the clipboard is empty, there is data on the clipboard but not in the requested format, or the data in the requested format was passed by name and is no longer available.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCopy** subroutine, **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardInquirePendingItems Subroutine

Purpose

A clipboard subroutine that returns a list of DataIdentification/PrivateIdentification pairs.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardInquirePendingItems(*Display*, *Window*,
FormatName,
ItemList, *Count*)

Display * *Display*;
Window *Window*;
char * *FormatName*;
XmClipboardPendingList * *ItemList*;
unsigned long * *Count*;

Description

The **XmClipboardInquirePendingItems** subroutine returns a list of DataIdentification/PrivateIdentification pairs for the specified format name. A data item is considered pending if the application originally passed it by name, the application has not yet copied the data, and the item has not yet been deleted from the clipboard. The application is responsible for freeing the memory this subroutine originally provided to store the list.

This subroutine is used by an application when it exits to determine if the data that is passed by name should be sent to the clipboard.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies the window identification that relates the application window to the clipboard. The same application instance should pass the same window identification to each of the clipboard subroutine it calls.
FormatName	Specifies a string that contains the name of the format for which the list of data–identification/private–identification pairs is to be obtained.
ItemList	Specifies the address of the array of data identification/private identification pairs for the specified format name. This parameter is an **XmClipboardPendingList** subroutine. The application is responsible for freeing the memory provided by this subroutine for storing the list.
Count	Specifies the number of items returned in a list. If there is no data for the specified format name, or if there is no item on the clipboard, this parameter equals zero.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine.

XmClipboardLock Subroutine

Purpose

A clipboard subroutine that locks the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardLock(*Display*, *Window***)**
Display * *Display*;
Window *Window*;

Description

The **XmClipboardLock** subroutine locks the clipboard from access by another application until the **XmClipboardUnlock** subroutine is called. All clipboard subroutines lock and unlock the clipboard to prevent simultaneous access. This subroutine allows the application to keep the clipboard data from changing between calls to the **XmClipboardInquireCount** subroutine and other clipboard subroutines. The application does not need to lock the clipboard between calls to the **XmClipboardStartCopy** subroutine and the **XmClipboardEndCopy** subroutine or to the **XmClipboardStartRetrieve** subroutine and the **XmClipboardEndRetrieve** subroutine.

The application should lock the clipboard before multiple calls to the **XmClipboardCopyByName** subroutine and should unlock the clipboard after completion.

If the clipboard is already locked by another application, the **XmClipboardLock** subroutine returns an error status. Multiple calls to this subroutine by the same application increases the lock level.

Parameters

Display Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.

Window Specifies the window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each of the clipboard subroutines that it calls.

Return Values

ClipboardSuccess The subroutine is successful.

ClipboardLocked The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCopyByName** subroutine, **XmClipboardEndCopy** subroutine, **XmClipboardEndRetrieve** subroutine, **XmClipboardInquireCount** subroutine, **XmClipboardStartCopy** subroutine, **XmClipboardStartRetrieve** subroutine, **XmClipboardUnlock** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardRegisterFormat Subroutine

Purpose

A clipboard subroutine that registers a new format.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardRegisterFormat(*Display*, *FormatName*, *FormatLength***)**
Display * *Display*;
char * *FormatName*;
unsigned long *FormatLength*;

Description

The **XmClipboardRegisterFormat** subroutine registers a new format. Each format stored on the clipboard should have a length associated with it; this length must be known to the clipboard routines. Formats are known as targets in the Inter–Client Communication Conventions Manual (ICCCM) . All of the formats specified by the ICCCM conventions are preregistered. Any other format that the application wants to use must either be 8–bit data or be registered through this routine. Failure to register the length of the data results in incompatible applications across platforms having different byte swapping orders.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to **XtOpenDisplay** or **XtDisplay**.
FormatName	Specifies the string name for the new format (target).
FormatLength	Specifies the format length in bits (8, 16, 32).

Return Values

ClipboardBadFormat	The *FormatName* parameter must not be **NULL**, and the *FormatLength* parameter must be 8, 16, or 32.
ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine was unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or wants to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine.

XmClipboardRetrieve Subroutine

Purpose

A clipboard subroutine that retrieves a data item from the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

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

int XmClipboardRetrieve(Display, Window, FormatName,
                        Buffer, Length, NumberBytes,
                        PrivateIdentification)
Display * Display;
Window Window;
char * FormatName;
char * Buffer;
unsigned long Length;
unsigned long * NumberBytes;
int * PrivateIdentification;
```

Description

The **XmClipboardRetrieve** subroutine retrieves the current data item from clipboard storage. It returns a warning if the clipboard is locked, if there is no data on the clipboard, or if the data needs to be truncated because the buffer length is too short.

Between a call to the **XmClipboardStartRetrieve** subroutine and the **XmClipboardEndRetrieve** subroutine, multiple calls to the **XmClipboardRetrieve** subroutine with the same format name results in data being incrementally copied from the clipboard until the data in that format has all been copied.

The value **ClipboardTruncate,** returned from calls to the **XmClipboardRetrieve** subroutine, indicates that more data remains to be copied in the given format. All calls to the **XmClipboardInquire** subroutines that the application needs to make to effect the copy from the clipboard should be made between the call to the **XmClipboardStartRetrieve** subroutine and the first call to the **XmClipboardRetrive** subroutine. That way, the application does not need to call the **XmClipboardLock** subroutine and the **XmClipboardUnlock** subroutine. Applications do not need to use the **XmClipboardStartRetrieve** subroutine and the **XmClipboardEndRetrieve** subroutine, in which case the **XmClipboardRetrieve** subroutine works as it did before.

Parameters

Display — Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.

Window — Specifies the window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each of the clipboard subroutines that it calls.

FormatName	Specifies the name of a format in which the data is stored on the clipboard.
Buffer	Specifies the buffer to which the application wants the clipboard to copy the data.
Length	Specifies the length of the application buffer.
NumberBytes	Specifies the number of bytes of data copied into the application buffer.
PrivateIdentification	Specifies the private data stored with the data item by the application that placed the data item on the clipboard. If the application did not store private data with the data item, this parameter returns zero.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until it goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.
ClipboardTruncate	The data returned is truncated because the user did not provide a buffer that was large enough to hold the data.
ClipboardNoData	The subroutine could not find data on the clipboard corresponding to the format requested. This could occur because the clipboard is empty, there is data on the clipboard but not in the requested format, or the data in the requested format was passed by name and is no longer available.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardEndRetrieve** subroutine, **XmClipboardLock** subroutine, **XmClipboardStartCopy** subroutine, **XmClipboardStartRetrieve** subroutine, **XmClipboardUnlock** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardStartCopy Subroutine

Purpose

A clipboard subroutine that sets up a storage and data structure.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardStartCopy(*Display*, *Window*, *ClipboardLabel*,
TimeStamp, *Widget*, *Callback*,
ItemIdentification)
Display * *Display*;
Window *Window*;
XmString *ClipboardLabel*;
Time *Timestamp*;
Widget *Widget*;
VoidProc *Callback*;
long * *ItemIdentification*;

Description

The **XmClipboardStartCopy** subroutine sets up storage and data structures to receive clipboard data. An application calls this subroutine during a cut or copy operation. The data item that these structures receive then becomes the next paste item in the clipboard.

Copying a large piece of data to the clipboard can take time. Since it is possible that, once copied, the data will never be requested by an application, the **AIXwindows Toolkit** provides a means of delaying the passing of clipboard data until the data has been requested.

Instead of passing clipboard data initially, the application passes format and length information to the **XmClipboardCopy** subroutine, along with a subroutine ID and a callback routine address that is passed in the **XmClipboardStartCopy** subroutine. The subroutine ID is needed for communications between the clipboard subroutines in the application that owns the data and the clipboard subroutines in the application that requests the data.

The callback routines are responsible for copying the actual data to the clipboard through the **XmClipboardCopyByName** subroutine. The callback routine is also called if the data item is removed from the clipboard and the actual data is no longer needed.

The *Widget* and *Callback* parameters must be present in order to pass data by name. The callback format is as follows:

subroutine name
Widget *Widget*;
int * *DataIdentification*;
int * *Private*;
int * *Reason*;

Widget Specifies the ID of the widget passed to this subroutine.

DataIdentification — Specifies the identifying number returned by the **XmClipboardCopy** subroutine, which identifies the pass–by–name data.

Private — Specifies the private information passed to the **XmClipboardCopy** subroutine.

Reason — Specifies the reason, which is either **XmCR_CLIPBOARD_DATA_DELETE** or **XmCR_CLIPBOARD_DATA_REQUEST**.

Parameters

Display — Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.

Window — Specifies the window ID that relates the application window to the clipboard. The subroutine window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each of the clipboard subroutines that it calls.

ClipboardLabel — Specifies the label to be associated with the data item. This parameter is used to identify the data item, for example, in a clipboard viewer. An example of a label is the name of the application that places the data in the clipboard.

TimeStamp — Specifies the time of the event that triggered the copy.

Widget — Specifies the ID of the widget that receives messages requesting data previously passed by name. This parameter must be present in order to pass data by name. Any valid subroutine ID in your application can be used for this purpose and all the message handling is taken care of by the cut and paste subroutines.

Callback — Specifies the address of the callback routine that is called when the clipboard needs data that was originally passed by name. This is also the callback to receive the **delete** message for items that were originally passed by name. This parameter must be present in order to pass data by name.

ItemIdentification — Specifies the number assigned to this data item. The application uses this number in calls to the **XmClipboardCopy** subroutine, the **XmClipboardEndCopy** subroutine, and the **XmClipboardCancelCopy** subroutine.

Return Values

ClipboardSuccess — The subroutine is successful.

ClipboardLocked — The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCancelCopy** subroutine, **XmClipboardCopy** subroutine, **XmClipboardCopyByName** subroutine, **XmClipboardEndCopy** subroutine, **XmClipboardEndRetrieve** subroutine, **XmClipboardInquireCount** subroutine, **XmClipboardInquireFormat** subroutine, **XmClipboardInquireLength** subroutine, **XmClipboardInquirePendingItems** subroutine, **XmClipboardLock** subroutine, **XmClipboardRegisterFormat** subroutine, **XmClipboardRetrieve** subroutine, **XmClipboardStartRetrieve** subroutine, **XmClipboardUndoCopy** subroutine, **XmClipboardUnlock** subroutine, **XmClipboardWithdrawFormat** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardStartRetrieve Subroutine

Purpose

A subroutine that starts a copy from the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

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

int XmClipboardStartRetrieve(Display, Window, Timestamp)
Display * Display;
Window Window;
Time Timestamp;
```

Description

The **XmClipboardStartRetrieve** subroutine tells the clipboard routines that the application is ready to start copying an item from the clipboard. The clipboard is locked by this subroutine, and stays locked until the **XmClipboardRetrieve** subroutine is called. Between a call to the **XmClipboardStartRetrieve** subroutine and a call to the **XmClipboardEndRetrieve** subroutine, multiple calls to the **XmClipboardRetrieve** subroutine with the same format name resulst in data being incrementally copied from the clipboard until the data in that format has all been copied.

The value **ClipboardTruncate**, returned by calls to the **XmClipboardRetrieve** subroutine, indicates that more data remains to be copied in the given format. Any calls to the **XmClipboardInquire** subroutines that the application must make to effect the copy from the clipboard should be made between the call to the **XmClipboardStartRetrieve** subroutine and the first call to the **XmClipboardRetrieve** subroutine. That way, the application does not need to call the **XmClipboardLock** subroutine and the **XmClipboardUnlock** subroutine. Applications do not need to use the **XmClipboardStartRetrieve** subroutine and the **XmClipboardEndRetrieve** subroutine, in which case **XmClipboardRetrieve** subroutine works as it did before.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies the window ID that relates the application window to the clipboard. The subroutine window ID can be obtained by using the **XtWindow** subroutine.The same application instance should pass the same window ID to each of the clipboard subroutines that it calls.
Timestamp	Specifies the time of the event that triggered the copy.

Return Value

ClipboardSuccess	The subroutine is successful.

ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardEndRetrieve** subroutine, **XmClipboardInquireCount** subroutine, **XmClipboardInquireFormat** subroutine, **XmClipboardInquireLength** subroutine, **XmClipboardInquirePendingItems** subroutine, **XmClipboardLock** subroutine, **XmClipboardRetrieve** subroutine, **XmClipboardStartCopy** subroutine, **XmClipboardUnlock** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardUndoCopy Subroutine

Purpose

A clipboard subroutine that deletes the last item placed on the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardUndoCopy(*Display*, *Window***)**
Display * *Display*;
Window *Window*;

Description

The **XmClipboardUndoCopy** subroutine deletes the last item placed on the clipboard if the item was placed there by an application with the passed *Display* and *Window* parameters. Any data item deleted from the clipboard by the original call to the **XmClipboardCopy** subroutine is restored. If the *Display* or *Window* parameter IDs do not match the last copied item, no action is taken, and this subroutine has no effect.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies the window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same instance should pass the same window ID to each of the clipboard subroutines it calls.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCopy** subroutine, **XmClipboardLock** subroutine, **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardUnlock Subroutine

Purpose

A clipboard subroutine that unlocks the clipboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

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

int XmClipboardUnlock (Display, Window, RemoveAllLocks)
Display * Display;
Window Window;
Boolean RemoveAllLocks
```

Description

The **XmClipboardUnlock** subroutine unlocks the clipboard, enabling it to be accessed by other applications.

If multiple calls to the **XmClipboardLock** subroutine have occurred, the same number of calls must be made to the **XmClipboardUnlock** subroutine to unlock the clipboard, unless the *RemoveAllLocks* parameter is set to **True**.

The application should lock the clipboard before multiple calls to the **XmClipboardCopyByName** subroutine and should unlock the clipboard after completion.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies the window ID that relates the application window to the clipboard. The widget window can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each of the clipboard subroutines it calls.
RemoveAllLocks	Indicates that all nested locks should be removed, when **True**. When **False**, indicates that only one level of lock should be removed.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCancelCopy** subroutine, **XmClipboardCopy** subroutine, **XmClipboardCopyByName** subroutine, **XmClipboardEndCopy** subroutine, **XmClipboardEndRetrieve** subroutine, **XmClipboardInquireCount** subroutine, **XmClipboardInquireFormat** subroutine, **XmClipboardInquireLength** subroutine, **XmClipboardInquirePendingItems** subroutine, **XmClipboardLock** subroutine, **XmClipboardRegisterFormat** subroutine, **XmClipboardRetrieve** subroutine, **XmClipboardStartCopy** subroutine, **XmClipboardStartRetrieve** subroutine, **XmClipboardUndoCopy** subroutine, **XmClipboardWithdrawFormat** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmClipboardWithdrawFormat Subroutine

Purpose

A clipboard subroutine that indicates that the application no longer wants to supply a data item.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <Xm/CutPaste.h>

int XmClipboardWithdrawFormat(*Display*, *Window*,
DataIdentification)
Display * *Display*;
Window *Window*;
int *DataIdentification*;

Description

The **XmClipboardWithdrawFormat** subroutine indicates that the application no longer supplies to the clipboard a data item that the application had previously passed by name.

Parameters

Display	Specifies a pointer to the **Display** structure that was returned in a previous call to the **XtOpenDisplay** subroutine or the **XtDisplay** subroutine.
Window	Specifies the window ID that relates the application window to the clipboard. The widget window ID can be obtained by using the **XtWindow** subroutine. The same application instance should pass the same window ID to each of the clipboard subroutine that it calls.
DataIdentification	Specifies an identifying number assigned to the data item that uniquely identifies the data item and the format. This was assigned to the item when it was originally passed by the **XmClipboardCopy** subroutine.

Return Values

ClipboardSuccess	The subroutine is successful.
ClipboardLocked	The subroutine is unsuccessful because the clipboard was locked by another application. The application can continue to call the subroutine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/Xm/CutPaste.h

Related Information

The **XmClipboardCopy** subroutine, **XmClipboardStartCopy** subroutine, **XtOpenDisplay** subroutine, **XtDisplay** subroutine, **XtWindow** subroutine.

XmCommandAppendValue Subroutine

Purpose

A **Command** subroutine that appends the passed **XmString** to the end of the string displayed in the command area of the widget.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Command.h>

void XmCreateAppendValue(*Widget, Command***)**
Widget *Widget;*
XmString *Command;*

Description

The **XmCommandAppendValue** subroutine appends the passed **XmString** type to the end of the string displayed in the command area of the **Command** widget.

Parameters

Widget	Specifies the **Command** widget ID.
Command	Specifies the passed **XmString** type.

File

/usr/include/Xm/Command.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmCommand** widget class.

XmCommandError Subroutine

Purpose

A **Command** subroutine that displays an error message.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Command.h>

void XmCommandError(*Widget, Error***)**
Widget *Widget*;
XmString *Error*;

Description

The **XmCommandError** subroutine displays an error message in the history area of a **Command** widget. The **XmString** error is displayed until the next command entered occurs.

Parameters

Widget	Specifies the **Command** widget ID.
Error	Specifies the passed **XmString** type.

File

/usr/include/Xm/Command.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmCommand** widget class.

XmCommandGetChild Subroutine

Purpose

A **Command** subroutine that is used to access a component.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Command.h>

Widget XmCommandGetChild(*Widget, Child***)**
Widget *Widget*;
unsigned char *Child*;

Description

The **XmCommandGetChild** subroutine is used to access a component within a **Command** subroutine. The parameters given to the subroutine are the **Command** widget and a value indicating which child to access.

Parameters

Widget	Specifies the **Command** widget ID.
Child	Specifies the component within the **Command** widget. The following are legal values for this parameter: **XmDIALOG_COMMAND_TEXT** **XmDIALOG_PROMPT_LABEL** **XmDIALOG_HISTORY_LIST**.

Return Value

Returns the widget ID of the specified **Command** child.

File

/usr/include/Xm/Command.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmCommand** widget class.

XmCommandSetValue Subroutine

Purpose

A **Command** subroutine that replaces a displayed string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Command.h>

void XmCommandSetValue(*Widget, Command***)**
Widget *Widget*;
XmString *Command*;

Description

The **XmCommandSetValue** subroutine is a **Command** subroutine that replaces the string that is displayed in the command area of the **Command** widget with the passed **XmString** type.

Parameters

Widget	Specifies the **Command** widget ID.
Command	Specifies the passes **XmString** type.

File

/usr/include/Xm/Command.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmCommand** widget class.

XmConvertUnits Subroutine

Purpose

Converts a value in one unit type to another unit type.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

int XmConvertUnits(*Widget, Orientation, FromUnitType,*
*FromValue, ToUnitType***)**
Widget *Widget*
int *Orientation*;
int *FromUnitType*;
int *FromValue*;
int *ToUnitType*;

Description

The **XmConvertUnits** subroutine converts the value and returns it as the return value from the subroutine.

The *FromUnitType* and *ToUnitType* parameters can have the following values:

- **XmPIXELS**–all values provided to the widget are treated as normal pixel values. This is the default for parameter.
- **Xm100TH_MILLIMETERS**–all values provided to the widget are treated as 1/100 millimeter.
- **Xm1000TH_INCHES**–all values provided to the widget are treated as 1/1000 inch.
- **Xm100TH_POINTS**–all values provided to the widget are treated as 1/100 point. A point is a unit typically used in text processing applications and is defined as 1/72 inch.
- **Xm100TH_FONT_UNITS**–all values provided to the widget are treated as 1/100–font unit. The value to be used for the font unit is determined in one of two ways. The resource **XmNfont** can be used in a defaults file or on the command line. The standard command line options of **–fn** and **–font** can also be used. The font unit value is taken as the **QUAD_WIDTH** property of the font. The **XmSetFontUnits** subroutine allows applications to specify the font unit values.

Parameters

Widget	Specifies the widget for which the data is to be converted.
Orientation	Specifies whether the converter uses the horizontal or vertical screen resolution when performing the conversions and can have values of **XmHORIZONTAL** or **XmVERTICAL**.
FromUnitType	Specifies the current unit type of the supplied value.
FromValue	Specifies the value to be converted.
ToUnitType	Converts the value to the unit type specified.

Return Value

Returns the converted value.

Error

If a **NULL** widget, incorrect orientation, or incorrect *UnitType* is supplied as parameter data, zero is returned.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmSetFontUnit** subroutine.

XmCreateArrowButton Subroutine

Purpose

The **ArrowButton** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ArrowB.h>

Widget XmCreateArrowButton(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateArrowButton** subroutine creates an instance of an **ArrowButton** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **ArrowButton** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/ArrowB.h

Related Information

The **XmArrowButton** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateArrowButtonGadget Subroutine

Purpose

The **ArrowButtonGadget** creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ArrowB.h>

Widget XmCreateArrowButtonGadget (*Parent, Name,*
ArgumentList,
ArgumentCount)

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateArrowButtonGadget** subroutine creates an instance of an **XmArrowButtonGadget** gadget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **ArrowButtonGadget** widget ID.

File

/usr/include/Xm/ArrowB.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmArrowButtonGadget** gadget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateBulletinBoard Subroutine

Purpose

A **BulletinBoard** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/BulletinB.h>

Widget XmCreateBulletinBoard(*Parent*, *Name*, *ArgumentList*,
ArgumentCount)
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*:

Description

The **XmCreateBulletinBoard** subroutine creates an instance of a **BulletinBoard** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
Argument Count	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **BulletinBoard** widget ID.

File

/usr/include/Xm/BulletinB.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmBulletinBoard** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateBulletinBoardDialog Subroutine

Purpose

The **BulletinBoard Dialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/BulletinB.h>

Widget XmCreateBulletinBoardDialog(*Parent, Name,*
ArgumentList,
*ArgumentCount***)**

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateBulletinBoardDialog** subroutine is a convenience creation subroutine that creates a **DialogShell** widget and an unmanaged **BulletinBoard** child of the **DialogShell** widget. A **BulletinBoardDialog** widget is used for interactions not supported by the standard dialog set. This subroutine does not automatically create any labels, buttons, or other dialog components. Such components should be added by the application after the **BulletinBoardDialog** widget is created.

Use the **XtManageChild** subroutine to pop up the **BulletinBoardDialog** widget (passing the **BulletinBoard** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value in the *ArgumentList* parameter.

Return Value

Returns the **BulletinBoard** widget ID.

File

/usr/include/Xm/BulletinB.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmBulletinBoard** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateCascadeButton Subroutine

Purpose

The **CascadeButton** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/CascadeB.h>

Widget XmCreateCascadeButton (*Parent, Name, ArgumentList, ArgumentCount*)
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateCascadeButton** subroutine creates an instance of a **CascadeButton** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID. The parent must be a **RowColumn** widget.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **CascadeButton** widget ID.

File

/usr/include/Xm/CascadeB.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmCascadeButton** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateCascadeButtonGadget Subroutine

Purpose

The **CascadeButtonGadget** creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/CascadeBG.h>

Widget XmCreateCascadeButtonGadget(*Parent, Name,*
*ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateCascadeButtonGadget** subroutine creates an instance of an **XmCascadeButtonGadget** gadget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID. The parent must be a **RowColumn** widget.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **XmCascadeButtonGadget** subroutine ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/CascadeBG.h

Related Information

The **XmCascadeButtonGadget** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateCommand Subroutine

Purpose

A **Command** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Command.h>

Widget XmCreateCommand(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent;*
String *Name;*
ArgList *ArgumentList*;
Cardinal *ArgumentCount*

Description

The **XmCreateCommand** subroutine creates an instance of a **Command** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget.
Name	Specifies a name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **Command** widget ID.

File

/usr/include/Xm/Command.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmCommand** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateDialogShell Subroutine

Purpose

The **DialogShell** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/DialogS.h>

Widget XmCreateDialogShell(*Parent, Name,*
ArgumentList,
ArgumentCount)

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateDialogShell** subroutine creates an instance of a **DialogShell** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **DialogShell** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/DialogS.h

Related Information

The **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateDrawingArea Subroutine

Purpose

The **DrawingArea** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/DrawingA.h>

Widget XmCreateDrawingArea(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateDrawingArea** subroutine creates an instance of a **DrawingArea** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget for the widget to be created.
Name	Specifies a name for the created widget. This name is used for retrieving resources; it should not be the same as the name of any other widget that is a child of the same parent widget, unless identical resource values are used for the child widgets.
ArgumentList	Specifies the argument list used to override the resource defaults.
ArgumentCount	Specifies the number of arguments in the *ArgumentList* parameter.

Return Value

Returns the **DrawingArea** widget ID.

File

/usr/include/Xm/DrawingA.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmDrawingArea** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateDrawnButton Subroutine

Purpose

The **DrawnButton** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/DrawnB.h>

Widget XmCreateDrawnButton(*Parent*, *Name*, *ArgumentList*, *ArgumentCount*)
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateDrawnButton** subroutine creates an instance of a **DrawnButton** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget for the widget to be created.
Name	Specifies a name for the created widget. This name is used for retrieving resources, and therefore it should not be the same as the name of any other widget that is a child of the same parent widget, unless identical resource values are used for the child widgets.
ArgumentList	Specifies the argument list used to override the resource defaults.
ArgumentCount	Specifies the number of arguments in the *ArgumentList* parameter.

Return Value

Returns the **DrawnButton** widget ID.

File

/usr/include/Xm/DrawnB.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmDrawnButton** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateErrorDialog Subroutine

Purpose

The **MessageBox ErrorDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h>

Widget XmCreateErrorDialog(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateErrorDialog** subroutine is a convenience creation subroutine that creates a **DialogShell** widget and an unmanaged **MessageBox** child of the **DialogShell** widget. A **ErrorDialog** widget warns the user of an invalid or potentially dangerous condition. It includes a symbol, a message, and three buttons. The default symbol is an octagon with a diagonal slash. The default button labels are **OK**, **Cancel**, and **Help**.

Use the **XtManageChild** subroutine to pop up the **ErrorDialog** widget (passing the **MessageBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **MessageBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MessageB.h

Related Information

The **XmMessageBox** widget class, **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateFileSelectionBox Subroutine

Purpose

The **FileSelectionBox** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/FileSB.h>

Widget XmCreateFileSelectionDialog(*Parent*, *Name*,
ArgumentList,
ArgumentCount)

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateFileSelectionBox** subroutine creates an unmanaged **FileSelectionBox** widget. A **FileSelectionBox** widget is used to select a file. It includes the following:

- An editable text field for the directory mask.
- A scrolling list of file names.
- An editable text field for the selected file.
- Labels for the list and text fields.
- Four buttons.

The default button labels are: **OK**, **Filter**, **Cancel**, and **Help**. One additional **WorkArea** can be added to the **FileSelectionBox** widget after creation.

If the parent of the **FileSelectionBox** widget is a **DialogShell** widget, use the **XtManageChild** subroutine to pop up the **FileSelectionBox** widget (passing the **FileSelectionBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **FileSelectionBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/FileSB.h

Related Information

The **XmFileSelectionBox** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateFileSelectionDialog Subroutine

Purpose

The **FileSelectionBox FileSelectionDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include<Xm/FileSB.h>

Widget XmCreateFileSelectionDialog(*Parent, Name,*
ArgumentList,
ArgumentCount)

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateFileSelectionDialog** is a convenience subroutine that creates a **DialogShell** widget and an unmanaged **FileSelectionBox** child of the **DialogShell** widget. A **FileSelectionDialog** widget selects a file. It includes the following:

- An editable text field for the directory mask.
- A scrolling list of file names.
- An editable text field for the selected file.
- Labels for the list and text fields.
- Four buttons.

The default button labels are: **OK**, **Filter**, **Cancel**, and **Help**. One additional **WorkArea** can be added to the **FileSelectionBox** widget after creation.

Use the **XtManageChild** subroutine to pop up the **FileSelectionBox** widget (passing the **XmFileSelectionBox** widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **FileSelectionBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/FileSB.h

Related Information

The **XmFileSelectionBox** widget class, **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateForm Subroutine

Purpose

The **Form** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Form.h>

Widget XmCreateForm(*Parent*, *Name*, *ArgumentList*, *ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateForm** subroutine creates an instance of a **Form** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **Form** widget ID.

File

/usr/include/Xm/Form.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmForm** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateFormDialog Subroutine

Purpose

A **Form FormDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Form.h>

Widget XmCreateFormDialog(*Parent*, *Name*, *ArgumentList*, *ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateFormDialog** subroutine is a convenience creation subroutine that creates a **DialogShell** widget and an unmanaged **Form** widget child of the **DialogShell** widget. A **FormDialog** widget is used for interactions not supported by the standard dialog set. This subroutine does not automatically create any labels, buttons, or other dialog components. Such components should be added by the application after the **FormDialog** widget is created.

Use the **XtManageChild** subroutine to pop up the **XmFormDialog** subroutine (passing the **Form** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **Form** widget ID.

File

/usr/include/Xm/Form.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmForm** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateFrame Subroutine

Purpose

The **Frame** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Frame.h>

Widget XmCreateFrame(*Parent*, *Name*, *ArgumentList*,
ArgumentCount)
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateFrame** subroutine creates an instance of a **Frame** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies a name for the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateFrame** subroutine returns the **Frame** widget ID.

File

/usr/include/Xm/Frame.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmFrame** widget, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateInformationDialog Subroutine

Purpose

The **MessageBox InformationDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h>

Widget XmCreateInformationDialog(*Parent, Name, ArgumentList,*
*ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateInformationDialog** subroutine creates a **DialogShell** widget and an unmanaged **MessageBox** child of the **DialogShell** widget. An **InformationDialog** widget gives the user information, such as the status of an action. It includes a symbol, a message, and three buttons. The default symbol is a lower case **i**. The default button labels are **OK**, **Cancel**, and **Help**.

Use the **XtManageChild** subroutine to pop up the **InformationDialog** widget (passing the **MessageBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateInformationDialog** subroutine returns the **MessageBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MessageB.h

Related Information

The **XmMessageBox** widget class, **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateLabel Subroutine

Purpose

The **Label** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Label.h>

Widget XmCreateLabel(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateLabel** subroutine creates an instance of a **Label** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies a name for the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateLabel** subroutine returns the **Label** widget ID.

File

/usr/include/Xm/Label.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmLabel** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateLabelGadget Subroutine

Purpose

The **LabelGadget** creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/LabelGadget.h>

Widget XmCreateLabelGadget(*Parent*, *Name*, *ArgumentList*, *ArgumentCount*)
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateLabelGadget** subroutine creates an instance of a **LabelGadget** gadget and returns the associated gadget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies a name for the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateLabelGadget** subroutine returns the **LabelGadget** gadget ID.

File

/usr/include/Xm/LabelGadget.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmLabelGadget** gadget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateList Subroutine

Purpose

The **List** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmCreateList(*Parent*, *Name*, *ArgumentList*,
ArgumentCount)
Widget *Widget*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateList** subroutine creates an instance of a **List** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies a name for the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateList** returns the **List** widget ID.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateMainWindow Subroutine

Purpose

The **MainWindow** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MainWindow.h>

Widget XmCreateMainWindow(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateMainWindow** subroutine creates an instance of a **MainWindow** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateMainWindow** subroutine returns the **MainWindow** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MainWindow.h

Related Information

The **XmMainWindow** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateMenuBar Subroutine

Purpose

A **RowColumn** widget convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Widget XmCreateMenuBar(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent;*
String *Name;*
ArgList *ArgumentList;*
Cardinal *ArgumentCount;*

Description

The **XmCreateMenuBar** subroutine is a convenience subroutine that creates an instance of a **RowColumn** widget of the **XmMENU_BAR** type and returns the associated widget ID.

This subroutine creates a **RowColumn** widget configured to operate as a **MenuBar** widget and is not implemented as a separate widget class.

The **MenuBar** widget is generally used for building a **Pulldown** menu system. Typically, a **MenuBar** widget is created and placed along the top of the application window, and several **CascadeButton** widgets are inserted as the children. Each **CascadeButton** widget has a **Pulldown MenuPane** associated with it. These **Pulldown MenuPanes** must have been created as children of the **MenuBar** widget. The user interacts with the **MenuBar** by using either the mouse or the keyboard.

The **MenuBar** widget displays a three–dimensional shadow along its border. The client application controls the shadow resources using the visual resources supported by the **XmManager** widget.

The **MenuBar** widget is homogeneous in that it only accepts children that are a subclass of the **CascadeButton** widget. Attempting to insert a child of a different widget class results in a warning message.

If the **MenuBar** widget does not have enough room to fit all of its subwidgets on a single line, the **MenuBar** widget attempts to wrap the remaining entries onto additional lines if allowed by the geometry manager of the parent widget.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **RowColumn** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h

Related Information

The **XmCascadeButton** widget class, **XmManager** widget class, **XmRowColumn** widget class, **XmCreatePulldownMenu** subroutine, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateMenuShell Subroutine

Purpose

The **MenuShell** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MenuShell.h>

Widget XmCreateMenuShell(*Parent*, *Name*, *ArgumentList*,
*ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateMenuShell** subroutine creates an instance of a **MenuShell** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **MenuShell** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MenuShell.h

Related Information

The **XmMenuShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateMessageBox Subroutine

Purpose

The **MessageBox** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Message.h>

Widget XmCreateMessageBox(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateMessageBox** subroutine creates an instance of a **MessageBox** widget and returns the associated widget ID.

A **MessageBox** widget is used for common interaction tasks, which include giving information, asking questions, and reporting errors. It includes the following:

- An optional symbol
- A message
- Three buttons.

By default, there is no symbol. The default button labels are **OK**, **Cancel**, and **Help**.

If the parent of the **MessageBox** widget is a **DialogShell** widget, use the **XtManageChild** subroutine to pop up the **MessageBox** widget (passing the **MessageBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **MessageBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Message.h

Related Information

The **XmMessageBox** widget class, **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateMessageDialog Subroutine

Purpose

The **MessageBox MessageDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h>

Widget XmCreateMessageDialog(*Parent, Name, ArgumentList,*
*ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateMessageDialog** subroutine is a convenience creation subroutine that creates a **DialogShell** widget and an unmanaged **MessageBox** child of the **DialogShell** widget. A **MessageDialog** widget is used for common interaction tasks, which include giving information, asking questions, and reporting errors. It includes a symbol, a message, and three buttons. By default, there is no symbol. The default button labels are **OK**, **Cancel**, and **Help**.

Use the **XtManageChild** subroutine to pop up the **XmMessageDialog** widget (passing the **MessageBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateMessageDialog** subroutine returns the **MessageBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MessageB.h

Related Information

The **XmMessageBox** widget class, **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateOptionMenu Subroutine

Purpose

A **RowColumn** widget convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Widget XmCreateOptionMenu(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateOptionMenu** subroutine creates an instance of a **RowColumn** widget of the type **XmMENU_OPTION** and returns the associated widget ID.

It is provided as a convenience subroutine for creating a **RowColumn** widget configured to operate as an **OptionMenu** widget, which is not implemented as a separate widget class.

The **OptionMenu** widget is a specialized **RowColumn** widget manager composed of a label, a selection area, and a single **Pulldown MenuPane**. When an application creates an **OptionMenu** widget, it supplies the label string and the **Pulldown MenuPane**. For the creation to succeed, there must be a valid **XmNsubMenuId** resource set when calling this subroutine. When the **OptionMenu** widget is created, the **Pulldown MenuPane** must have been created as a child of the **OptionMenu** parent widget and must be specified.The **LabelGadget** gadget and the selection area (a **CascadeButton** widget) are created by the **OptionMenu** widget.

An **OptionMenu** widget is laid out with the label displayed on the left side of the widget and the selection area on the right side. The selection area has a dual purpose; it displays the label of the last item selected from the associated **Pulldown MenuPane**, and it provides the means for posting the **Pulldown MenuPane**.

An **OptionMenu** widget typically does not display any 3–D visuals around itself or the internal **LabelGadget** gadget. By default, the internal **CascadeButtonGadget** gadget has a visible 3–D shadow. The application may change the value of the parameter associated with this resource by getting the **XmCascadeButtonGadget** gadget ID using the **XmOptionButtonGadget** subroutine, and then calling the **XtSetValues** subroutine using the standard visual–related resources.

The **Pulldown MenuPane** is posted by moving the mouse pointer over the selection area and pressing the mouse button defined by the **OptionMenu** widget's **XmNwhichButton** resource. The **Pulldown MenuPane** is posted and positioned so that the last selected item is directly over the selection area. The mouse is then used to arm the desired menu item. When the mouse button is released, the armed menu item is selected and the label within the selection area is changed to match that of the selected item. By default, the left mouse button (Button 1) is used to interact with an **OptionMenu** widget. The default can be changed through the **RowColumn** resource **XmNwhichButton**.

An **OptionMenu** widget also operates by using the keyboard interface mechanism. If the application has established a mnemonic with the **OptionMenu** widget, typing the mnemonic causes the **Pulldown MenuPane** to be posted with keyboard traversal enabled. The standard traversal keys can then be used to move within the **MenuPane** widget. Selection can occur as a result of pressing the Return key or typing a mnemonic or accelerator for one of the menu items.

An application can use the **XmNmenuHistory** resource to indicate which item in the **Pulldown MenuPane** widget should be treated as the current choice and have its label displayed in the selection area. By default, the first item in the **Pulldown MenuPane** widget is used.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **RowColumn** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h

Related Information

The **XmCacadeButtonGadget** gadget class, **XmCreatePulldownMenu** subroutine, **XmLabelGadget** gadget class, **XmRowColumn** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreatePanedWindow Subroutine

Purpose

The **PanedWindow** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <X11/Intrinsic.h>
#include <Xm/Xm.h>
#include <Xm/PanedW.h>

Widget XmCreatePanedWindow(*Parent, Name, ArgumentList, ArgumentCount***)**

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreatePanedWindow** subroutine creates an instance of a **XmPanedWindow** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreatePanedWindow** subroutine returns the **XmPanedWindow** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/X11/Intrinsic.h
/usr/include/Xm/Xm.h
/usr/include/Xm/PanedW.h

Related Information

The **XmPanedWindow** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreatePopupMenu Subroutine

Purpose

A **RowColumn** widget convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Widget XmCreatePopupMenu(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreatePopupMenu** subroutine is a convenience subroutine that creates an instance of a **RowColumn** widget of type **XmMENU_POPUP** and returns the associated widget ID. When this subroutine is used to create the **Popup Menupane** widget, a **MenuShell** widget is automatically created as the *Parent* of the **MenuPane** widget. The parent of the **MenuShell** widget is the widget indicated by the *Parent* parameter.

The **XmCreatePopupMenu** subroutine creates a **RowColumn** widget configured to operate as a **Popup MenuPane** and is not implemented as a separate widget class.

The **XmPopupMenu** subroutine is used as the first **MenuPane** widget within a **XmPopupMenu** system; all other **MenuPane** widgets are of the **Pulldown** type. A **Popup MenuPane** widget displays a three–dimensional shadow, unless the feature is disabled by the application. The shadow appears around the edge of the **MenuPane** widget.

The **Popup MenuPane** widget must be created as the child of a **MenuShell** widget in order to function properly when it is incorporated into a menu. If the application uses this convenience subroutine to create a **Popup MenuPane**, the **MenuShell** widget is automatically created as the real parent of the **XmMenuPane** widget. If the application does not use this convenience subroutine to create the **XmRowColumn** widget to subroutine as a **Popup MenuPane**, it is the application's responsibility to create the **MenuShell** widget.

To access the **Popup Menu** widget, the application must first position the widget using the **XmMenuPosition** subroutine and then manage it using the **XtManageChild** subroutine.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h>

Related Information

The **XmRowColumn** widget, **XmMenuPosition** subroutine, **XmMenuShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreatePromptDialog Subroutine

Purpose

The **SelectionBox PromptDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

AIXwindows Library (**libIM.a**)

Syntax

#include <Xm/SelectioB.h>

Widget XmCreatePromptDialog(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreatePromptDialog** subroutine is a convenience subroutine that creates a **DialogShell** widget and an unmanaged **SelectionBox** child of the **DialogShell** widget. A **PromptDialog** widget prompts the user for text input. It includes a message, a text input region, and three managed buttons. The default button labels are **OK**, **Cancel**, and **Help**. An additional button, with **Apply** as the default label, is created unmanaged; it can be explicitly managed if needed. One additional **WorkArea** child can be added to the **SelectionBox** widget after creation.

Note: You should be aware of the proper usage of the **XmText** widget class before using the **XmCreatePromptDialog** subroutine.

Use the **XtManageChild** subroutine to pop up the **PromptDialog** widget (passing the **SelectionBox** as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **SelectionBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/SelectioB.h

Related Information

The **XmSelectionBox** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreatePulldownMenu Subroutine

Purpose

A **RowColumn** widget convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Widget XmCreatePulldownMenu(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreatePulldownMenu** subroutine creates an instance of a **RowColumn** widget of type **XmMENU_PULLDOWN** and returns the associated widget ID. When using this subroutine to create the **Pulldown MenuPane** widget, a **MenuShell** widget is automatically created as the parent of the **MenuPane** widget. If the widget specified by the *Parent* parameter is a **Popup** or **Pulldown MenuPane** widget, the **MenuShell** widget is created as a child of the parent's **MenuShell** widget; otherwise, it is created as a child of the specified *Parent* widget.

The **XmCreatePulldownMenu** subroutine is provided as a convenience subroutine for creating **RowColumn** widgets configured to operate as **Pulldown MenuPane** widgets and is not implemented as a separate widget class.

A **Pulldown MenuPane** widget displays a three–dimensional shadow, unless the feature is disabled by the application. The shadow appears around the edge of the **MenuPane** widget.

A **Pulldown MenuPane** widget is used when creating submenus that are to be attached to a **CascadeButton** widget of **CascadeButtonGadget**. This is the case for all **MenuPane** widgets that are part of a **Pulldown** menu system (a **MenuBar** widget), the **MenuPane** widget associated with an **OptionMenu** widget, and any **MenuPane** widgets that cascade from a **Popup MenuPane** widget. Any **Pulldown MenuPane** widgets that are to be associated with an **XmOptionMenu** widget must be created before the **OptionMenu** widget is created.

The **Pulldown MenuPane** must be attached to a **CascadeButton** widget or **CascadeButtonGadget** which resides in a **MenuBar**, a **Popup MenuPane**, a **Pulldown MenuPane**, or an **OptionMenu**. This is done by using the button resource **XmNsubMenuId**.

A **MenuShell** widget is required between the **Pulldown MenuPane** and its parent. If the application uses this convenience subroutine for creating a **Pulldown MenuPane**, the **MenuShell** is automatically created as the real parent of the **MenuPane**; otherwise, it is the application's responsibility to create the **MenuShell** widget.

To function correctly when incorporated into a menu, the **Pulldown MenuPane** widget's hierarchy must be considered; this hierarchy is dependent upon the type of menu system that is being built:

- If the **Pulldown MenuPane** is to be pulled down from a **MenuBar**, its *Parent* parameter must be the **MenuBar**.
- If the **Pulldown MenuPane** is to be pulled down from a **Popup** or another **Pulldown MenuPane**, its *Parent* parameter must be **Popup** or **Pulldown MenuPane**.
- If the **Pulldown MenuPane** is to be pulled down from an **OptionMenu**, its *Parent* parameter must be the same as the **OptionMenu** parent.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmRowColumn** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreatePushButton Subroutine

Purpose

The **PushButton** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/PushB.h>

Widget XmCreatePushButton(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreatePushButton** subroutine creates an instance of a **PushButton** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Values

The **XmCreatePushButton** subroutine returns the **PushButton** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/PushB.h

Related Information

The **XmPushButton** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreatePushButtonGadget Subroutine

Purpose

The **PushButtonGadget** creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/PushBG.h>

Widget XmCreatePushButtonGadget(*Parent*, *Name*,
ArgumentList,
ArgumentCount)

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreatePushButtonGadget** subroutine creates an instance of a **PushButtonGadget** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **PushButtonGadget** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/PushBG.h

Related Information

The **XmPushButtonGadget** gadget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateQuestionDialog Subroutine

Purpose

The **MessageBox QuestionDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h>

Widget XmCreateQuestionDialog(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateQuestionDialog** subroutine is a convenience creation subroutine that creates a **DialogShell** widget and an unmanaged **MessageBox** child of a **DialogShell** widget. A **QuestionDialog** widget is used to get the answer to a question from the user. It includes a symbol, a message, and three buttons. The default symbol is a question mark. The default button labels are **OK**, **Cancel**, and **Help**.

Use the **XtManageChild** subroutine to pop up the **XmQuestionDialog** widget (passing the **MessageBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateQuestionDialog** subroutine returns the **MessageBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/MessageB.h

Related Information

The **XmMessageBox** widget, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateRadioBox Subroutine

Purpose

A **RowColumn** widget convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Widget XmCreateRadioBox(*Parent, Name, ArgumentList,*
*ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateRadioBox** subroutine creates an instance of a **RowColumn** widget of the type **XmWORK_AREA** and returns the associated widget ID. Typically, this is a composite widget that contains multiple **ToggleButtonGadget** gadgets. The **RadioBox** widget arbitrates and ensures that only one **ToggleButtonGadget** gadget is on at any given time.

The **ToggleButton** widget is forced to have the **XmNindicatorType** set to the **XmONE_OF_MANY** value and the **XmNvisibleWhenOff** resource set to **True**.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **RowColumn** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h

Related Information

The **XmRowColumn** widget class, **XmToggleButtonGadget** gadget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateRowColumn Subroutine

Purpose

The **RowColumn** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <X11/Intrinsic.h>
#include <Xm/Xm.h>
#include <Xm/RowColumn.h>

Widget XmCreateRowColumn(Parent, Name, ArgumentList,
                                        ArgumentCount)
Widget Parent;
String Name;
Arglist ArgumentList;
Cardinal ArgumentCount;
```

Description

The **XmCreateRowColumn** subroutine creates an instance of an **XmRowColumn** widget and returns the associated widget ID. If **XmNrowColumnType** is not specified, then it is created with **XmWORK_AREA**, which is the default.

If this subroutine is used to create a **Popup Menu** of type **XmMENU_POPUP** or a **Pulldown Menu** of type **XmMENU_PULLDOWN**, a **MenuShell** widget is not automatically created as the parent of the Menupane. The application must first create the **MenuShell** widget by using either the **XmCreateMenuShell** subroutine or the **XtCreateWidget** subroutine.

The **XmRowColumn** widget contains a complete definition of the **XmRowColumn** subroutine and its associated resources.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Values

The **XmCreateRowColumn** subroutine returns the **XmRowColumn** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/X11/Intrinsic.h
/usr/include/Xm/Xm.h
/usr/include/Xm/RowColumn.h

Related Information

The **XmRowColumn** widget class, **XmCreateMenuShell** subroutine, **XmCreatePopupMenu** subroutine, **XtManageChild** subroutine, **XtUnmanageChild** subroutine, **XtCreateWidget** subroutine.

XmCreateScale Subroutine

Purpose

The **Scale** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Scale.h>

Widget XmCreateScale(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateScale** subroutine creates an instance of a **Scale** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Scale.h

Related Information

The **XmScale** widget, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateScrollBar Subroutine

Purpose

The **ScrollBar** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ScrollBar.h>

Widget XmCreateScrollBar(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateScrollBar** creates an instance of a **ScrollBar** widget and returns the associated widget ID.

Parameters

Parent Specifies the parent widget ID.

Name Specifies the name of the created widget.

ArgumentList Specifies the argument list.

ArgumentCount Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Values

The **XmCreateScrollBar** subroutine returns the **ScrollBar** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ScrollBar.h

Related Information

The **XmScrollBar** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateScrolledList Subroutine

Purpose

The **List ScrolledList** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmCreateScrolledList(*Parent*, *Name*, *ArgumentList*, *ArgumentCount*)
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateScrolledList** subroutine creates an instance of a **List** widget that is contained within a **ScrolledWindow** widget. All **ScrolledWindow** subarea widgets are created automatically by this subroutine. The ID returned by this subroutine is that of the **List** widget. Use this ID for all normal **List** widget operations, as well as for those that are relevant for the **ScrolledList** widget.

Other aspects of appearance and behavior of the **ScrolledList** widget should be controlled by using the **ScrolledWindow** widget resource. For instance, to specify the *x,y* location of a **ScrolledList** widget within a larger manager, set the **XmNx** and **XmNy** resources of the **ScrolledWindow** widget rather than that of the **List** widget.

To obtain the ID of the **ScrolledWindow** widget associated with the **ScrolledList** widget, use the **XtParent** subroutine. The name of the **ScrolledWindow** widget created by this subroutine is formed by concatenating the letters **SW** onto the end of the *Name* parameter specified in the parameter list.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies a name for the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

The **XmCreateScrolledList** subroutine returns the **List** widget ID.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class, **XmScrolledWindow** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateScrolledText Subroutine

Purpose

The **TextScrolledText** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

AIXwindows Library (**libIM.a**)

Syntax

#include <Xm/Text.h>

Widget XmCreateScrolledText(*Parent, Name, ArgumentList, ArgumentCount*)
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateScrolledText** subroutine creates an instance of a **Text** widget within a **ScrolledWindow** widget. All **ScrolledWindow** subarea widgets are automatically created by this subroutine. The ID returned by this subroutine is that of the **Text** widget. Use this ID for all normal Text operations as well as for operations relevant for the **ScrolledText** widget.

Note: You should be aware of the proper usage of the **XmText** widget class before using this subroutine.

The **Text** widget defaults to single–line text edit; no scroll bars are displayed. The **Text** resource **XmNeditMode** must be set to **XmMULTI_LINE_EDIT** to display the scroll bars.

Other aspects of the appearance and behavior of the **ScrolledText** widget should be controlled by using the **ScrolledWindow** widget resources. For instance, to specify the *x,y* location of a **ScrolledText** widget within a larger manager, set the **XmNx** and **XmNy** resources of the **ScrolledWindow** widget rather than of the **Text** widget.

To obtain the ID of the **ScrolledWindow** widget associated with the **ScrolledText** widget, use the **XtParent** subroutine. The name of the **ScrolledWindow** widget created by this subroutine is formed by concatenating the letters **SW** onto the end of the *Name* parameter specified in the parameter list.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **Text** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmScrolledWindow** widget class, **XmText** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateScrolledWindow Subroutine

Purpose

The **Scrolledwindow** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ScrolledW.h>

Widget XmCreateScrolledWindow(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateScrolledWindow** subroutine creates an instance of a **ScrolledWindow** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget for the widget ID.
Name	Specifies a name for the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **ScrolledWindow** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ScrolledW.h

Related Information

The **XmScrolledWindow** widget, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateSelectionBox Subroutine

Purpose

The **SelectionBox** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/SelectioB.h>

Widget XmCreateSelectionBox(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateSelectionBox** subroutine creates an unmanaged **SelectionBox** widget. A **SelectionBox** widget offers the user a choice from a list of alternatives and gets a selection. It includes the following:

- A scrolling list of alternatives.
- An editable text field for the selected alternative.
- Labels for the list and text field.
- Three buttons.

The default button labels are **OK**, **Cancel**, and **Help**. An **Apply** button is created unmanaged and can be explicitly managed as needed. One additional **WorkArea** child can be added to the **SelectionBox** widget after creation.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **SelectionBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/SelectioB.h

Related Information

The **XmSelectionBox** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateSelectionDialog Subroutine

Purpose

The **SelectionBox SelectionDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

AIXwindows Library (**libIM.a**)

Syntax

#include <Xm/SelectioB.h>

Widget XmCreateSelectionDialog(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateSelectionDialog** subroutine is a convenience subroutine that creates a **DialogShell** widget and an unmanaged **SelectionBox** widget child of the **DialogShell** widget. A **SelectionDialog** widget offers the user a choice from a list of alternatives and gets a selection. It includes the following:

- A scrolling list of alternatives.
- An editable text field for the selected alternative.
- Labels for the text field.
- Three buttons.

Note: You should be aware of the proper usage of the **XmText** widget class before using the **XmCreateSelectionDialog** subroutine.

The default button labels are **OK**, **Cancel**, and **Help**. One additional **WorkArea** child can be added to the **SelectionBox** widget after creation.

Use the **XtManageChild** subroutine to pop up the **SelectionDialog** widget (passing the **SelectionBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **SelectionBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/SelectioB.h

Related Information

The **XmSelectionBox** widget, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateSeparator Subroutine

Purpose

The **Separator** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Separator.h>

Description

The **XmCreateSeparator** subroutine creates an instance of a **Separator** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **Separator** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Separator.h

Related Information

The **XmSeparator** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateSeparatorGadget Subroutine

Purpose

The **SeparatorGadget** creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/SeparatoG.h>

Widget XmCreateSeparatorGadget(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateSeparatorGadget** subroutine creates an instance of a **SeparatorGadget** gadget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **SeparatorGadget** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/SeparatoG.h

Related Information

The **XmSeparatorGadget** gadget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateText Subroutine

Purpose

The **Text** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

AIXwindows Library (**libIM.a**)

Syntax

#include <Xm/Text.h>

Widget XmCreateText(*Parent, Name, ArgumentList,*
*ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateText** subroutine creates an instance of a **Text** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **Text** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateToggleButton Subroutine

Purpose

The **ToggleButton** widget creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleB.h>

Widget XmCreateToggleButton(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateToggleButton** subroutine creates an instance of a **ToggleButton** widget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **ToggleButton** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleB.h

Related Information

The **XmToggleButton** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateToggleButtonGadget Subroutine

Purpose

The **ToggleButtonGadget** creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleBG.h>

Widget XmCreateToggleButtonGadget(*Parent, Name,*
ArgumentList,
*ArgumentCount***)**

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateToggleButtonGadget** subroutine creates an instance of a **ToggleButtonGadget** gadget and returns the associated widget ID.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Values

Returns the **ToggleButtonGadget** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleBG.h

Related Information

The **XmToggleButtonGadget** gadget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateWarningDialog Subroutine

Purpose

A **MessageBox WarningDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h>

Widget XmCreateWarningDialog(*Parent, Name,*
ArgumentList,
*ArgumentCount***)**

Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateWarningDialog** subroutine is a convenience creation subroutine that creates a **DialogShell** widget and an unmanaged **MessageBox** child of the **DialogShell** parent widget. A **WarningDialog** widget warns the user of action consequences and gives the user a choice of resolutions. It includes a symbol, a message, and three buttons. The default symbol is an exclamation point. The default button labels are **OK**, **Cancel**, and **Help**.

Use the **XtManageChild** subroutine to pop up the **WarningDialog** widget (passing the **MessageBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MessageB.h

Related Information

The **XmMessageBox** widget class, **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCreateWorkingDialog Subroutine

Purpose

The **MessageBox WorkingDialog** convenience creation subroutine.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h

Widget XmCreateWorkingDialog(*Parent, Name, ArgumentList, ArgumentCount***)**
Widget *Parent*;
String *Name*;
ArgList *ArgumentList*;
Cardinal *ArgumentCount*;

Description

The **XmCreateWorkingDialog** subroutine is a convenience subroutine that creates a **DialogShell** widget and an unmanaged **MessageBox** widget child of the **DialogShell** widget. A **WorkingDialog** widget informs the user that there is a time–consuming operation in progress and gives the user the ability to cancel the operation. It includes a symbol, a message, and three buttons. The default symbol is an hourglass. The default button labels are **OK**, **Cancel**, and **Help**.

Use the **XtManageChild** subroutine to pop up the **WorkingDialog** widget (passing the **MessageBox** widget as the widget parameter); use the **XtUnmanageChild** subroutine to pop it down.

Parameters

Parent	Specifies the parent widget ID.
Name	Specifies the name of the created widget.
ArgumentList	Specifies the argument list.
ArgumentCount	Specifies the number of resource/value pairs in the *ArgumentList* parameter.

Return Value

Returns the **MessageBox** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MessageB.h

Related Information

The **XmMessageBox** widget class, **XmDialogShell** widget class, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmCvtStringToUnitType Subroutine

Purpose

Converts a string to a unit–type value.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmCvtStringToUnitType(*Arguments*, *NumberArguments*,
FromValue, *ToValue*)
XrmValuePtr *Arguments*;
Cardinal * *NumberArguments*;
XrmValue * *FromValue*;
XrmValue * *ToValue*;

Description

The **XmCvtStringToUnitType** subroutine converts a string to a unit type. Refer to the **XmGadget** gadget class, the **XmManager** widget class, or the **XmPrimitive** widget class for a description of the valid unit types.

Install this subroutine as a resource converter using the **XtAddConverter** subroutine, rather than calling it directly. The following code segment installs the converter into the toolkit's converter cache. The **XtAddConverter** subroutine permits the **XmNunitType** resource to be specified through a resource file.

XtAddConverter (XmRString, XmRUnitType, XmCvtStringToUnitType, NULL, 0);

The **CvtStringToUnitType** subroutine should only be installed by applications that need to allow the unit type resource to be specified through a resource file. This subroutine must be installed before any widget is created that is to have its **XmNunitType** resource set by data in a resource file.

Parameters

Arguments	Specifies a list of additional **XrmValue** parameters to the converter if additional context is needed to perform the conversion. For example, the string–to–font converter needs the widget screen, or the string–to–pixel converter needs the widget screen and colormap. This parameter is often **NULL**.
NumberArguments	Specifies the number of additional **XrmValue** parameters. This parameter is often zero.
FromValue	Specifies the value to convert.
ToValue	Specifies the descriptor to use to return the converted value.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmGadget** widget class, **XmManager** widget class, **XmPrimitive** widget class, **XtAddConverter** subroutine.

XmDeactivateProtocol Subroutine

Purpose

A **VendorShell** subroutine that deactivates a protocol without removing it.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <X11/protocols.h>
#include <Xm/Xm.h>

void XmDeactivateProtocol(*Shell*, *Property*, *Protocol***)**
Widget *Shell*;
Atom *Property*;
Atom *Protocol*;

Description

The **XmDeactivateProtocol** subroutine updates the handlers, and updates the *Property* parameter if the *Shell* parameter is realized. It is sometimes useful to allow a protocol's state information (callback routine lists, etc.) to persist, even though the client application may choose to resign temporarily from the interaction. The main use of this capability is to "gray/ungray" **f.send_msg** entries in the **mwm** system menu. The "graying/ungraying" of menu entries is accomplished by changing the state of a *Protocol* parameter from active to inactive, or the other way around. If the *Protocol* parameter is active and the *Shell* parameter is realized, the *Property* parameter will contain the *Protocol* parameter **Atom**. If the *Protocol* parameter is inactive, the **Atom** *Protocol* parameter will not be present in the *Property* parameter.

Parameters

Shell	Specifies the widget with which the protocol property is associated.
Property	Specifies the protocol property.
Protocol	Specifies the protocol atom (or an **int** cast to **Atom**).

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/protocols.h

Related Information

The **VendorShell** widget class, **XmInternAtom** subroutine.

XmDestroyPixmap Subroutine

Purpose

A pixmap–caching subroutine that removes a pixmap from the image cache.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmDestroyPixmap(*Screen*, *Pixmap*)
Screen * *Screen*;
Pixmap *Pixmap*;

Description

The **XmDestroyPixmap** subroutine removes pixmaps that are no longer used. Pixmaps are only completely freed when there is no further references to them.

Parameters

Screen	Specifies the display screen for which the pixmap was requested.
Pixmap	Specifies the pixmap to be destroyed.

Return Values

Returns **True** when successful; returns **False** if there is no matching screen and pixmap in the pixmap cache.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmInstallImage** subroutine, **XmUninstallImage** subroutine, **XmGetPixmap** subroutine.

XmFileSelectionBoxGetChild Subroutine

Purpose

A **FileSelectionBox** subroutine that is used to access a component.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/FileSB.h>

Widget XmFileSelectionBoxGetChild(*Widget*, *Child***)**
Widget *Widget*;
unsigned char *Child*;

Description

The **XmFileSelectionBoxGetChild** subroutine is used to access a component within a **FileSelectionBox** widget. The parameters given to the subroutine are the **FileSelectionBox** widget and a value indicating which child to access.

Parameters

Widget Specifies the **FileSelectionBox** widget ID.

Child Specifies a component within the **XmFileSelectionBox** widget. The following are legal values for this parameter:

- **XmDIALOG_APPLY_BUTTON**
- **XmDIALOG_CANCEL_BUTTON**
- **XmDIALOG_DEFAULT_BUTTON**
- **XmDIALOG_FILTER_LABEL**
- **XmDIALOG_FILTER_TEXT**
- **XmDIALOG_HELP_BUTTON**
- **XmDIALOG_LIST**
- **XmDIALOG_LIST_LABEL**
- **XmDIALOG_OK_BUTTON**
- **XmDIALOG_SELECTION__LABEL**
- **XmDIALOG_TEXT**

Return Value

Returns the widget ID of the specified **FileSelectionBox** child.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/FileSB.h

Related Information

The **XmFileSelectionBox** widget class.

XmFileSelectionDoSearch Subroutine

Purpose

A **FileSelectionBox** subroutine that initiates a directory search.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/FileSB.h>

void XmFileSelectionDoSearch(*Widget*, *DirectoryMask***)**
Widget *Widget*;
XmString *DirectoryMask*;

Description

The **XmFileSelectionDoSearch** subroutine initiates a directory search. If the *DirectoryMask* parameter is not **NULL,** the directory mask is updated before the search is initiated.

Parameters

Widget	Specifies the **FileSelectionBox** widget ID.
DirectoryMask	Specifies the directory mask used in determining the files displayed in the **FileSelectionBox** widget. This sets the **XmNdirMask** resource associated with the **XmCreateFileSelectionBox** subroutine. This is an optional resource. If you do not specify a directory mask, the current directory mask is used.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/FileSB.h

Related Information

The **XmFileSelectionBox** widget class.

XmFontListAdd Subroutine

Purpose

A compound string subroutine that creates a new font list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmFontList XmFontListAdd(*OldList*, *Font*, *CharacterSet***)**
XmFontList *OldList*;
XFontStruct * *Font*;
XmStringCharSet *CharacterSet*;

Description

The **XmFontListAdd** subroutine creates a new font list consisting of the contents of the *OldList* parameter and the new font list element being added. This subroutine deallocates the *OldList* parameter after extracting the required information; therefore, do not reference the *OldList* parameter thereafter.

Parameters

OldList	Specifies a pointer to the font list to which an entry is to be added.
Font	Specifies a pointer to a font structure for which the new font list is generated. This is the structure returned by the **Xlib XLoadQueryFont** subroutine.
CharacterSet	Specifies the character set identifier for the font. This can be **XmSTRING_DEFAULT_CHARSET**.

Return Value

Returns a new font list.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmFontListCreate** subroutine.

XmFontListCreate Subroutine

Purpose

A compound string subroutine that creates a font list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmFontList XmFontListCreate(*Font*, *CharacterSet***)**
XFontStruct * *Font*;
XmStringCharSet *CharacterSet*;

Description

The **XmFontListCreate** subroutine creates a new font list with a single element specified by the provided font and character set. This subroutine also allocates the space for the font list.

Parameters

Font	Specifies a pointer to a font structure for which the new font list is generated. This is the structure returned by the **XLib** Library **XLoadQueryFont** subroutine.
CharacterSet	Specifies the character set identifier for the font. This can be the **XmSTRING_DEFAULT_CHARSET** value.

Return Value

Returns a new font list.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmFontListAdd** subroutine, **XmFontListFree** subroutine, **XmStringBaseline** subroutine, **XmStringByteCompare** subroutine, **XmStringCompare** subroutine, **XmStringConcat** subroutine, **XmStringCopy** subroutine, **XmStringCreate** subroutine, **XmStringCreateLtoR** subroutine, **XmStringDirectionCreate** subroutine, **XmStringDraw** subroutine, **XmStringDrawImage** subroutine, **XmStringDrawUnderline** subroutine, **XmStringEmpty** subroutine, **XmStringExtent** subroutine, **XmStringFree** subroutine, **XmStringFreeContext** subroutine, **XmStringGetLtoR** subroutine, **XmStringGetNextComponent** subroutine, **XmStringGetNextSegment** subroutine, **XmStringHeight** subroutine, **XmStringInitContext** subroutine, **XmStringLength** subroutine, **XmStringLineCount** subroutine, **XmStringNConcat** subroutine, **XmStringNCopy** subroutine, **XmStringPeekNextComponent** subroutine,

XmStringSegmentCreate subroutine, **XmStringSeparatorCreate** subroutine, **XmStringWidth** subroutine.

XmFontListFree Subroutine

Purpose

A compound string subroutine that recovers memory used by a font list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmFontListFree(*List***)**
XmFontList *List*;

Description

The **XmFontListFree** subroutine recovers memory used by a font list.

Parameter

List Specifies the font list to be freed.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmFontListCreate** subroutine.

XmGetMenuCursor Subroutine

Purpose

A **Row Column** subroutine that returns the cursor ID for the current menu cursor.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Cursor XmGetMenuCursor(*Display***)**
Display * *Display*;

Description

The **XmGetMenuCursor** subroutine queries the menu cursor currently being used by this client application on the specified display, and then returns the cursor ID.

Parameters

Display Specifies the display whose menu cursor is to be queried.

Return Value

This subroutine returns the cursor ID for the current menu cursor of the value **None** if a cursor is not yet defined. A cursor is be defined if the application makes this call before the client has created any menus on the specified display.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmRowColumn** widget class.

XmGetPixmap Subroutine

Purpose

A pixmap caching subroutine that generates a pixmap, stores it in an image cache, and returns the pixmap.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Pixmap XmGetPixmap(*Screen, ImageName, Foreground, Background***)**
Screen **Screen*;
char **ImageName*;
Pixel *Foreground*;
Pixel *Background*;

Description

The **XmGetPixmap** subroutine uses the parameter data to perform a lookup in the pixmap cache to see if a pixmap that matches the data has already been generated. If one is found, a reference count is incremented and the pixmap is returned. Applications should use the **XmDestroyPixmap** subroutine when the pixmap is no longer needed.

If a pixmap is not found, the *ImageName* parameter is used to perform a lookup in the image cache. If an image is found, it is used to generate the pixmap, which is then cached and returned.

If an image is not found, the *ImageName* parameter is used as a file name, and a search is made for an **X10** or **X11** bitmap file. If it is found, the file is read, converted into an image, and cached in the image cache. The image is then used to generate a pixmap, which is cached and returned.

Several paths are searched to find the file. The user can specify the environment variable **XBMLANGPATH**, which is used to generate one set of paths. The **XtInitialize** subroutine explains the use of this environment variable. If the **XBMLANGPATH** environment variable is not set, the following path names are searched:

/usr/lib/X11/%L/bitmaps/%N/%B

/usr/lib/X11/%L/bitmaps/%b

/usr/lib/X11/bitmaps/%B

/usr/include/X11/bitmaps/%B

Parameters

Screen	Specifies the display screen on which the pixmap is to be drawn and ensures that the pixmap matches the visual required for the screen.
ImageName	Specifies the name of the image used to generate the pixmap.

Foreground	Combines the image with the *Foreground* parameter color to create the pixmap if the image referenced is a bit–per–pixel image.
Background	Combines the image with the *Background* parameter color to create the pixmap if the image referenced is a bit–per–pixel image.

Return Values

The **XmGetPixmap** subroutine returns a pixmap when successful; returns the **XmUNSPECIFIED_PIXMAP** value if the image corresponding to the *ImageName* parameter cannot be found.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmDestroyPixmap** subroutine, **XmInstallImage** subroutine, **XmUninstallImage** subroutine, **XtInitialize** subroutine.

XmInstallImage Subroutine

Purpose

A pixmap–caching subroutine that adds an image to the image cache.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmInstallImage(*Image, ImageName***)**
XImage **Image*;
char **ImageName*;

Description

The **XmInstallImage** subroutine stores an image in an image cache so that the image can later be used to generate a pixmap. Part of the installation process involves extending the resource converter used to refer to these images. The resource converter is given the image name so that the image can be referenced in an **.Xdefaults** file. Since an image can be referenced by a widget, the application must ensure that the image is installed before the widget is created.

The image–caching subroutines provide a set of eight pre–installed images. The names of these images can be used as values within a **.Xdefaults** file to generate pixmaps for the following resources:

Image Name	Description
background	A tile of solid background
25_foreground	A tile of 25% foreground, 75% background
50_foreground	A tile of 50% foreground, 50% background
75_foreground	A tile of 75% foreground, 25% background
horizontal	A tile of horizontal lines of the two colors
vertical	A tile of vertical lines of the two colors
slant_right	A tile of slanting lines of the two colors
slant_left	A tile of slanting lines of the two colors

Parameters

Image Points to the image structure to be installed. The installation process does not make a local copy of the image. Therefore, the application should not destroy the image until it is uninstalled from the caching subroutines.

ImageName Specifies a string that the application uses to name the image. After installation, this name can be used in a **.Xdefaults** file for referencing the

image. A local copy of the name is created by the image–caching subroutines.

Return Values

The **XmInstallImage** subroutine returns a **True** value when successful. A value of **False** is returned if a **NULL** *Image* parameter, a **NULL** *ImageName* parameter, or a duplicate *ImageName* parameter is used as a parameter value.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmUninstallImage** subroutine, **XmGetPixmap** subroutine, **XmDestroyPixmap** subroutine.

XmInternAtom Subroutine

Purpose

A subroutine that returns an atom for a given name.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <X11/AtomMgr.h>

Atom XmInternAtom(*Display*, *Name*, *OnlyIfExists***)**
Display * *Display*;
String *Name*;
Boolean *OnlyIfExists*;

Description

The **XmInternAtom** subroutine returns an atom for a given name. This subroutine mirrors the **Xlib** interfaces for atom management, but provides client side caching. When and where caching is provided in **Xlib**, the routines will become pseudonyms for the **Xlib** routines.

Parameters

Display	Specifies the connection to the **X Server**.
Name	Specifies the name associated with the atom you want returned.
OnlyIfExists	Specifies a Boolean value that indicates whether the **XmInternAtom** subroutine creates the atom.

Return Value

Returns an atom.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/AtomMgr.h

XmIsMotifWMRunning Subroutine

Purpose

Specifies if the window manager is running.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <X11/Shell.h>

Boolean XmIsMotifWMRunning(*Shell***)**
Widget *Shell*;

Description

The **XmIsMotifWMRunning** subroutine lets a user know if the **mwm** is running on a screen that contains a specific widget hierarchy. This subroutine determines whether the **MOTIF_WM_INFO** property is present on the root window of the shell screen. If it is, its window field is used to query for the presence of the specified window as a child of root.

Parameter

Shell Specifies the shell whose screen is to be tested for **mwm**'s presence.

Return Value

The **XmIsMotifWMRunning** subroutine returns **True** if the **mwm** is running.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/X11/Shell.h

XmListAddItem Subroutine

Purpose

A **List** subroutine that adds an item to the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

void XmListAddItem(*Widget, Item, Position***)**
Widget *Widget*;
XmString *Item*;
int *Position*;

Description

The **XmListAddItem** subroutine adds an item at a given position. The position specifies the location of the new item in the list. For example, Position 1 is the first element and position 2 is the second, etc.. If the *Position* parameter is zero, the item is added after the last item in the list, When the item is inserted into the list, it is compared against the current **XmNselectedItems** list. If the new item matches an item on the selected list, it appears selected.

Parameters

Widget	Specifies the ID of the **List** widget from which list an item is added.
Item	Specifies the text of the item to be added to the list.
Position	Specifies the placement of the item within the list in terms of its cell position. It uses an insert mode/cell number scheme with a 1 specifying the top–entry position and a zero specifying the bottom entry for adding an item to the bottom of the list.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListAddItemUnselected Subroutine

Purpose

A **List** subroutine that adds an item to the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

void XmListAddItemUnselected(*Widget, Item, Position***)**
Widget *Widget*;
XmString *Item*;
int *Position*;

Description

The **XmListAddItemUnselected** subroutine adds an item to the list at the given position. The position specifies the location of the new item in the list. Position one is the first element, position two is the second, and so on. If the *Position* parameter is zero, the item is added after the last item in the list.

Parameters

Widget	Specifies the ID of the **List** widget from whose list an item is added.
Item	Specifies the text of the item to be added to the list.
Position	Specifies the placement of the item within the list in terms of its cell position. It uses an insert mode/cell number scheme with a 1 specifying the top–entry position and a zero specifying the bottom entry for adding an item to the bottom of the list.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/List.h

Related Information

The **XmList** widget class.

XmListDeleteItem Subroutine

Purpose

A **List** subroutine that deletes an item from the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListDeleteItem(*Widget*, *Item***)**
Widget *Widget*;
XmString *Item*;

Description

The **XmListDeleteItem** subroutine deletes a specified item from the list. A warning message is displayed if the item does not exist.

Parameters

Widget	Specifies the ID of the **List** from whose list an item is added.
Item	Specifies the text of the item to be added to the list.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListDeletePos Subroutine

Purpose

A **List** subroutine that deletes an item from a list at a specified position.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

void XmListDeletePos(*Widget*, *Position***)**
Widget *Widget*;
int *Position*;

Description

The **XmListDeletePos** subroutine deletes an item at a specified position. A *Position* parameter of zero deletes the last item in the list. A warning message is displayed if the position does not exist.

Parameters

Widget	Specifies the ID of the **List** widget from which list an item is added.
Position	Identifies the position of the item to be deleted.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListDeselectAllItems Subroutine

Purpose

A **List** subroutine that unhighlights and removes all items from the selected list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListDeselectAllItems(*Widget***)**
Widget *Widget*;

Description

The **XmListDeselectAllItems** subroutine unhighlights and removes all items from the selected list.

Parameter

Widget	Specifies the ID of the **List** widget from which list all selected items are deselected.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListDeselectItem Subroutine

Purpose

A **List** subroutine that deselects the specified item from the selected list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListDeselectItem(*Widget*, *Item***)**
Widget *Widget*;
XmString *Item*;

Description

The **XmListDeselectItem** subroutine unhighlights and removes the specified item from the selected list.

Parameters

Widget	Specifies the ID of the **List** widget from which list an item is deselected.
Item	Specifies the text of the item to be deselected from the list.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListDeselectPos Subroutine

Purpose

A **List** subroutine that deselects an item at a specified position in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListDeselectPos(*Widget, Item***)**
Widget *Widget*;
int *Position*;

Description

The **XmListDeselectPos** subroutine unhighlights the item at a specified position and deletes it from the selected list.

Parameters

Widget	Specifies the ID of the **List** widget.
Item	Identifies the specified position.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListItemExists Subroutine

Purpose

A **List** subroutine that determines whether a specified item is in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

Boolean XmListItemExists(*Widget, Item***)**
Widget *Widget*;
XmString *Item*;

Description

The **XmListItemExists** subroutine is a Boolean subroutine that determines whether a specified item is present in the list.

Parameters

Widget	Specifies the ID of a **List** widget.
Item	Specifies the text of the given item.

Return Value

The **XmListItemExists** subroutine returns **True** if the specified item is present in the list.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/List.h

Related Information

The **XmList** widget class.

XmListSelectItem Subroutine

Purpose

A **List** subroutine that selects an item in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListSelectItems(*Widget, Item, Notify*)
Widget *Widget*;
XmString *Item*;
Boolean *Notify*;

Description

The **XmListSelectItem** subroutine highlights and adds the specified item to the current selected list.

Parameters

Widget	Specifies the ID of the **List** widget from which list an item is selected.
Item	Specifies the item to be added to the **List** widget.
Notify	Specifies a Boolean value that, when a **True** value invokes the selection callback for the current mode. From an application interface view, calling this subroutine with the *Notify* parameter as a **True** value is indistinguishable from a user–initiated selection action.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListSelectPos Subroutine

Purpose

A **List** subroutine that selects an item at a specified position in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListSelectPos(*Widget, Position, Notify***)**
Widget *Widget*;
int *Position*;
Boolean *Notify*;

Description

The **XmListSelectPos** subroutine highlights an item at the specified position and adds it to the current selected list. A position of zero specifies the last item in the list.

Parameters

Widget	Specifies the ID of the **List** widget.
Position	Identifies the specified position.
Notify	Specifies a Boolean value that when a **True** value invokes the selection callback for the current mode. From an application interface view, calling this subroutine with the *Notify* parameter as a **True** value is indistinguishable from a user–initiated selection action.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListSetBottomItem Subroutine

Purpose

A **List** subroutine that makes an existing item the last visible item in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

void XmListSetBottomItem(*Widget, Item***)**
Widget *Widget*;
XmString *Item*;

Description

The **XmListSetBottomItem** subroutine makes an existing item the last visible item in the list. The item can be any valid item in the list.

Parameters

Widget	Specifies the ID of the **List** widget from whose list an item is made the last visible item.
Item	Specifies the text of the given item.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/List.h

Related Information

The **XmList** widget class.

XmListSetBottomPos Subroutine

Purpose

A **List** subroutine that makes a specified item the last visible position in a list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

void XmListSetBottomPos(*Widget, Position***)**
Widget *Widget*;
int *Position*;

Description

The **XmListSetBottomPos** subroutine makes a given item the last visible position in a list. The position can be any valid position in the list. A position of zero specifies the last item in the list.

Parameters

Widget Specifies the ID of the **List** widget.

Position Identifies the specified position.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/List.h

Related Information

The **XmList** widget class.

XmListSetHorizPos Subroutine

Purpose

A **List** subroutine that moves a **Scrollbar** widget to the specified position in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListSetHorizPos(*Widget*, *Position*)
Widget *Widget*;
int *Position*;

Description

The **XmListSetHorizPos** subroutine sets the **XmNvalue** resource of the **ScrollBar** widget to a specified position and updates the visible portion of the list with the new value if the **List** widget **XmNlistSizePolicy** resource is set to the **XmCONSTANT** value or the **XmRESIZE_IF_POSSIBLE** value and if the horizontal **ScrollBar** widget is currently visible. This is equivalent to moving the **ScrollBar** widget to the specified position.

Parameters

Widget	Specifies the ID of the **List** widget.
Position	Identifies the specified position.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListSetItem Subroutine

Purpose

A **List** subroutine that makes an existing item the first visible item in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListSetPos(*Widget*, *Item*)
Widget *Widget*;
XmString *Item*;

Description

The **XmListSetItem** subroutine makes an existing item the first visible item in the list. The item can be any valid item in the list.

Parameters

Widget	Specifies the ID of the **List** widget from which list an item is made the first visible item in the list.
Item	Specifies the given item.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmListSetPos Subroutine

Purpose

A **List** subroutine that makes the item at the given position the first visible position in the list.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/List.h>

XmListSetPos(*Widget*, *Position*)
Widget *Widget*;
int *Position*;

Description

The **XmListSetPos** subroutine makes the item at the given position the first visible position in the list. The position can be any valid position in the list.

Parameters

Widget	Specifies the ID of the **List** widget.
Position	Identifies the specified position.

File

/usr/include/Xm/List.h

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmList** widget class.

XmMainWindowSep1 Subroutine

Purpose

A **MainWindow** subroutine that returns the widget ID of the first **Separator** widget.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MainW.h>

Widget XmMainWindowSep1(*Widget***)**
Widget *Widget*;

Description

The **XmMainWindowSep1** subroutine returns the widget ID of the first **Separator** widget in a **MainWindow** widget. The first **Separator** widget is located between the **MenuBar** and the **CommandWindow** widget. This **Separator** widget is visible only when the **XmNshowSeparator** resource is **True**.

Parameter

Widget Specifies the **MainWindow** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MainW.h

Related Information

The **XmMainWindow** widget class.

XmMainWindowSep2 Subroutine

Purpose

A **MainWindow** subroutine that returns the widget ID of the second **Separator** widget.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MainW.h>

Widget XmMainWindowSep2(*Widget***)**
Widget *Widget*;

Description

The **XmMainWindowSep2** subroutine returns the widget ID of the second **Separator** widget in a **MainWindow** widget. The second **Separator** widget is located between the **CommandWindow** widget and the **ScrolledWindow** widget. This **Separator** is visible only when the **XmNshowSeparator** resource is **True**.

Parameter

Widget Specifies the **MainWindow** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MainW.h

Related Information

The **XmMainWindow** widget class.

XmMainWindowSetAreas Subroutine

Purpose

A **MainWindow** subroutine that identifies the manageable children for each area.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MainW.h>

void XmMainWindowSetAreas(*Widget*, *MenuBar*, *CommandWindow*, *HorizontalScrollbar*, *VerticalScrollbar*, *WorkRegion*)

Widget *Widget*;
Widget *MenuBar*;
Widget *CommandWindow*;
Widget *HorizontalScrollbar*;
Widget *VerticalScrollbar*;
Widget *WorkRegion*;

Description

The **XmMainWindowSetAreas** subroutine identifies for each area the valid children (**MenuBar**, **WorkRegion**, and so on) that are to be actively managed by the **MainWindow** widget. This subroutine also sets up or adds the **MenuBar**, **Work Window**, **Command Window**, and **ScrollBar** widgets to the application's **MainWindow** widget. The **mwm** provides the title bar.

Each area is optional; the user can pass **NULL** to one or more of the following parameters:

Parameters

Widget	Specifies the **MainWindow** widget ID.
MenuBar	Specifies the widget ID for the **MenuBar** to be associated with the **MainWindow** widget. Set this ID only after creating an instance of the **MainWindow** widget. The resource associated with this parameter is **XmNmenuBar**.
CommandWindow	Specifies the widget ID for the command window to be associated with the **MainWindow** widget. Set this ID only after creating an instance of the **MainWindow** widget. The resource associated with this parameter is **XmNcommandWindow**.
HorizontalScrollbar	Specifies the widget ID for the horizontal **ScrollBar** to be associated with the **MainWindow** widget. Set this ID only after creating an instance of the **MainWindow** widget. The resource associated with this parameter is **XmNhorizontalScrollBar**.
VerticalScrollbar	Specifies the widget ID for the vertical **ScrollBar** to be associated with the **MainWindow** widget. Set this ID only after creating an instance of the **MainWindow** widget. The resource associated with this parameter is **XmNverticalScrollBar**.

WorkRegion	Specifies the widget ID for the work window to be associated with the **MainWindow** widget. Set this ID only after creating an instance of the **MainWindow** widget. The resource associated with this parameter is **XmNworkWindow**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MainW.h

Related Information

The **XmMainWindow** widget class.

XmMenuPosition Subroutine

Purpose

A **RowColumn** subroutine that positions a **Popup MenuPane**.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

void XmMenuPosition(*Menu*, *Event***)**
Widget *Menu*;
XButtonPressedEvent * *Event*;

Description

The **XmMenuPosition** subroutine positions a **Popup MenuPane** widget using the information in the specified event. Unless a client application is positioning the **MenuPane**, it must first invoke this subroutine before managing the **PopupMenu**. The *x root* and *y root* values in the specified event are used to determine the menu position.

Parameters

Menu	Specifies the **PopupMenu** widget to be positioned.
Event	Specifies the event passed to the action procedure that manages the **PopupMenu** widget.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h

Related Information

The **XmRowColumn** widget class.

XmMessageBoxGetChild Subroutine

Purpose

A **MessageBox** subroutine that is used to access a component.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/MessageB.h>

Widget XmMessageBoxGetChild(*Widget, Child***)**
Widget *Widget*;
unsigned char *Child*;

Description

The **XmMessageBoxGetChild** subroutine is used to access a component within a **MessageBox** widget. The parameters given to the subroutine are the **MessageBox** widget and a value indicating which child to access.

Parameters

Widget — Specifies the **MessageBox** widget ID.

Child — Specifies a component within the **MessageBox** widget. The following are legal values for this parameter:

- **XmDIALOG_CANCEL_BUTTON**
- **XmDIALOG_DEFAULT_BUTTON**
- **XmDIALOG_HELP_BUTTON**
- **XmDIALOG_MESSAGE_LABEL**
- **XmDIALOG_OK_BUTTON**
- **XmDIALOG_SEPARATOR**
- **XmDIALOG_SYMBOL_LABEL**

Return Value

Returns the widget ID of the specified **MessageBox** widget child.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/MessageB.h

Related Information

The **XmMessageBox** widget.

XmOptionButtonGadget Subroutine

Purpose

A **RowColumn** subroutine that obtains the widget ID for the **CascadeButtonGadget** in an **OptionMenu**.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Widget XmOptionButtonGadget(*OptionMenu***)**
Widget *OptionMenu*;

Description

The **XmOptionButtonGadget** subroutine provides a client application with the means of obtaining widget ID's for internally created **CascadeButtonGadget** gadgets. Once the application has obtained a widget ID, it has the ability to adjust the visuals for the **CascadeButtonGadget** gadget, if desired.

When an application creates an instance of the **OptionMenu** widget, the widget creates two internal widgets. One is a **LabelGadget** gadget that is used to display the **RowColumn** widget **XmNlabelString** resource. The other is a **CascadeButtonGadget** gadget that displays the current selection and provides the means for posting the **XmOptionMenu** submenu.

Parameter

OptionMenu Specifies the **OptionMenu** widget ID.

Return Value

Returns the widget ID for the internal button.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h

Related Information

The **XmCreateOptionMenu** subroutine, **XmCascadeButtonGadget** gadget class, **XmOptionLabelGadget** subroutine, **XmRowColumn** widget class.

XmOptionLabelGadget Subroutine

Purpose

A **RowColumn** subroutine that obtains the widget ID for the **LabelGadget** in an **OptionMenu**.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/RowColumn.h>

Widget XmOptionLabelGadget(*OptionMenu***)**
Widget *OptionMenu*;

Description

The **XmOptionLabelGadget** subroutine provides the application with the means of obtaining the widget ID for internally created **LabelGadget** gadgets. Once the application has obtained a widget ID, it has the ability to adjust the visuals for the **LabelGadget** gadget, if desired.

When an application creates an instance of the **OptionMenu** widget (a **RowColumn** widget of type **XmMENU_OPTION**), the widget creates two internal gadgets. One is a **LabelGadget** gadget that is used to display a **RowColumn** widget's **XmNlabelString** resource. The other is a **CascadeButtonGadget** gadget that displays the current selection and provides the means for posting the **XmOptionMenu** submenu.

Parameter

OptionMenu Specifies the **OptionMenu** widget ID.

Return Value

Returns the widget ID for an internal label.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/RowColumn.h

Related Information

The **XmCreateOptionMenu** subroutine, **XmLabelGadget** gadget class, **XmOptionLabelGadget** subroutine, **XmRowColumn** widget class, **XmCascadeButtonGadget** gadget class.

XmRemoveProtocolCallback Subroutine

Purpose

A **VendorShell** subroutine that removes a callback routine from the internal list.

Library

AIXwindows Library (**libXm.a**)

Syntax

```
#include <Xm/Xm.h>
#include <X11/Protocols.h>

void XmRemoveProtocolCallback(Shell, Property, Protocol,
                              Callback, Closure)
Widget Shell;
Atom Property;
Atom Protocol;
XtCallbackProc Callback;
caddr_t Closure;
```

Description

The **XmRemoveProtocolCallback** subroutine removes a callback routine from the internal list.

Parameters

Shell	Specifies the widget with which the protocol property is associated.
Property	Specifies the protocol property.
Protocol	Specifies the protocol atom (or an **int** cast to **Atom**).
Callback	Specifies the procedure to call when a protocol message is received.
Closure	Specifies the client data to be passed to the callback when it is invoked.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h

/usr/include/X11/Protocols.h

Related Information

The **VendorShell** widget class, **XmInternAtom** subroutine.

XmRemoveProtocols Subroutine

Purpose

A **VendorShell** subroutine that removes the protocols from the protocol manager and deallocates the internal tables.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <X11/Protocols.h>

void XmRemoveProtocols(*Shell*, *Property*, *Protocols*, *NumberProtocols***)**
Widget *Shell*;
Atom *Property*;
Atom * *Protocols*;
Cardinal *NumberProtocols*;

Description

The **XmRemoveProtocols** subroutine removes the protocols from the protocol manager and deallocates the internal tables. If any of the protocols are active, it updates the handlers and updates the property, if the *Shell* parameter is realized.

Parameters

Shell	Specifies the widget with which the protocol property is associated.
Property	Specifies the protocol property.
Protocols	Specifies the protocol atoms (or **ints** cast to **Atom**).
NumberProtocols	Specifies the number of elements in protocols.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/11/Protocols.h

Related Information

The **VendorShell** widget class, **XmInternAtom** subroutine.

XmRemoveTabGroup Subroutine

Purpose

Removes a tab group.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmRemoveTabGroup(*TabGroup***)**
Widget *TabGroup*;

Description

The **XmRemoveTabGroup** subroutine removes a **Manager** widget or a **Primitive** widget from the list of tab groups associated with a particular widget hierarchy.

Parameter

TabGroup Specifies the **Manager** widget or the **Primitive** widget ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmAddTabGroup** subroutine, **XmManager** widget class, **XmPrimitive** widget class.

XmResolvePartOffsets Subroutine

Purpose

Allows writing of upward–compatible applications and widgets.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/XmP.h>

void XmResolvePartOffsets(*WidgetClass, Offset***)**
WidgetClass *WidgetClass*;
XmOffsetPtr * *Offset*;

Description

The use of offset records requires one extra global variable per widget class. The variable consists of a pointer to an array of offsets into the widget record for each part of the widget structure. The **XmResolvePartOffsets** subroutine allocates the offset records needed by a client application to guarantee upward–compatible applications and widgets. These offset records are used by the widget to access all of the widget variables. A widget needs to take the following steps:

- Instead of creating a resource list, the widget creates an offset resource list. To accomplish this, use the **XmPartResource** structure and the **XmPartOffset** macro. The **XmPartResource** data structure looks just like a resource list, but instead of having one integer for its offset, it has two shorts. This data structure is put into the class record as if it were a normal resource list. Instead of using the **XtOffset** subroutine for the offset, it uses **XmPartOffset**.
- Instead of putting the widget size in the class record, the widget puts the widget part in the same field.
- Instead of putting **XtVersion** in the class record, the widget puts **XtVersionDontCheck** in the class record.
- The widget defines a variable to point to the offset record. This can be part of the widget's class record or a separate global variable.
- In class initialization, the widget calls **XmResolvePartOffsets**, passing it the offset address and the class record. This does several things:
  - Adds the superclass size field (which, by definition, is already initialized) to the part size field.
  - Allocates an array based upon the number of superclasses.
  - Fills in the offsets of all the widget parts with the appropriate values, determined by examining the size fields of all superclass records.
  - Uses the part offset array to modify the offset entries in the resource list to be real offsets, in place.

- Instead of accessing fields directly, the widget must also go through the offset table. You will probably define macros for each area to make this easier. Assume an integer field "xyz":

```
#define BarXyz(w) (*(int *)(((char *)w) + offset[BarIndex] +
\XtOffset(BarPart,xyz)))
```

- The **XmField** macro helps you access these fields. Because the **XmPartOffset** and **XmField** macros concatenate things together, you must ensure there is no space before or after the part parameter. For example, the following does not work because of the space before or after the part (Label) parameter:

```
XmField(w, offset, Label, text, char *)

XmPartOffset(Label, text)
```

Therefore, you must not have any spaces before or after the part (Label) parameter, as illustrated here:

```
XmField(w, offset,Label, text, char *)
```

Parameters

WidgetClass	Specifies the widget class pointer for the created widget.
Offset	Specifies the offset record.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/XmP.h

Related Information

The **XtOffset** subroutine, **XtVersionDontCheck** subroutine.

XmScaleGetValue Subroutine

Purpose

A **Scale** subroutine that returns the current slider position.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Scale.h>

void XmScaleGetValue(*Widget*, *ValueReturn***)**
Widget *Widget*;
int * *ValueReturn*;

Description

The **XmScaleGetValue** subroutine returns the current slider position value displayed in the scale.

Parameters

Widget	Specifies the **Scale** widget ID.
ValueReturn	Returns the current slider position value.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Scale.h

Related Information

The **XmScale** widget class.

XmScaleSetValue Subroutine

Purpose

A **Scale** subroutine that sets a slider value.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Scale.h>

void XmScaleSetValue(*Widget*, *Value***)**
Widget *Widget*;
int *Value*;

Description

The **XmScaleSetValue** subroutine sets the slider *Value* parameter within the **Scale** widget.

Parameters

Widget Specifies the **Scale** widget ID.

Value Specifies the slider position along the scale. This sets the **XmNvalue** resource.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Scale.h

Related Information

The **XmScale** widget class.

XmScrollBarGetValues Subroutine

Purpose

A **ScrollBar** subroutine that returns the **ScrollBar** increment values and changes the slider's size and position.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ScrollBar.h>

void XmScrollBarGetValues(*Widget*, *ValueReturn*, *SliderSizeReturn*, *IncrementReturn*, *PageIncrementReturn*)
Widget *Widget*;
int * *ValueReturn*;
int **SliderSizeReturn*;
int * *IncrementReturn*;
int * *PageIncrementReturn*;

Description

The **XmScrollBarGetValues** subroutine returns the **XmScrollBar** widget increment values and changes the slider size and position. The scroll region is overlaid with a slider bar that is adjusted in size and position using the main **ScrollBar** or set slider subroutine resources.

Parameters

Widget	Specifies the **ScrollBar** widget ID.
ValueReturn	Returns the **ScrollBar** slider position between the **XmNminimum** and **XmNmaximum** resources to the **ScrollBar** widget.
SliderSizeReturn	Returns the size of the slider as a value between zero and the absolute value of **XmNmaximum** minus **XmNminimum**. The size of the slider varies, depending on how much of the slider scroll area it represents.
IncrementReturn	Returns the amount of button increment and decrement.
PageIncrementReturn	Returns the amount of page increment and decrement.

Return Values

The **XmScrollBarGetValues** subroutine returns the **ScrollBar** increment values and changes the slider's size and position.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ScrollBar.h

Related Information

The **XmScrollBar** widget class.

XmScrollBarSetValues Subroutine

Purpose

A **ScrollBar** subroutine that changes **ScrollBar** increment values and the slider's size and position.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ScrollBar.h>

void XmScrollBarGetValues(*Widget*, *Value*, *SliderSize*
Increment, *PageIncrement*,
Notify)
Widget *Widget*;
int *Value*;
int *SliderSize*;
int *Increment*;
int *PageIncrement*;
Boolean *Notify*;

Description

The **XmScrollBarSetValues** subroutine changes the **ScrollBar** increment values and the slider's size and position. The scroll region is overlaid with a slider bar that is adjusted in size and position using the main **ScrollBar** or set slider subroutine resources.

Parameters

Widget Specifies the **ScrollBar** widget ID.

Value Specifies the **ScrollBar** slider position between the **XmNminimum** and **XmNmaximum**. The resource name associated with this parameter is **XmNvalue**.

SliderSize Specifies the size of the slider as a value between zero and the absolute value of **XmNmaximum** minus **XmNminimum**. The size of the slider varies, depending on how much of the slider scroll area it represents. This sets the **XmNsliderSize** resource associated with **ScrollBar**.

Increment Specifies the amount of button increment and decrement. If this parameter is not zero, the **ScrollBar** widget automatically adjusts the slider when an increment or decrement action occurs. This sets the **XmNincrement** resource associated with **ScrollBar**.

PageIncrement Specifies the amount of page increment and decrement. If this parameter is not zero, the **ScrollBar** widget automatically adjusts the slider when an increment or decrement action occurs. This sets the **XmNpageIncrement** resource associated with **ScrollBar**.

Notify Specifies a Boolean value that when **True**, indicates a change in the **ScrollBar** value and that the **ScrollBar** widget automatically activates

the **XmNvalueChangedCallback** with the recent change. If **False**, no change in the **ScrollBar** value has occurred, and **XmNvalueChangedCallback** is not activated.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ScrollBar.h

Related Information

The **XmScrollBar** widget class.

XmScrolledWindowSetAreas Subroutine

Purpose

A **ScrolledWindow** subroutine that adds or changes a window work region and a horizontal or vertical **ScrollBar** widget to the **ScrolledWindow** widget.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ScrolledW.h>

void XmScrolledWindowSetAreas(*Widget*, *HorizontalScrollbar*
VerticalScrollbar, *WorkRegion*)
Widget *Widget*;
Widget *HorizontalScrollbar*;
Widget *VerticalScrollbar*;
Widget *WorkRegion*;

Description

The **XmScrolledWindowSetAreas** subroutine adds or changes a window work region and a horizontal or vertical **ScrollBar** widget to the **ScrolledWindow** widget for the application. Each widget is optional and can be passed **NULL**.

Parameters

Widget	Specifies the **ScrolledWindow** widget ID.
HorizontalScrollbar	Specifies the **ScrollBar** widget ID for the horizontal **ScrollBar** to be associated with the **ScrolledWindow** widget. Set this ID only after creating an instance of the **ScrolledWindow** widget. The resource name associated with this parameter is **XmNhorizontalScrollBar**.
VerticalScrollbar	Specifies the **ScrollBar** widget ID for the vertical **ScrollBar** to be associated with the **ScrolledWindow** widget. Set this ID only after creating an instance of the **ScrolledWindow** widget. The resource name associated with this parameter is **XmNverticalScrollBar**.
WorkRegion	Specifies the widget ID for the work window to be associated with the **ScrolledWindow** widget. Set this ID only after creating an instance of the **ScrolledWindow** widget. The resource name associated with this parameter is **XmNworkWindow**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ScrolledW.h

Related Information

The **XmScrolledWindow** widget class.

XmSelectionBoxGetChild Subroutine

Purpose

A **SelectionBox** subroutine that is used to access a component.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/SelectioB.h>

Widget XmSelectionBoxGetChild(*Widget, Child***)**
Widget *Widget*;
unsigned char *Child*;

Description

The **XmSelectionBoxGetChild** subroutine is used to access a component within a **SelectionBox** widget. The parameters given to the subroutine are the **SelectionBox** widget and a value indicating which child to access.

Parameters

Widget Specifies the **SelectionBox** widget ID.

Child Specifies a component within the **SelectionBox** widget. The following are legal values for this parameter:

- **XmDIALOG_APPLY_BUTTON**
- **XmDIALOG_CANCEL_BUTTON**
- **XmDIALOG_DEFAULT_BUTTON**
- **XmDIALOG_HELP_BUTTON**
- **XmDIALOG_LIST**
- **XmDIALOG_LIST_LABEL**
- **XmDIALOG_OK_BUTTON**
- **XmDIALOG_SELECTION_LABEL**
- **XmDIALOG_SEPARATOR**
- **XmDIALOG_TEXT**
- **XmDIALOG_WORK_AREA**

Return Value

Returns the widget ID of the specified **SelectionBox** widget child.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/SelectioB.h

Related Information

The **XmSelectionBox** widget class.

XmSetFontUnit Subroutine

Purpose

Sets the font unit value for a display.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmSetFontUnit(*Display*, *FontUnitValue*)
Display *Display*;
int *FontUnitValue*;

Description

The **XmSetFontUnit** subroutine provides an external subroutine to initialize font unit values. Client applications may need to specify resolution–independent data based on a global font size. This subroutine must be called before any widgets with resolution–independent data are created. The **XmNunitType** resource descriptions in the **XmGadget**, **XmManager**, and **XmPrimitive** widgets have more information on resolution.

Parameters

Display	Defines the display for which this font value is to be applied.
FontUnitValue	Specifies the value to be used in the conversion calculations. The font unit value is normally taken as the **QUAD WIDTH** property of the font; however, the application can specify any integer value.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmConvertUnits** subroutine.

XmSetMenuCursor Subroutine

Purpose

A **RowColumn** subroutine that modifies the menu cursor for a client application.

Library

AIXwindows Library (**libXm.a**)

Syntax

void XmSetMenuCursor(*Display, CursorIdentification***)**
Display * *Display*;
Cursor *CursorIdentification*;

Description

The **XmSetMenuCursor** subroutine programmatically modifies the menu cursor for a client application; after the cursor is created by the client, this subroutine registers the cursor with the menu system. After calling this subroutine, the specified cursor is displayed whenever this client displays an **AIXwindows** menu on the indicated display. The client can then specify different cursors on different displays.

Parameters

Display	Specifies the display to which the cursor is to be associated.
CursorIdentification	Specifies the **X** cursor ID.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmRowColumn** widget class.

XmSetProtocolHooks Subroutine

Purpose

A **VendorShell** subroutine that executes pre– and post–actions when a protocol message is received from the Aixwindows window manager.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>
#include <X11/Protocols.h>

void XmSetProtocolHooks(*Shell, Property, Protocol,*
PreHook, PreClosure,
*PostHook, PostClosure***)**

Widget *Shell*;
Atom *Property*;
Atom *Protocol*;
XtCallbackProc *PreHook*;
caddr_t *PreClosure*;
XtCallbackProc *PostHook*;
caddr_t *PostClosure*;

Description

The **XmSetProtocolHooks** subroutine executes pre– and post–actions when a protocol message is received from the **mwm**. Since there is no guaranteed ordering in running event handlers or callback lists, this allows the shell to control the flow while leaving the protocol manager structures opaque.

Parameters

Shell	Specifies the widget with which the protocol property is associated.
Property	Specifies the protocol property.
Protocol	Specifies the protocol atom (or an **int** cast to **Atom**).
PreHook	Specifies the procedure to call before calling entries on the client callback list.
PreClosure	Specifies the client data to be passed to the *Prehook* parameter when it is invoked.
PostHook	Specifies the procedure to call after calling entries on the client callback list.
PostClosure	Specifies the client data to be passed to the *Posthook* parameter when it is invoked.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Files

/usr/include/Xm/Xm.h
/usr/include/X11/Protocols.h

Related Information

The **VendorShell** widget class, **XmInternAtom** subroutine.

XmString Subroutine

Purpose

The AIXwindows compound string type .

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/>

Description

The **AIXwindows Toolkit** provides a set of string subroutines that enable the creation and manipulation of compound strings and font lists. Compound strings are designed to allow encoding of textual data that is independent of underlying assumptions. Compound strings provide a means of separating the actual data that encodes the message and the resources that describe the representation of that data. The most obvious means of separating data from its graphic representations involves the mapping between character codes and display codes (possibly ASCII). In the past much of this information was simply assumed; compound strings make this information explicit. Compound strings allow description of the following resources of the actual text:

Character Set Identifier — Specifies the mapping between the string of bits that encode a character and the character being represented. For example, ASCII, ISO Latin1, or Kanji.

Direction — Defines the relationship between the logical (keystroke entry) order and the display order of the characters in a string. In English, the display order is left to right; as characters are typed, they are displayed left to right. In Hebrew, the display order is right to left; as characters are typed they are displayed right to left.

To perform output, the **AIXwindows Toolkit** needs more information than what is in the the compound string, namely, what X server font is to be used when displaying the string. The displaying widget must associate an X font with the character set of the string. This is done by supplying widgets with a resource not only for compound strings (which contain character set references), but another resource for a font list. The font list contains character set references matched to an X font. To display a compound string, the widget starts with the character set in the compound string. The widget font list is then searched for a matching character set; when found, the associated X font is used to display the compound string characters.

Compound String Types and Defines

A compound string is a stream of tag–length–value components; each component is one of the following:

Character Set Identifier — This is a **NULL** terminated sequence of octets that identifies the desired character set. This information is not interpreted by the **AIXwindows Toolkit**; it is only used to match a font with a compound string segment. As long as the application

is consistent in the naming of character sets in the compound string and the font list, the toolkit functions correctly. All text seen between two character set ID's are interpreted to be in the first set. It is an error for a text component to precede the first character set ID component.

There are times when a client application must create a string without knowing which character sets are available at the time the string is displayed. The **AIXwindows Toolkit** provides a special character set identifier that matches any available font. This universal character set is specified by the **XmSTRING_DEFAULT_CHARSET** identifier. If this identifier is used as the character set when a compound string is created, it matches the first font in the fontlist used to display the string, regardless of the character set associated with that particular font. By using the universal character set, an application can construct its strings so they are displayed in any font desired by the user, no matter what character set is associated with that font.

The universal character set can also be associated with a font in a font list. When the universal character set is used with a font, that font matches any string, no matter what the character set of the string. An application can construct a multiple–font fontlist and specify a default font to be used when no other font is matched.

Direction — This is a three–state value: left–to–right, right–to–left and revert. Like the character set ID, it has persistence. The default direction is left–to–right: text components preceding the first direction component are assigned a direction of left–to–right.

Text — This is the octet string of the actual character data. There are no semantics for any octets. Characters like /n do not have any meaning. As a convenience, the **XmStringLtoRCreate** subroutine imposes the left–to–right semantic.

Separator — This is a "marked" tag with no value. It allows an array of compound string segments to be presented as a single entity.

Compound String Devices

typedef unsigned char XmStringDirection /* an enumerated type */

The set of possible values for this type are:
XmSTRING_DIRECTION_L_TO_R
XmSTRING_DIRECTION_R_TO_L
XmSTRING_DIRECTION_REVERT

typedef char * XmStringCharSet /* octet chars, null terminated */

typedef char * XmString /* opaque to users */

```
typedef unsigned char XmStringComponentType /* component
 tag types */
```

The set of currently possible values for this type are:

```
#define XmSTRING_COMPONENT_UNKNOWN 0
#define XmSTRING_COMPONENT_CHARSET 1
#define XmSTRING_COMPONENT_TEXT 2
#define XmSTRING_COMPONENT_DIRECTION 3
#define XmSTRING_COMPONENT_SEPARATOR 4
#define XmSTRING_COMPONENT_END 126 /* no more
 components */
#define XmSTRING_DEFAULT_CHARSET (–1) /* The
 universal character set */
```

Subroutines

XmFontListCreate, **XmFontListFree**, **XmStringBaseline**, **XmStringByteCompare**, **XmStringCompare**, **XmStringConcat**, **XmStringCopy**, **XmStringCreate**, **XmStringCreateLtoR**, **XmStringDirectionCreate**, **XmStringDraw**, **XmStringDrawImage**, **XmStringDrawUnderline**, **XmStringEmpty**, **XmStringExtent**, **XmStringFree**, **XmStringFreeContext**, **XmStringGetLtoR**, **XmStringGetNextComponent**, **XmStringGetNextSegment**, **XmStringHeight**, **XmStringInitContext**, **XmStringLength**, **XmStringLineCount**, **XmStringNConcat**, **XmStringNCopy**, **XmStringPeekNextComponent**, **XmStringSegmentCreate**, **XmStringSeparatorCreate**, **XmStringWidth**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmFontListAdd** subroutine, **XmFontListCreate** subroutine, **XmFontListFree** subroutine, **XmStringBaseline** subroutine, **XmStringByteCompare** subroutine, **XmStringCompare** subroutine, **XmStringConcat** subroutine, **XmStringCopy** subroutine, **XmStringCreate** subroutine, **XmStringCreateLtoR** subroutine, **XmStringDirectionCreate** subroutine, **XmStringDraw** subroutine, **XmStringDrawImage** subroutine, **XmStringDrawUnderline** subroutine, **XmStringEmpty** subroutine, **XmStringExtent** subroutine, **XmStringFree** subroutine, **XmStringFreeContext** subroutine, **XmStringGetLtoR** subroutine, **XmStringGetNextComponent** subroutine, **XmStringGetNextSegment** subroutine, **XmStringHeight** subroutine, **XmStringInitContext** subroutine, **XmStringLength** subroutine, **XmStringLineCount** subroutine, **XmStringNConcat** subroutine, **XmStringNCopy** subroutine, **XmStringPeekNextComponent** subroutine, **XmStringSegmentCreate** subroutine, **XmStringSeparatorCreate** subroutine, **XmStringWidth** subroutine.

XmStringBaseline Subroutine

Purpose

A compound string subroutine that returns the number of pixels between the top of the character box and the baseline of the first line of text.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Dimension XmStringBaseline(*FontList*, *String***)**
XmFontList *FontList*;
XmString *String*;

Description

The **XmStringBaseline** subroutine returns the number of pixels between the top of the character box and the baseline of the first line of text in the provided compound string.

Parameters

FontList Specifies the **FontList** widget.

String Specifies the string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringByteCompare Subroutine

Purpose

A compound string subroutine that indicates the results of a byte–by–byte comparison.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmStringByteCompare(*String1*, *String2***)**
XmString *String1*;
XmString *String2*;

Description

The **XmStringByteCompare** subroutine returns a Boolean value indicating the results of a byte–by–byte comparison of two compound strings. In some cases, once a compound string is put into a widget, that string is converted into an internal form to allow faster processing. Part of the conversion process strips out unnecessary or redundant information. If an application then invokes an **XtGetValues** subroutine to retrieve a compound string from a widget (specifically, a **Label** widget with all of its subclasses), the compound string returned might not be the byte–by–byte equivalent of the original string given to the widget.

Parameters

String1 Specifies a compound string to be compared with *String2* parameter.

String2 Specifies a compound string to be compared with *String1* parameter.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringCompare Subroutine

Purpose

A compound string subroutine that compares two strings.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmStringCompare(*String1*, *String2*)
XmString *String1*;
XmString *String2*;

Description

The **XmStringCompare** subroutine returns a Boolean value indicating the results of a semantically equivalent comparison of two compound strings.

"Semantically equivalent" means that the strings have the same text components, directions, and separators. If character sets are specified, they must be equal as well. If either one has a character set of **XmSTRING_DEFAULT_CHARSET**, it matches the other character set.

Parameters

String1 Specifies a compound string to be compared with *String2* parameter.

String2 Specifies a compound string to be compared with *String1* parameter.

Return Value

Returns **True** if two compound strings are equivalent.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringConcat Subroutine

Purpose

A compound string subroutine that appends one string to another.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringConcat (*String1, String2*)
XmString *String1;*
XmString *String2*;

Description

The **XmStringConcat** subroutine appends a *String2* parameter to the end of a *String1* parameter and returns the resulting compound string. The original strings are preserved. The space for the resulting compound string is allocated within the subroutine. After using this subroutine, free this space by calling the **XmStringFree** convenience subroutine.

Parameters

String1 Specifies the compound string to which a copy of *String2* parameter is appended.

String2 Specifies the compound string that is appended to the end of *String1* parameter.

Return Value

Returns a new compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringFree** subroutine.

XmStringCopy Subroutine

Purpose

A compound string subroutine that makes a copy of a string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringCopy(*String1***)**
XmString *String1*;

Description

The **XmStringCopy** subroutine makes a copy of a compound string. The space for the resulting compound string is allocated within the subroutine. The application is responsible for managing the allocated space. The memory can be recovered by calling the **XmStringFree** subroutine.

Parameter

String1 Specifies the compound string.

Return Value

Returns a new compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringFree** subroutine.

XmStringCreate Subroutine

Purpose

A compound string subroutine that creates a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringCreate(*Text, CharacterSet***)**
**char ****Text*;
XmStringCharSet *CharacterSet*;

Description

The **XmStringCreate** subroutine creates a compound string with two components: text and a character set.

Parameters

Text	Specifies a pointer to a null–terminated string.
CharacterSet	Specifies the character set identifier to be associated with the given text. This can be **XmSTRING_DEFAULT_CHARSET**.

Return Value

Returns a new compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmFontListAdd** subroutine, **XmFontListCreate** subroutine, **XmFontListFree** subroutine, **XmStringBaseline** subroutine, **XmStringByteCompare** subroutine, **XmStringCompare** subroutine, **XmStringConcat** subroutine, **XmStringCopy** subroutine, **XmStringCreateLtoR** subroutine, **XmStringDirectionCreate** subroutine, **XmStringDraw** subroutine, **XmStringDrawImage** subroutine, **XmStringDrawUnderline** subroutine, **XmStringEmpty** subroutine, **XmStringExtent** subroutine, **XmStringFree** subroutine, **XmStringFreeContext** subroutine, **XmStringGetLtoR** subroutine, **XmStringGetNextComponent** subroutine, **XmStringGetNextSegment** subroutine, **XmStringHeight** subroutine, **XmStringInitContext** subroutine, **XmStringLength** subroutine, **XmStringLineCount** subroutine, **XmStringNConcat** subroutine, **XmStringNCopy** subroutine, **XmStringPeekNextComponent** subroutine, **XmStringSegmentCreate** subroutine, **XmStringSeparatorCreate** subroutine, **XmStringWidth** subroutine.

XmStringCreateLtoR Subroutine

Purpose

A compound string subroutine that creates a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringCreateLtoR(*Text*, *CharacterSet***)**
char **Text*;
XmStringCharSet *CharacterSet*;

Description

The **XmStringCreateLtoR** subroutine creates a compound string with two components: text and a character set. This subroutine imposes the semantic of scanning for \n characters in the text. When one is found, the text up to that point is put into a segment followed by separator component. No final separator component is appended to the end of the compound string. The direction defaults to left–to–right. This subroutine assumes that the encoding is single octet rather than double octet per character of text.

Parameters

Text	Specifies a pointer to a null–terminated string.
CharacterSet	Specifies the character set identifier to be associated with the given text. This can be **XmSTRING_DEFAULT_CHARSET**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringDirectionCreate Subroutine

Purpose

A compound string subroutine that creates a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringDirectionCreate (*Direction***)**
XmStringDirection *Direction*;

Description

The **XmStringDirectionCreate** subroutine creates a compound string with a single component, a direction with the given value.

Parameter

Direction Specifies the value of the directional component.

Return Value

Returns a new compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringDraw Subroutine

Purpose

A compound string subroutine that draws a compound string in an Enhanced X–Windows window.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmStringDraw(*Display, Window, FontList,*
String, GraphicsContext,
X, Y, Width, Alignment
*LayoutDirection, Clip***)**

Display **Display;*
Window *Window;*
XmFontList *FontList;*
XmString *String;*
GC *GraphicsContext;*
Position *X;*
Position *Y;*
Dimension *Width;*
Byte *Alignment;*
Byte *LayoutDirection;*
XRectangle **Clip;*

Description

The **XmStringDraw** subroutine draws a compound string in an X Window.

Parameters

Display	Specifies the display.
Window	Specifies the window.
Fontlist	Specifies the *FontList* parameter.
String	Specifies the string.
GraphicsContext	Specifies the graphics context to use.
X	Specifies a coordinate of the rectangle that contains the displayed compound string.
Y	Specifies a coordinate of the rectangle that contains the displayed compound string.
Width	Specifies the width of the rectangle that is to contain the displayed compound string.

Alignment	Specifies how the string is to be aligned within the specified rectangle. It is either the **XmALIGNMENT_BEGINNING**, **XmALIGNMENT_CENTER**, or **XmALIGNMENT_END** value.
LayoutDirection	Controls the direction in which the segments of the compound string are to be laid out. It also determines the meaning of the *Alignment* parameter.
Clip	Allows the application to restrict the area into which the compound string is to be drawn. If **NULL**, no clipping is done.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringDrawImage Subroutine

Purpose

A compound string subroutine that draws a compound string in an Enhanced X–Windows window and creates an image.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmStringDrawImage(*Display, Window, Fontlist, String, GraphicsContext, X, Y, Width, Alignment, LayoutDirection, Clip*)

Display **Display;*
Window *Window;*
XmFontList *Fontlist;*
XmString *String;*
GC *GraphicsContext;*
Position *X;*
Position *Y;*
Dimension *Width;*
Byte *Alignment;*
Byte *LayoutDirection;*
XRectangle **Clip;*

Description

The **XmStringDrawImage** subroutine draws a compound string in an Enhanced X–Windows window and paints both the foreground and background bits of each character.

Parameters

Display	Specifies the display.
Window	Specifies the window.
Fontlist	Specifies the *FontList* parameter.
String	Specifies the string.
GraphicsContext	Specifies the graphics context to use.
X	Specifies a coordinate of the rectangle that contains the displayed compound string.
Y	Specifies a coordinate of the rectangle that contains the displayed compound string.
Width	Specifies the width of the rectangle that contains the displayed compound string.

Alignment	Specifies how the string is aligned within the specified rectangle. It is either the **XmALIGNMENT_BEGINNING**, **XmALIGNMENT_CENTER**, or **XmALIGNMENT_END** value.
LayoutDirection	Controls the direction in which the segments of the compound string are laid out. It also determines the meaning of the *Alignment* parameter.
Clip	Allows the application to restrict the area into which the compound string is drawn. If **NULL**, no clipping is done.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringDrawUnderline Subroutine

Purpose

A compound string subroutine that underlines a string drawn in an X Window.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmStringDrawUnderline(*Display, Window, Fontlist, String, GraphicsContext, X, Y, Width, Alignment, LayoutDirection, Clip, Underline***)**

Display **Display*;*
Window *Window*;
XmFontList *FontList*;
XmString *String*;
GC *GraphicsContext*;
Position *X*;
Position *Y*;
Dimension *Width*;
Byte *Alignment*;
Byte *LayoutDirection*;
XRectangle **Clip*;
XmString *Underline*;

Description

The **XmStringDrawUnderline** subroutine draws a compound string in an X Window. If the substring identified by the *Underline* parameter can be matched in the *String* parameter, the substring is underlined. Once a match has occurred, no further matches or underlining is be done.

Parameters

Display	Specifies the display.
Window	Specifies the window.
FontList	Specifies the font list.
String	Specifies the string.
GraphicsContext	Specifies the graphics context to use.
X	Specifies a coordinate of the rectangle that contains the displayed compound string.
Y	Specifies a coordinate of the rectangle that contains the displayed compound string.

Width	Specifies the width of the rectangle that is to contain the displayed compound string.
Alignment	Specifies how the string is to be aligned within the specified rectangle. It is either the **XmALIGNMENT_BEGINNING**, **XmALIGNMENT_CENTER**, or **XmALIGNMENT_END** value.
LayoutDirection	Controls the direction in which the segments of the compound string are to be laid out. It also determines the meaning of the *Alignment* parameter.
Clip	Allows the application to restrict the area into which the compound string is to be drawn. If **NULL**, no clipping is done.
Underline	Specifies the substring to be underlined.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringEmpty Subroutine

Purpose

A compound string subroutine that provides information on the existence of non–zero length text components.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmStringEmpty(*String1***)**
XmString *String1;*

Description

The **XmStringEmpty** subroutine returns a Boolean value indicating whether or not any non–zero text components exist in the provided compound string. It returns **True** if there are no text segments in the string. If this routine is passed **NULL** as the string, it returns **True.**

Parameter

String1 Specifies the compound string.

Return Value

Returns **True** if there are no text segments in the string. If this routine is passed **NULL** as the string, it returns **True.**

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringExtent Subroutine

Purpose

A compound string subroutine that determines the size of the smallest rectangle that encloses the compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmStringExtent(*FontList, String, Width, Height***)**
XmFontList *FontList*;
XmString *String*;
Dimension *Width*;
Dimension *Height*;

Description

The **XmStringExtent** subroutine determines the width and height, in pixels, of the smallest rectangle that encloses the provided compound string.

Parameters

FontList	Specifies the *FontList* parameter.
String	Specifies the string.
Width	Specifies the width of the rectangle.
Height	Specifies the height of the rectangle.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringFree Subroutine

Purpose

A compound string subroutine that recovers memory.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmStringFree(*String***)**
XmString *String;*

Description

The **XmStringFree** subroutine recovers memory used by a compound string.

Parameter

String Specifies the compound string to be freed.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringFreeContext Subroutine

Purpose

A compound string subroutine that instructs the **AIXwindowsToolkit** program that the context is no longer needed.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

void XmStringFreeContext(*Context***)**
XmStringContext **Context*;*

Description

The **XmStringFreeContext** subroutine instructs the **AIXwindows Toolkit** program that the context is no longer needed and is not to be used without reinitialization.

Parameter

Context Specifies the string context structure that was allocated by the **XmStringInitContext** subroutine.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringInitContext** subroutine.

XmStringGetLtoR Subroutine

Purpose

A compound string subroutine that searches the input compound string for a text segment.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmStringGetLtoR(*String, CharacterSet, Text***)**
XmString *String*;
XmStringCharSet *CharacterSet*;
Char ** *Text;*

Description

The **XmStringGetLtoR** subroutine searches the input compound string for a segment that matches the given character set identifier.

Parameters

String	Specifies the compound string.
CharacterSet	Specifies the character set identifier to be associated with the text. This can be **XmSTRING_DEFAULT_CHARSET**.
Text	Specifies a pointer to a sequence of null terminated strings. Note that this parameter is a double pointer to **Char**.

Return Value

Returns **True** if the matching text segment can be found. On return, the *Text* parameter will have a null terminated octet sequence containing the matched segment.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringGetNextComponent Subroutine

Purpose

A compound string subroutine that returns the type and value of the next component in the compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmStringComponentType XmStringGetNextComponent(*Context*, *Text*, *CharacterSet*, *Direction*, *UnknownTag*, *UnknownLength*, *UnknownValue*)

XmStringContext *Context*;
char ***Text*;
XmStringCharSet **CharacterSet*;
XmStringDirection **Direction*;
XmStringComponentType **UnknownTag*;
short **UnknownLength*;
char ***UnknownValue*;

Description

The **XmStringGetNextComponent** subroutine returns the type and value of the next component in the compound string identified by the *Context* parameter. It is a low–level component subroutine. Components are returned one at a time. On return, only some output parameters are valid; valid parameters can be determined by examining the return status. In the case of *Text, CharacterSet* or *Direction* parameter components, only one output parameter is valid. If the return status indicates that an unknown component was encountered, the *Tag, Length*, and *Value* parameters are returned. The subroutine allocates the space necessary to hold returned values; freeing this space is the caller's responsibility.

Parameters

Context	Specifies the string context structure that was allocated by the **XmStringInitContext** subroutine.
Text	Specifies a pointer to a null terminated string.
CharacterSet	Specifies the character set identifier to be associated with the text. This can be **XmSTRING_DEFAULT_CHARSET**.
Direction	Specifies the direction of the text.
UnknownTag	Specifies the tag of an unknown component.
UnknownLength	Specifies the length of an unknown component.
UnknownValue	Specifies the value of an unknown component.

Return Value

Returns the type of component found.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringInitContext** subroutine.

XmStringGetNextSegment Subroutine

Purpose

A compound string subroutine that fetches the octets in the next segment of a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmStringGetNextSegment(*Context, Text, CharacterSet, Direction, Separator***)**
XmStringContext *Context*;
char ***Text*;
XmStringCharSet **CharacterSet*;
XmStringDirection **Direction*;
Boolean **Separator*;

Description

The **XmStringGetNextSegment** subroutine fetches the octets in the next segment of a compound string; repeated calls fetch sequential segments. The *Text, CharacterSet*, and *Direction* parameters of the fetched segment are returned each time. A Boolean value is returned to indicate whether a valid segment was successfully parsed.

Parameters

Context	Specifies the string context structure that was allocated by the **XmStringInitContext** subroutine.
Text	Specifies a pointer to a null terminated string.
CharacterSet	Specifies the character set identifier to be associated with the text. This can be **XmSTRING_DEFAULT_CHARSET**.
Direction	Specifies the direction of the text.
Separator	Specifies if the next component of the compound string is a separator.

Return Value

Returns **True** if a valid segment is found.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringInitContext** subroutine, **XtMalloc** subroutine.

XmStringHeight Subroutine

Purpose

A compound string subroutine that returns the line height of the given compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Dimension XmStringHeight(*FontList, String***)**
XmFontList *FontList*;
XmString *String;*

Description

The **XmStringHeight** subroutine returns the height, in pixels, of the sum of all the line heights of the given compound string. Separator components delimit lines.

Parameters

FontList Specifies the *FontList* parameter.

String Specifies the string.

Return Value

Returns the height of the specified string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringInitContext Subroutine

Purpose

A compound string subroutine that allows client applications to read out the content segment by segment, or component by component.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmStringInitContext(*Context, String***)**
XmStringContext ***Context**;
XmString *String;*

Description

The **XmStringInitContext** subroutine maintains a context to allow client applications to read out the contents of a compound string segment by segment. This subroutine establishes the context for this read–out. A Boolean value is returned to indicate if the input string is able to be parsed.

Parameters

Context Specifies a pointer to the allocated context.

String Specifies the string.

Return Value

Returns **True** if the context was allocated.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XtMalloc** subroutine.

XmStringLength Subroutine

Purpose

A compound string subroutine that obtains the length of a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

int XmStringLength(*String1***)**
XmString *String1*;

Description

The **XmStringLength** subroutine obtains the length of a compound string. This subroutine returns the number of bytes in the *String1* parameter, including all tags, direction indicators, and separators. If the compound string has an invalid structure, **zero** is returned.

Parameter

String1 Specifies the compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringLineCount Subroutine

Purpose

A compound string subroutine that returns the number of separators plus one in the provided compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

int XmStringLineCount(*String***)**
XmString *String*;

Description

The **XmStringLineCount** subroutine returns the number of separators plus one in the provided compound string. In effect, it counts the number of lines of text. For compound strings that have separators not representing /n, this subroutine gives incorrect results; not every separator is /n. This subroutine only effectually counts the number of lines if every separator is a new–line character.

Parameter

String Specifies the string.

Return Value

Returns the number of lines in the compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringNConcat Subroutine

Purpose

A compound string subroutine that appends a specified number of bytes to a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringNConcat(*String1, String2, NumberBytes***)**
XmString *String1*;
XmString *String2*;
int *NumberBytes*;

Description

The **XmStringNConcat** subroutine appends a specified number of bytes from the *String2* parameter to the end of the *String1* parameter, including tags, directional indicators, and separators. This subroutine then returns the resulting compound string. The original strings are preserved. The space for the resulting compound string is allocated within the subroutine. The client application is responsible for managing the allocated space. The memory can be recovered by calling the **XmStringFree** subroutine.

Parameters

String1	Specifies the compound string to which a copy of the *String2* parameter is appended.
String2	Specifies the compound string that is appended to the end of the *String1* parameter.
NumberBytes	Specifies the number of bytes of the *String2* parameter to append to the *String1* parameter. If this value is more than the length of the *String2* parameter, the resulting string is not a valid compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringFree** subroutine.

XmStringNCopy Subroutine

Purpose

A compound string subroutine that creates a copy of a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringNCopy(*String1, NumberBytes***)**
XmString *String1*;
int *NumberBytes*;

Description

The **XmStringNCopy** subroutine creates a copy of the *String1* parameter that contains a specified number of bytes, including tags, directional indicators, and separators. This subroutine then returns the resulting compound string. The original string is preserved. The space for the resulting compound string is allocated within the subroutine. The application is responsible for managing the allocated space. The allocated memory can be recovered by calling the **XmStringFree** subroutine.

Parameters

String1	Specifies the compound string.
NumberBytes	Specifies the number of bytes of the *String1* parameter to copy. If this value is less than the length of the *String1* parameter, the resulting string is not a valid compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringFree** subroutine.

XmStringPeekNextComponent Subroutine

Purpose

A compound string subroutine that returns the component type of the next component fetched.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmStringComponentType XmStringPeekNextComponent(*Context***)**
XmStringContext ****Context*;**

Description

The **XmStringPeekNextComponent** subroutine examines the next component that is to be fetched by the **XmStringGetNextComponent** subroutine and returns the component type.

Parameter

Context Specifies the string context structure that was allocated by the **XmStringInitContext** subroutine.

Return Value

Returns the type of component found.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XmStringInitContext** subroutine.

XmStringSegmentCreate Subroutine

Purpose

A compound string subroutine that creates a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringSegmentCreate(*Text, CharacterSet, Direction, Separator***)**
char * *Text*;
XmStringCharSet *CharacterSet*;
XmStringDirection *Direction*;
Boolean *Separator*;

Description

The **XmStringSegmentCreate** subroutine is a high level subroutine that assembles a compound string consisting of a character set identifier, a direction component, a text component, and an optional separator component.

Parameters

Text	Specifies a pointer to a null–terminated string.
CharacterSet	Specifies the character set identifier to be associated with the text. This can be **XmSTRING_DEFAULT_CHARSET**.
Direction	Specifies the direction of the text.
Separator	Specifies separator addition. If **False**, the compound string does not have a separator at the end. If **True**, a separator immediately follows the text component.

Return Value

Returns a new compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmStringSeparatorCreate Subroutine

Purpose

A compound string subroutine that creates one single compound string, a separator.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

XmString XmStringSeparatorCreate()

Description

The **XmStringSeparatorCreate** subroutine creates a compound string with a separator as its only component. Possible usages might be for compound strings concatenations into compound strings that need separators interspersed in between segments.

Return Value

Returns a new compound string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine, **XtManageChild** subroutine, **XtUnmanageChild** subroutine.

XmStringWidth Subroutine

Purpose

A compound string subroutine that returns the width of the longest sequence of text components in a compound string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Dimension XmStringWidth(*FontList*, *String***)**
XmFontList *FontList*;
XmString *String*;

Description

The **XmStringWidth** subroutine returns the width, in pixels, of the longest sequence of text components in the provided compound string. Separator components are used to delimit sequences of text components.

Parameters

FontList	Specifies the font list.
String	Specifies the string.

Return Value

Returns a pixel value of type 'Dimension'.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmStringCreate** subroutine.

XmTextClearSelection Subroutine

Purpose

A **Text** subroutine that clears the primary selection.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

void XmTextClearSelection(*Widget, Time***)**
Widget *Widget*;
Time *Time*;

Description

The **XmTextClearSelection** subroutine clears the primary selection in the **Text** widget; it has no effect on previously selected text.

Parameters

Widget	Specifies the **Text** widget ID.
Time	Specifies the time at which the selection value is desired. This should be the time of the event that triggered this request.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class.

XmTextGetEditable Subroutine

Purpose

A **Text** subroutine that accesses the edit permission state.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

Boolean XmTextGetEditable(*Widget***)**
Widget *Widget*;

Description

The **XmTextGetEditable** subroutine accesses the edit permission state of the **Text** widget.

Parameter

Widget Specifies the **Text** widget ID.

Return Value

Returns a Boolean value that indicates the state of the **XmNeditable** resource.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class.

XmTextGetMaxLength Subroutine

Purpose

A **Text** subroutine that accesses the value of the current maximum allowable length of the text string entered from the keyboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

int XmTextGetMaxLength(*Widget***)**
Widget *Widget*;

Description

The **XmTextGetMaxLength** subroutine accesses the value of the current maximum allowable length of the text string in the **Text** widget entered from the keyboard. The maximum allowable length prevents the user from entering a text string larger than this limit.

Parameter

Widget Specifies the **Text** widget ID.

Return Values

Returns the integer value that indicates the maximum allowable string length that can be entered from the keyboard.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class.

XmTextGetSelection Subroutine

Purpose

A **Text** subroutine that retrieves the value of the primary selection.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

char * XmTextGetSelection(*Widget***)**
Widget *Widget*;

Description

The **XmTextGetSelection** subroutine retrieves the value of the primary selection. This subroutine returns a **NULL** pointer if no text is selected in the widget. The client application is responsible for freeing the storage associated with the string by calling the **XtFree** subroutine.

Parameter

Widget Specifies the **Text** widget ID.

Return Values

Returns a character pointer to the string that is associated with the primary selection.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class, **XtFree** subroutine.

XmTextGetString Subroutine

Purpose

A **Text** subroutine that accesses the string value.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

char * XmTextGetString (*Widget***)**
Widget *Widget*;

Description

The **XmTextGetString** subroutine accesses the string value of the **Text** widget. The application is responsible for freeing the storage associated with the string by calling the **XtFree** subroutine.

Parameter

Widget Specifies the **Text** widget ID.

Return Value

Returns a character pointer to the string value of the **Text** widget

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class, **XtFree** subroutine.

XmTextReplace Subroutine

Purpose

A **Text** subroutine that replaces part of the text string.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

void XmTextReplace (*Widget, FromPosition, ToPosition, Value*)
Widget *Widget*;
int *FromPosition*;
int *ToPosition*;
char **Value*;

Description

The **XmTextReplace** subroutine replaces part of the text string in the **Text** widget. The character positions begin at zero and are numbered sequentially from the beginning of the text.

For example, a text replacement might require replacement of the second and third characters in the text string. To accomplish this, the parameter *FromPosition* must be a value of 1 and the parameter *ToPosition* must be a value of 3. To insert a string after the fourth character, the parameters *FromPosition* and *ToPosition* must both be a value of 4.

Parameters

Widget	Specifies the **Text** widget ID.
FromPosition	Specifies the start position of the text to be replaced.
ToPosition	Specifies the end position of the text to be replaced.
Value	Specifies the character string value to be added to the **Text** widget.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class.

XmTextSetEditable Subroutine

Purpose

A **Text** subroutine that sets the edit permission.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

void XmTextSetEditable (*Widget, Editable*)
Widget *Widget*;
Boolean *Editable*;

Description

The **XmTextSetEditable** subroutine sets the edit permission state of the **Text** widget. When the edit permission state is set to **True**, the text string can be edited. This subroutine changes the **XmNeditable** resource of the **Text** widget.

Parameters

Widget Specifies the **Text** widget ID.

Editable Specifies a Boolean value that, when **True**, allows text string edits.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class.

XmTextSetMaxLength Subroutine

Purpose

A **Text** subroutine that sets the value of the current maximum allowable length of the text string entered from the keyboard.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

void XmTextSetMaxLength (*Widget, MaxLength*)
Widget *Widget*;
int *MaxLength*;

Description

The **XmTextSetMaxLength** subroutine sets the value of the current maximum allowable length of the text string in the **Text** widget. The maximum allowable length prevents the user from entering a text string from the keyboard that is larger than this limit. This subroutine changes the **XmNmaxLength** resource of the **Text** widget. Thus, as applicable to that particular resource, this subroutine has no effect when strings are entered using the **XmNvalue** resource or the **XmTextSetString** subroutine call.

Parameters

Widget	Specifies the **Text** widget ID.
MaxLength	Specifies the maximum allowable length of the text string.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class, **XmTextSetString** subroutine.

XmTextSetSelection Subroutine

Purpose

A **Text** subroutine that sets the primary selection of the text.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

void XmTextSetSelection (*Widget, First, Last, Time*)
Widget *Widget*;
int *First*;
int *Last*;
Time *Time*;

Description

The **XmTextSetSelection** subroutine sets the primary selection of the text in the widget.

Parameters

Widget	Specifies the **Text** Widget ID.
First	Marks the first character position.
Last	Marks the last position of the text to be selected.
Time	Specifies the time at which the selection value is desired. This should be the same as the time of the event that triggered this request.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class.

XmTextSetString Subroutine

Purpose

A **Text** subroutine that sets the string value.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Text.h>

void XmTextSetString (*Widget, Value*)
Widget *Widget*;
char **Value*;

Description

The **XmTextSetString** subroutine sets the string value of the **Text** widget.

Parameters

Widget	Specifies the **Text** widget ID.
Value	Specifies the character pointer to the string value and places the string into the text edit window.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Text.h

Related Information

The **XmText** widget class.

XmToggleButtonGadgetGetState Subroutine

Purpose

A **ToggleButtonGadget** subroutine that obtains the state of a **ToggleButtonGadget** gadget.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleBG.h>

Boolean XmToggleButtonGadgetGetState(*Widget***)**
Widget *Widget*;

Description

The **XmToggleButtonGadgetGetState** subroutine obtains the state of an **ToggleButtonGadget** gadget.

Parameter

Widget Specifies the **ToggleButtonGadget** gadget.

Return Value

Returns **True** if the button is selected and **False** if the button is unselected.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleBG.h

Related Information

The **XmToggleButtonGadget** gadget class.

XmToggleButtonGadgetSetState Subroutine

Purpose

A **ToggleButtonGadget** subroutine that sets or changes the current state.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleBG.h>

void XmToggleButtonGadgetSetState(*Widget, State, Notify***)**
Widget *Widget*;
Boolean *State*;
Boolean *Notify*;

Description

The **XmToggleButtonGadgetSetState** subroutine sets or changes the current state of a **ToggleButtonGadget** gadget.

Parameters

Widget	Specifies the **ToggleButtonGadget** gadget.
State	Specifies a Boolean value that indicates whether the **ToggleButtonGadget** gadget state is on or off. If **True**, the button state is selected; if **False**, the button state is unselected.
Notify	Indicates whether the subroutines specified in the **XmNvalueChangedCallback** resource are called; it can be either **True** or **False**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleBG.h

Related Information

The **XmToggleButtonGadget** gadget class.

XmToggleButtonGetState Subroutine

Purpose

A **ToggleButton** subroutine that obtains the state of a **ToggleButton** widget.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleB.h>

Boolean XmToggleButtonGetState(*Widget***)**
Widget *Widget;*

Description

The **XmToggleButtonGetState** subroutine obtains the state of a **ToggleButton** widget.

Parameters

Widget Specifies the **ToggleButton** widget ID.

Return Value

The **XmToggleButtonGetState** subroutine returns **True** if the button is selected and **False** if the button is unselected.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleB.h

Related Information

The **XmToggleButton** widget class.

XmToggleButtonSetState Subroutine

Purpose

A **ToggleButton** subroutine that sets or changes the current state.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/ToggleB.h>

void XmToggleButtonSetState(*Widget, State, Notify***)**
Widget *Widget;*
Boolean *State;*
Boolean *Notify*;

Description

The **XmToggleButtonSetState** subroutine sets or changes the current state of the **ToggleButton** widget.

Parameters

Widget	Specifies the **ToggleButton** widget ID.
State	Specifies a Boolean value that indicates whether the **ToggleButton** state is selected or unselected. If **True**, the button state is selected; if **False**, the button state is unselected.
Notify	Indicates whether the subroutines specified in the **XmNvalueChangedCallback** resource are called; it can be either **True** or **False**.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/ToggleB.h

Related Information

The **XmToggleButton** widget class.

XmUninstallImage Subroutine

Purpose

A pixmap–caching subroutine that removes an image from the image cache.

Library

AIXwindows Library (**libXm.a**)

Syntax

#include <Xm/Xm.h>

Boolean XmUninstallImage(*Image***)**
XImage ***Image**;

Description

The **XmUninstallImage** subroutine removes an image from an image cache.

Parameter

Image	Points to the image structure given to the **XmInstallImage** subroutine.

Return Values

The **XmUninstallImage** subroutine returns a **True** value when successful. A value of **False** is returned if the image is a **NULL** value, or if it cannot be found to be uninstalled.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

File

/usr/include/Xm/Xm.h

Related Information

The **XmInstallImage** subroutine, **XmGetPixmap** subroutine, **XmDestroyPixmap** subroutine.

XmUpdateDisplay Subroutine

Purpose

Processes all pending exposure events immediately.

Library

AIXwindows Library (**libXm.a**)

Syntax

Widget XmUpdateDisplay(*Widget***)**
Widget *Widget*;

Description

The **XmUpdateDisplay** subroutine provides a mechanism for forcing all pending exposure events to be removed from the input queue and to be processed immediately.

When a user selects a button within a **MenuPane** widget, the **MenuPane** widget is unposted and then any activation callback routines registered by the application is invoked. If one of the callbacks performs a time consuming action, the portion of the application window that is covered by the **MenuPane** widget is not redrawn; normal exposure processing does not occur until all of the callback routines are invoked. If the user suspects that a callback routine takes a long time, the callback may choose to invoke the **XmUpdateDisplay** subroutine before starting its time consuming operation.

This subroutine is also useful any time a transient window is unposted (such as a dialog box) and callbacks are invoked before normal exposure processing can occur.

Parameter

Widget Specifies any widget or gadget.

Implementation Specifics

This subroutine is part of AIXwindows Development Environment in AIXwindows Environment/6000.

Related Information

The **XmCreatePopupMenu** subroutine, **XmCreatePulldownMenu** subroutine.

AIXwindows Resource Sets

ApplicationShell Resource Set

XmNargc **XmNargv**

XmNargc

Class	**XmCNargc**
Type	**int**
Default	**NULL**
Access	CSG

Description

The **XmNargc** resource specifies the number of parameters in the **XmNargv** resource. The **XtInitialize** subroutine sets this resource on the **Shell** widget instance that it creates by using its parameters as the values.

XmNargv

Class	**XmCNargv**
Type	**String ***
Default	**NULL**
Access	CSG

Description

The **XmNargv** resource specifies the parameter list required by a session manager to restart the application, should it be killed. This list should be updated at appropriate points by the application if a new state is reached that can be directly restarted. The **XtInitialize** subroutine sets this resource and the **XmNargc** resource to the **Shell** widget instance it creates, to the parameter list it is passed.

Composite Resource Set

XmNinsertPosition

XmNinsertPosition

Class	**XmCInsertPosition**
Type	**XmRFunction**
Default	**NULL**
Access	CSG

Description

The **XminsertPosition** resource points to the **XtOrderProc** data type.

The following procedure pointer in a composite widget instance is of type **XtOrderProc**:

Cardinal(*XtOrderProc)(*Widget***)**
Widget *W*;

W Specifies the widget.

Composite widgets that allow clients to order their children (usually homogeneous boxes) can call their widget instance's insert_position procedure from the class's insert_child procedure to determine where a new child should go in its children array. Thus, a client of a composite class can apply different sorting criteria to widget instances of the class, passing in a different insert_position procedure when it creates each composite widget instance.

The return value of the insert_position procedure indicates how many children should go before the widget. Returning **zero** indicates that the widget should go before all other children; returning num_children indicates that it should go after all other children. The default insert_position subroutine returns num_children and can be overidden by a specific composite widget's resource list or by the argument list provided when the composite widget is created.

Core Resource Set

XmNaccelerators
XmNbackground
XmNborderColor
XmNborderWidth
XmNdepth
XmNheight
XmNscreen
XmNtranslations
XmNx
XmNancestorSensitive
XmNbackgroundPixmap
XmNborderPixmap
XmNcolormap
XmNdestroyCallback
XmNmappedWhenManaged
XmNsensitive
XmNwidth
XmNy

XmNaccelerators

Class	**XmCAccelerators**
Type	**XtTranslations**
Default	**NULL**
Access	CSG

Description

The **XmNaccelerators** resource specifies a translation table that is bound with its actions in the context of a particular widget. The accelerator table can then be installed on some destination widget.

XmNancestorSensitive

Class	**XmCSensitive**
Type	**Boolean**
Default	**True**; **ShellAncestorSensitive** if inherited for the **ApplicationShell**, **OverrideShell**, **Shell**, **TopLevelShell**, **TransientShell**, **VendorShell**, **WMShell**, **DialogShell**, or **MenuShell** widget
Access	G; CSG if inherited for the **MenuShell** widget

Description

The **XmNancestorSensitive** resource specifies whether the immediate parent of the widget receives input events. Use the **XtSetSensitive** subroutine to change the parameter to preserve data integrity (see the **XmNsensitive** resource later in this article).

XmNbackground

Class **XmCBackground**

Type **Pixel**

Default **White**; **dynamic** if inherited for the **ArrowButton**, **BulletinBoard**, **CascadeButton**, **Command**, **DrawingArea**, **DrawnButton**, **FileSelectionBox**, **Form**, **Frame**, **Label**, **List**, **MainWindow**, **Manager**, **MessageBox**, **PanedWindow**, **Primitive**, **PushButton**, **RowColumn**, **Scale**, **ScrollBar**, **ScrolledWindow**, **SelectionBox**, **Separator**, **Text**, or **ToggleButton** widget

Access CSG

Description

The **XmNbackground** resource specifies the background color for the widget.

XmNbackgroundPixmap

Class **XmCPixmap**

Type **Pixmap**

Default **XmUNSPECIFIED_PIXMAP**

Access CSG

Description

The **XmNBackgroundPixmap** resource specifies a pixmap for tiling the background. The first tile is placed at the upper left–hand corner of the widget window.

XmNborderColor

Class **XmCBorderColor**

Type **Pixel**

Default **Black**

Access CSG

Description

The **XmNborderColor** resource specifies the color of the border in a pixel value.

XmNborderPixmap

Class **XmCPixmap**

Type **Pixmap**

Default **XmUNSPECIFIED_PIXMAP**

Access CSG

Description

The **XmNborderPixmap** resource specifies the pixmap to be used for tiling the border. The first tile is placed at the upper left–hand corner of the border.

XmNborderWidth

Class **XmCBorderWidth**

Type **Dimension**

Default **1**; **0** if inherited for the **ArrowButton**, **BulletinBoard**, **CascadeButton**, **Command**, **DrawingArea**, **DrawnButton**, **FileSelectionBox**, **Form**,**Frame**, **Label**, **List**, **MainWindow**, **Manager**, **MessageBox**,**PanedWindow**, **Primitive**, **PushButton**, **RowColumn**, **Scale**, **ScrollBar**,**ScrolledWindow**, **SelectionBox**, **Separator**, **Text**, or **ToggleButton**widget; **dynamic** if inherited for the **RowColumn** widget

Access CSG

Description

The **XmNborderWidth** resource specifies the width of the border that surrounds the widget window on all four sides. The width is specified in pixels. A width value of zero specifies that no border will show.

XmNcolormap

Class **XmCColormap**

Type **Colormap**

Default **XtCopyFromParent**; **ShellColormap** if inherited for the **ApplicationShell**, **OverrideShell**, **Shell**, **TopLevelShell**, **TransientShell**, **VendorShell**, **WMShell**, **DialogShell**, or **MenuShell** widget

Access CG

Description

The **XmNcolormap** resource specifies the colormap to be used for conversions to the **Pixel** type for this widget instance. When changed, previously generated pixel values are not be affected, but newly generated values are affected in the new colormap.

XmNdepth

Class **XmCDepth**

Type **int**

Default **XtCopyFromParent**;

ShellDepth if inherited for the **ApplicationShell**, **OverrideShell**, **Shell**,**TopLevelShell**, **TransientShell**, **VendorShell**, **WMShell**, **DialogShell**, or **MenuShell** widget

Access CG

Description

The **XmNdepth** resource specifies the number of bits that can be used for each pixel in the widget window. Applications should not change or set the value of this resource as it is set by the **XtIntrinsics** subroutine when the widget is created.

XmNdestroyCallback

Class **XmCCallback**

Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdestroyCallback** resource points to a callback list containing routines to be called when the widget is destroyed.

XmNheight

Class	**XmCHeight**
Type	**Dimension**
Default	**0**; **16** if inherited for the **RowColumn** widget
Access	CSG

Description

The **XmNheight** resource specifies the height of the widget window in pixels, not including the border area.

XmNmappedWhenManaged

Class	**XmCMappedWhenManaged**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNmappedWhenManaged** resource maps the widget (made visible) as soon as it is both realized and managed, if set to the **True** value. If set to the **False** value, the client is responsible for mapping and unmapping the widget. If the value is changed from the **True** value to the **False** value after the widget is realized and managed, the widget is unmapped.

XmNscreen

Class	**XmCScreen**
Type	**Pointer**
Default	**XtCopyScreen**
Access	CG

Description

The **XmNscreen** resource specifies the screen on which a widget instance resides. It is read only, except for shells.

XmNsensitive

Class	**XmCSensitive**
Type	**Boolean**
Default	**True**

Access CSG

Description

The **XmNsensitive** resource determines whether a widget receives input events. If a widget is sensitive, the **XtIntrinsics** Event Manager dispatches to the widget all keyboard, mouse button, motion, window enter/leave, and focus events. Insensitive widgets do not receive these events. Use the **XtSetSensitive** subroutine to change the sensitivity parameter. Using the **XtSetSensitive** subroutine ensures that if a parent widget has the **XmNsensitive** resource set to the **False** value, the ancestor–sensitive flag of all its children is appropriately set.

XmNtranslations

Class **XmCTranslations**

Type **XtTranslations**

Default **NULL**

Access CSG

Description

The **XmNtranslations** resource points to a translations list, which is a list of events and actions that are to be performed when the events occur.

XmNwidth

Class **XmCWidth**

Type **Dimension**

Default **0**; **16** if inherited for the **RowColumn** widget

Access CSG

Description

The **XmNwidth** resource specifies the width of the widget window in pixels, not including the border area.

XmNx

Class **XmCPosition**

Type **Position**

Default **0**

Access CSG

Description

The **XmNx** resource specifies the x–coordinate of the upper left–hand corner of the widget (excluding the border) in relation to its parent widget.

XmNy

Class **XmCPosition**

Type **Position**

Default **0**

Access CSG

Description

The **XmNy** resource specifies the y–coordinate of the upper left–hand corner of the widget (excluding the border) in relation to its parent widget.

Object Resource Set

XmNdestroyCallback

XmNdestroyCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdestroyCallback** resource calls a callback list when the gadget is destroyed.

RectObj Resource Set

XmNancestorSensitive
XmNborderWidth
XmNheight
XmNsensitive
XmNwidth
XmNx
XmNy

XmNancestorSensitive

Class	**XmCSensitive**
Type	**Boolean**
Default	**XtCopyFromParent**; **0** if inherited from the **ArrowButtonGadget**, **Gadget**, **LabelGadget**, or **ToggleButtonGadget**
Access	CSG

Description

The **XmNancestorSensitive** resource specifies whether the immediate parent of the widget receives input events. Use the **XtSetSensitive** subroutine to change the argument to preserve data integrity (see the **XmNsensitive** resource that follows).

XmNborderWidth

Class	**XmCBorderWidth**
Type	**Dimension**
Default	**1**; **0** if inherited from the **ArrowButtonGadget**, **Gadget**, **LabelGadget**, or **ToggleButtonGadget** gadget
Access	CSG

Description

The **XmNborderWidth** resource forces a value of zero since an **RectObj** widget does not have border width; this resource is not used.

XmNheight

Class	**XmCHeight**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNheight** resource specifies in pixels the height of the widget window, not including the border area.

XmNsensitive

Class	**XmCSensitive**
Type	**Boolean**

Default	**True**
Access	CSG

Description

The **XmNsensitive** resource determines whether a widget receives input events. If a widget is sensitive, the **XtIntrinsics** Event Manager dispatches to the widget all keyboard, mouse button, motion, window enter/leave, and focus events. Insensitive widgets do not receive these events. Use the **XtSetSensitive** subroutine to change the sensitivity parameter. Using the **XtSetSensitive** subroutine ensures that if a parent widget has the **XmNsensitive** resource set to the **False** value, the ancestor–sensitive flag of all its children is appropriately set.

XmNwidth

Class	**XmCWidth**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNwidth** resource contains the width of the **XmRectObj** rectangular display image.

XmNx

Class	**XmCPosition**
Type	**Position**
Default	**0**
Access	CSG

Description

The **XmNx** resource contains the x–coordinate of the gadget upper left–hand corner in relation to its parent window.

XmNy

Class	**XmCPosition**
Type	**Position**
Default	**0**
Access	CSG

Description

The **XmNy** resource contains the y–coordinate of the gadget upper left–hand corner in relation to its parent window.

Shell Resource Set

XmNallowShellResize
XmNgeometry
XmNpopdownCallback
XmNsaveUnder
XmNcreatePopupChildProc
XmNoverrideRedirect
XmNpopupCallback

XmNallowShellResize

Class	**XmCAllowShellResize**
Type	**Boolean**
Default	**False**; **True** if inherited for the **MenuShell** widget
Access	CSG; G if inherited for the **MenuShell** widget

Description

The **XmNallowShellResize** resource specifies that if this resource is the **False** value, the **XmShell** widget instance returns the **XtGeometryNo** value to all geometry requests from its children.

XmNcreatePopupChildProc

Class	**XmCCreatePopupChildProc**
Type	**XmCreatePopupChildProc**
Default	**NULL**
Access	CSG

Description

The **XmNcreatePopupChildProc** resource specifies the pointer to a subroutine that is called when the **XmShell** widget instance is popped up by the **XtPopup** subroutine.

XmNgeometry

Class	**XmCGeometry**
Type	**String**; **caddr_t** if inherited for the **MenuShell** widget
Default	**NULL**
Access	CSG

Description

The **XmNgeometry** resource specifies the desired geometry for the widget instance. This resource is only examined when the widget instance is unrealized and the number of its managed children is changed. It is used to change the values of the **XmNx**, **XmNy**, **XmNwidth**, and **XmNheight** resources.

XmNoverrideRedirect

Class **XmCOverrideRedirect**

Type **Boolean**

Default **False**; **True** if inherited for the **OverrideShell** or **MenuShell** widget

Access CSG

Description

The **XmNoverrideRedirect** resource specifies the **True** value if the widget instance is an extremely temporary window that should be ignored by the window manager. Applications and users should not normally alter this resource.

XmNpopdownCallback

Class **XmCCallback**

Type **XtCallbackList**; **caddr_t** if inherited for the **MenuShell** widget

Default **NULL**

Access C

Description

The **XmNpopdownCallback** resource specifies a list of callbacks that is called when the widget instance is popped down by the **XtPopdown** subroutine.

XmNpopupCallback

Class **XmCCallback**

Type **XtCallbackList**;

caddr_t if inherited for the **MenuShell** widget

Default **NULL**

Access C

Description

The **XmNpopupCallback** resource specifies a list of callbacks that is called when the widget instance is popped down by the **XtPopup** subroutine.

XmNsaveUnder

Class **XmCSaveUnder**

Type **Boolean**

Default **False**; **True** if inherited for the **OverrideShell** or **MenuShell** widget

Access CSG

Description

The **XmNsaveUnder** resource specifies a **True** value if it is desirable to save the contents of the screen beneath this widget instance, thus avoiding expose events when the instance is unmapped. This is a hint, and an implementation can save contents whenever it needs to, including always or never.

TopLevelShell Resource Set

XmNiconic **XmNiconName**

XmNiconic

Class	**XmCIconic**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNiconic** resource specifies, if it is the **True** value when the widget instance is realized, that the widget instance indicates to the window manager that the application wishes to start iconic, irrespective of the **XtinitialState** resource. This resource is only examined by the Intrinsics during a call to the **XtRealizeWidget** subroutine, and is ignored at all other times.

XmNiconName

Class	**XmCIconName**
Type	**String**
Default	**NULL**
Access	CSG

Description

The **XmNiconName** resource specifies the short form of the application name to be displayed by the window manager when the application is iconified.

VendorShell Resource Set

XmNdeleteResponse
XmNkeyboardFocusPolicy
XmNmwmDecorations
XmNmwmFunctions
XmNmwmInputMode
XmNmwmMenu
XmNshellUnitType

XmNdeleteResponse

Class	**XmCDeleteResponse**
Type	**unsigned char**
Default	**XmDESTROY**; **XmUNMAP** if inherited for the **DialogShell** widget
Access	CSG

Description

The **XmNdeleteResponse** resource determines what action the shell takes in response to a **WM_DELETE_WINDOW** message. The setting can be one of three values: **XmDESTROY**, **XmUNMAP**, and **XmDO_NOTHING**. The resource is scanned, and the appropriate action is taken, after the **WM_DELETE_WINDOW** callback list (if any) that is registered with the Protocol manager has been called.

XmNkeyboardFocusPolicy

Class	**XmCKeyboardFocusPolicy**
Type	**unsigned char**
Default	**XmEXPLICIT**
Access	CSG

Description

The **XmNkeyboardFocusPolicy** resource determines allocation fo keyboard focus within the widget hierarchy whose root begins at this shell. The X keyboard focus must be directed to somewhere in the hierarchy for this client–side focus management to take effect.

XmNmwmDecorations

Class	**XmCMwmDecorations**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNmwmDecorations** resource includes the decoration flags (specific decorations to add or remove from the window manager frame) for **MWM_HINTS**.

XmNmwmFunctions

Class	**XmCMwmFunctions**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNmwmFunctions** resource includes the function flags (specific window manager subroutines to include or exclude from the system menu) for **MWM_HINTS**.

XmNmwmInputMode

Class	**XmCMwmInputMode**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNmwmInputMode** includes the input mode flag (application modal or system modal input focus constraints) for **MWM_HINTS**.

XmNmwmMenu

Class	**XmCMwmMenu**
Type	**String**
Default	**NULL**
Access	CSG

Description

The **XmNmwmMenu** resource specifies the menu items that the AIXwindows window manager should add to the end of the system menu. The contents of the string are a list of items separated by \n with the following format:

label [mnemonic] [accelerator] function

If more than one item is specified, the items should be separated by a newline character.

XmNshellUnitType

Class	**XmCShellUnitType**
Type	**unsigned char**
Default	**XmPIXELS**
Access	CSG

Description

The **XmNshellUnitType** resource determines geometric resource interpretation. The following values are allowed:

- **XmPixels** – all values provided to the widget are treated as normal pixel values.
- **Xm100TH_MILLIMETERS** – all values provided to the widget are treated as 1/100 millimeter.
- **Xm100TH_INCHES** – all values provided to the widget are treated as 1/100 inch.
- **Xm100TH_POINTS** – all values provided to the widget are treated as 1/100 point. A point is a unit used in text processing applications and is defined as 1/72 inch.
- **Xm100TH_FONT_UNITS** – all values provided to the widget are treated as 1/100 font unit. The value used for the font unit is determined in one of two ways:

  – the **XmNfont** resource can be used in a defaults file or on the command line, or

  – the standard command line options of **–fn** and **–font** can be used.

  The font unit value is taken as the **QUAD_WIDTH** property of the font. The **XmSetFontUnits** subroutine allows applications to specify the font unit values.

WMShell Resource Set

XmNheightInc	**XmNiconMask**
XmNiconPixmap	**XmNiconWindow**
XmNiconX	**XmNiconY**
XmNinitialState	**XmNinput**
XmNmaxAspectX	**XmNmaxAspectY**
XmNmaxHeight	**XmNmaxWidth**
XmNminAspectX	**XmNminAspectY**
XmNminHeight	**XmNminWidth**
XmNtitle	**XmNtransient**
XmNwaitForWm	**XmNwidthInc**
XmNwindowGroup	**XmNmwmTimeout**

XmNheightInc

Class	**XmCHeightInc**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNheightInc** resource specifies allowable height for the widget instance by the window manager if this resource is defined. The sizes are the **XmNminHeight** resource plus an integral multiple of the **XmNheightInc** resource, subject to the **XmNmaxHeight** resource.

XmNiconMask

Class	**XmCIconMask**
Type	**Pixmap**
Default	**NULL**
Access	CSG

Description

The **XmNiconMask** resource specifies a bitmap that could be used by the window manager to clip the **XmNiconPixmap** value bitmap to make the icon non–rectangular.

XmNiconPixmap

Class	**XmCIconPixmap**
Type	**Pixmap**
Default	**NULL**
Access	CSG

Description

The **XmNiconPixmap** resource specifies a bitmap that could be used by the window manager as the application icon.

XmNiconWindow

Class	**XmCIconWindow**
Type	**Window**
Default	**NULL**
Access	CSG

Description

The **XmNiconWindow** resource specifies the ID of a window that could be used by the window manager as the application icon.

XmNiconX

Class	**XmCIconX**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNiconX** resource specifies a suitable place to put the application icon; it is a hint to the window manager in root window coordinates. Since the window manager controls icon placement policy, this resource can be ignored.

XmNiconY

Class	**XmCIconY**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNiconY** resource specifies a suitable place to put the application icon; this is a hint to the window manager in root window coordinates. Since the window manager controls icon placement policy, this may be ignored.

XmNinitialState

Class	**XmCInitialState**
Type	**int**
Default	**1**
Access	CSG

Description

The **XmNinitialState** resource specifies the state in which the application wishes the widget instance to start. It must be the **NormalState** or **IconicState** constants.

XmNinput

Class	**XmCInput**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNinput** resource specifies the application input model for this widget and its descendants.

XmNmaxAspectX

Class	**XmCMaxAspectX**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNmaxAspectX** resource gives the maximum aspect ratio (X/Y) that the application wishes the widget instance to have.

XmNmaxAspectY

Class	**XmCMaxAspectY**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNmaxAspectY** resource specifies the maximum aspect ratio (X/Y) that the application allows the widget instance to have.

XmNmaxHeight

Class	**XmCMaxHeight**
Type	**int**
Default	**–1**
Access	CSG

Description

The **XmNmaxHeight** resource specifies the maximum height that the application allows the widget instance to have.

XmNmaxWidth

Class **XmCMaxWidth**

Type **int**

Default **–1**

Access CSG

Description

The **XmNmaxWidth** resource specifies the maximum width that the application allows the widget instance to have.

XmNminAspectX

Class **XmCMinAspectX**

Type **int**

Default **–1**

Access CSG

Description

The **XmNminAspectX** resource specifies the minimum aspect ratio (X/Y) that the application allows the widget instance to have.

XmNminAspectY

Class **XmCMinAspectY**

Type **int**

Default **–1**

Access CSG

Description

The **XmNminAspectY** resource specifies the minimum aspect ratio (X/Y) that the application allows the widget instance to have.

XmNminHeight

Class **XmCMinHeight**

Type **int**

Default **–1**

Access CSG

Description

The **XmNminHeight** resource specifies the minimum height that the application allows the widget instance to have.

XmNminWidth

Class **XmCMinWidth**

Type **int**

Default	**–1**
Access	CSG

Description

The **XmNminWidth** resource specifies the minimum width that the application allows the widget instance to have.

XmNtitle

Class	**XmCTitle**
Type	**char ***
Default	**NULL**
Access	CSG

Description

The **XmNtitle** resource specifies the application name to be displayed by the window manager.

XmNtransient

Class	**XmCTransient**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNtransient** resource specifies a Boolean value that is the **True** value if the widget instance is a transient window that should be treated more lightly by the window manager. Applications and users should not normally alter this resource. The window for which the widget instance is a transient is set to the **XmNwindowGroup** value.

XmNwaitForWm

Class	**XmCWaitForWm**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNwaitForWm** resource specifies that the **XtIntrinsics** subroutine waits the length of time given by the **XmNwmTimeout** value for the window manager to respond to certain actions when a **True** value, before assuming that there is no window manager present. This resource is altered by **XtIntrinsics** as it receives, or fails to receive, responses from the window manager.

XmNwidthInc

Class	**XmCWidthInc**
Type	**int**

Default **–1**

Access CSG

Description

The **XmNwidthInc** resource specifies allowable width for the widget instance by the window manager if this resource is defined. The sizes are the **XmNminWidth** resource plus an integral multiple of the **XmNwidthInc** resource, subject to the **XmNmaxWidth** resource.

XmNwindowGroup

Class **XmCWindowGroup**

Type **XID**

Default **None**

Access CSG

Description

The **XmNwindowGroup** resource specifies the ID of a window for which this widget instance is associated; a window manager may treat all windows in a group in some way, for example, by always moving or iconifying them together.

If this is set on an **Shell** widget instance which has no parent but has popup children, this resource is set to the same value on all popup children of the widget instance, all popup children of these children, and so on.

See also the **XmNtransient** resource.

XmNmwmTimeout

Class **XmCWmTimeout**

Type **int**

Default **fivesecond**

Access CSG

Description

The **XmNmwmTimeout** resource specifies the length of time that the **XtIntrinsics** subroutine waits for the window manager to respond to certain actions before assuming that there is no window manager present.

XmArrowButton Resource Set

XmNactivateCallback **XmNarrowDirection**

XmNarmCallback **XmNdisarmCallback**

XmNactivateCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNactivateCallback** resource specifies a callback subroutine that is called when the **ArrowButton** widget is activated. To activate the button, press and release the left mouse button while the pointer is inside the **ArrowButton** widget. Activating the **ArrowButton** widget also disarms it. The callback reason is the **XmCR_ACTIVATE** value.

XmNarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNarmCallback** resource specifies a callback subroutine that is called when the **ArrowButton** widget is armed. To arm this widget, press the left mouse button while the pointer is inside the **ArrowButton** widget. The callback reason is the **XmCR_ARM** value.

XmNarrowDirection

Class	**XmCArrowDirection**
Type	**unsigned char**
Default	**XmARROW_UP**
Access	CSG

Description

The **XmNarrowDirection** resource sets the arrow direction. The values for this resource are the **XmARROW_UP**, **XmARROW_DOWN**, **XmARROW_LEFT**, and **XmARROW_RIGHT** values.

XmNdisarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**

Access C

Description

The **XmNdisarmCallback** resource specifies a callback subroutine that is called when the **ArrowButton** widget is disarmed. To disarm this widget, press and release the left mouse button while the pointer is inside the **ArrowButton** widget. The callback reason is the **XmCR_DISARM** value.

XmArrowButtonGadget Resource Set

XmNactivateCallback	**XmNarrowDirection**
XmNarmCallback	**XmNdisarmCallback**

XmNactivateCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNactivateCallback** resource specifies a list of callbacks that is called when the **ArrowButtonGadget** gadget is activated. To activate the button, press and release the left mouse button while the pointer is inside the **ArrowButtonGadget** gadget. Activating the **ArrowButtonGadget** gadget also disarms it. The callback reason is the **XmCR_ACTIVATE** value.

XmNarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNarmCallback** resource specifies a list of callbacks that is called when the **ArrowButtonGadget** gadget is armed. To arm this gadget, press the left mouse button while the pointer is inside the **ArrowButtonGadget** gadget. The callback reason is the **XmCR_ARM** value.

XmNarrowDirection

Class	**XmCArrowDirection**
Type	**int**
Default	**XmARROW_UP**
Access	CSG

Description

The **XmNarrowDirection** resource sets the arrow direction. The values for this resource are as follows:

- **XmARROW_UP**
- **XmARROW_DOWN**
- **XmARROW_LEFT**
- **XmARROW_RIGHT**

XmNdisarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdisarmCallback** resource specifies a list of callbacks that is called when the **ArrowButtonGadget** gadget is disarmed. To disarm this gadget, press and release the left mouse button while the pointer is inside the **ArrowButtonGadget** gadget. The callback reason is the **XmCR_DISARM** value.

XmBulletinBoard Resource Set

XmNallowOverlap	**XmNautoUnmanage**
XmNbuttonFontList	**XmNcancelButton**
XmNdefaultButton	**XmNdefaultPosition**
XmNdialogStyle	**XmNdialogTitle**
XmNfocusCallback	**XmNlabelFontList**
XmNmapCallback	**XmNmarginHeight**
XmNmarginWidth	**XmNnoResize**
XmNresizePolicy	**XmNshadowType**
XmNstringDirection	**XmNtextFontList**
XmNtextTranslations	**XmNunmapCallback**

XmNallowOverlap

Class	**XmCAllowOverlap**
Type	**Boolean**
Default	**True**
Access	CSG; N/A if inherited for the **Command**, **Form**, or **MessageBox** widget

Description

The **XmNallowOverlap** resource controls the policy for overlapping children widgets. If a **True** value, the **BulletinBoard** widget allows geometry requests that result in overlapping children.

XmNautoUnmanage

Class	**XmCAutoUnmanage**
Type	**Boolean**
Default	**True**; **False** if inherited for the **Command** or **FileSelectionBox** widget
Access	CSG; N/A if inherited for the **Form** widget

Description

The **XmNautoUnmanage** resource controls whether the **BulletinBoard** widget is automatically unmanaged after a button is activated. If this resource is the **True** value, the **BulletinBoard** widget adds a callback to button children (PushButtons, PushButtonGadgets, and DrawnButtons) that unmanages the **BulletinBoard** widget when a button is activated. In addition, if the parent of the **BulletinBoard** widget is a **DialogShell** widget, the unmap callback subroutines are called. If this resource is the **False** value, the **BulletinBoard** widget is not automatically unmanaged.

XmNbuttonFontList

Class	**XmCButtonFontList**
Type	**XmFontList**

Default **NULL**

Access CSG; N/A if inherited for the **Command** or **Form** widget

Description

The **XmNbuttonFont** resource specifies the font list for the **BulletinBoard** widget's button children (PushButtons, PushButtonGadgets, ToggleButtons, and ToggleButtonGadgets). If this resource is the **NULL** value, the **XmNtextFontList** resource is used for buttons.

XmNcancelButton

Class **XmCWidget**

Type **Widget**

Default **NULL**; **Cancel button** if inherited for the **MessageBox** or **SelectionBox** widget

Access SG; N/A if inherited for the **Command** or **Form** widget; G if inherited for the **MessageBox** widget

Description

The **XmNcancelButton** resource specifies the widget ID of the **Cancel** button. This resource is set by the subclasses of the **BulletinBoard** widget, which define a default button. The **BulletinBoard** widget does not directly provide any behavior for that button.

XmNdefaultButton

Class **XmCWidget**

Type **Widget**

Default **NULL**; **OK button** if inherited for the **MessageBox** or **SelectionBox** widget

Access SG; N/A if inherited for the **Command** or **Form** widget; G if inherited for the **MessageBox** widget

Description

The **XmNdefaultButton** resource specifies the widget ID of the default button. This resource is set by the subclasses of the **BulletinBoard** widget, which define a a default button. The **BulletinBoard** widget defines translations and installs accelerators that activate that button when the return key is pressed.

XmNdefaultPosition

Class **XmCDefaultPosition**

Type **Boolean**

Default **True**; **False** if inherited for the **Command** widget

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNdefaultPosition** resource controls whether the **BulletinBoard** widget is automatically positioned by its parent. If this resource is the **True** value, and the parent of the **BulletinBoard** widget is a **DialogShell** widget, then the **BulletinBoard** widget is centered within or around the parent of the **DialogShell** widget when the **BulletinBoard**

widget is mapped and managed. If this resource is the **False** value, the **BulletinBoard** widget is not automatically positioned.

XmNdialogStyle

Class **XmCDialogStyle**

Type **unsigned char**

Default **dynamic**

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNdialogStyle** resource indicates the dialog style associated with the **BulletinBoard** widget. If the parent of the **BulletinBoard** widget is a **DialogShell** widget, the parent widget is configured according to this resource. The possible values for this resource are as follows:

- **XmDIALOG_SYSTEM_MODAL** – used for dialogs that must be responded to before any other interaction in any application.
- **XmDIALOG_APPLICATION_MODAL** – used for dialogs that must be responded to before any other interactions in the same application.
- **XmDIALOG_MODELESS** – used for dialogs that do not interrupt interaction of any application.
- **XmDIALOG_WORK_AREA** – used for non–dialog (the parent is not a subclass of the DialogShell) **BulletinBoard** widgets.

XmNdialogTitle

Class **XmCXmString**

Type **XmString**

Default **NULL**

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNdialogTitle** resource specifies the dialog title. If this resource is not **NULL**, and the parent of the **BulletinBoard** widget is a subclass of **WMShell**, the **BulletinBoard** widget sets the **XmNTitle** resource of its parent to the value of this resource.

XmNfocusCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNfocusCallback** resource specifies the list of callbacks that is called when the **BulletinBoard** widget (or one of its descendants) accepts the input focus. The callback reason is the **XmCR_FOCUS** value.

XmNlabelFontList

Class **XmCLabelFontList**

Type **XmFontList**

Default **NULL**

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNlabelFontList** resource specifies the font list used for the **Label** children of the **BulletinBoard** widget (Labels and LabelGadgets). If this resource is the **NULL** value, the **XmNtextFontList** resource is used for labels also.

XmNmapCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNmapCallback** resource specifies the list of callback subroutines that is called only when the parent of the **BulletinBoard** widget is a **DialogShell** widget; in which case, this callback list is invoked when the **BulletinBoard** widget is mapped. The callback reason is the **XmCR_MAP** value.

XmNmarginHeight

Class **XmCMarginHeight**

Type **short**

Default **10**

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNmarginHeight** resource specifies the minimum spacing in pixels between the top or bottom edge of the **BulletinBoard** widget and any child widget.

XmNmarginWidth

Class XmCMarginWidth

Type short

Default 10

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNmarginWidth** resource specifies the minimum spacing in pixels between the left or right edge of the **BulletinBoard** widget and any child widget.

XmNnoResize

Class XmCNoResize

Type Boolean

Default False

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNnoResize** resource controls whether resize controls are included in the window manager frame around the dialog. If this resource is the **True** value, the AIXwindows window manager does not include resize controls in the window manager frame containing the **DialogShell** or **TopLevelShell** parent of the **BulletinBoard** widget. If this resource is the **False** value, the window manager frame does include resize controls. The preferred way to manipulate the set of controls provided by the AIXwindows window manager is to specify values for the mwm resources provided by the **VendorShell** widget.

XmNresizePolicy

Class XmCResizePolicy

Type unsigned char

Default XmRESIZE_ANY; **XmRESIZE_NONE** if inherited for the **Command** widget

Return Values XmRESIZE_NONE (fixed size)
XmRESIZE_ANY (shrink or grow as needed)
XmRESIZE_GROW (grow only)

Access CSG

Description

The **XmNresizePolicy** resource controls the policy for resizing the **BulletinBoard** widgets. Possible values include the following:

- **XmRESIZE_NONE** – fixed size.
- **XmRESIZE_ANY** – shrink or grow as needed.
- **XmRESIZE_GROW** – grow only.

XmNshadowType

Class **XmCShadowType**

Type **unsigned char**

Default **XmSHADOW_OUT**

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNshadowType** resource describes the shadow drawing style for the **BulletinBoard** widget. This resource can have the following values:

- **XmSHADOW_IN** – draws the **BulletinBoard** shadow such that it appears inset. This means that the bottom shadow visuals and top shadow visuals are reversed.
- **XmSHADOW_OUT** – draws the **BulletinBoard** shadow such that it appears outset.
- **XmSHADOW_ETCHED_IN** – draws the **BulletinBoard** shadow using a double line, giving the effect of a line etched into the window, similar to the **Separator** widget.
- **XmSHADOW_ETCHED_OUT** – draws the **BulletinBoard** shadow using a double line, giving the effect of a line coming out of the window, similar to the **Separator** widget.

BulletinBoard widgets draw shadows just within their borders if the **XmNshadowThickness** resource is greater than zero. If the parent of a **BulletinBoard** widget is a **DialogShell** widget, the **BulletinBoard** widget dynamically changes the default for the **XmNshadowThickness** resource from 0 to 1.

XmNstringDirection

Class **XmCStringDirection**

Type **XmStringDirection**

Default **XmSTRING_DIRECTION_L_TO_R**

Return Values **XmSTRING_DIRECTION_L_TO_R (the default)**
XmSTRING_DIRECTION_R_TO_L
XmSTRING_DIRECTION_REVERT

Access CSG; N/A if inherited for the **Form** widget

Description

The **XmNstringDirection** resource specifies the initial rendering direction for text within a dialog. The subclasses of the **BulletinBoard** widget which create **XmString** components set the **XmNstringDirection** resource of these components based on the value of this resource. The **BulletinBoard** widget does not directly provide any behavior for this resource.

XmNtextFontList

Class **XmCTextFont**

Type **XmFontList**

Default **NULL**

Access CSG; N/A if inherited for the **Form** or **MessageBox** widget

Description

The **XmNtextFont** resource specifies the font list used for the **Text** children of the **BulletinBoard** widget. If there is no **XmNbuttonFontList** resource specified, this resource is used for buttons. If there is no **XmNlabelFontList** resource specified, this resource is used for labels also.

XmNtextTranslations

Class	**XmCTranslations**
Type	**XtTranslations**
Default	**NULL**
Access	C; N/A if inherited for the **Form** or **MessageBox** widget

Description

The **XmNtextTranslations** resource adds translations to any **Text** widget (or a **Text** widget subclass) that is added as a child of the **BulletinBoard** widget.

XmNunmapCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNunmapCallback** resource specifies the list of callback subroutines that is called only when the parent of the **BulletinBoard** widget is a **DialogShell** widget; this callback list is invoked when the **BulletinBoard** widget is unmapped. The callback reason is the **XmCR_UNMAP** value.

XmCascadeButton Resource Set

XmNactivateCallback
XmNcascadingCallback
XmNsubMenuId
XmNcascadePixmap
XmNmappingDelay

XmNactivateCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNactivateCallback** resource specifies the list of callbacks that is called when the user activates the **CascadeButton** widget, and there is no submenu attached to pop up. The activation occurs by releasing a mouse button or by typing the mnemonic associated with the widget. The specific mouse button depends on information in the **RowColumn** widget parent. The callback reason is the **XmCR_ACTIVATE** value.

XmNcascadePixmap

Class **XmCPixmap**

Type **Pixmap**

Default **"menu_cascade"**

Access CSG

Description

The **XmNcascadePixmap** resource specifies the cascade pixmap displayed on the right end of the widget when an **CascadeButton** widget is used within an **Popup** widget or **Pulldown MenuPane** widget and a submenu is attached. The **XmLabel** widget class resources: **XmNmarginRight**, **XmNmarginTop**, and **XmNmarginBottom**, can be modified to ensure room is left for the cascade pixmap. The default cascade pixmap is an arrow pointing to the right.

XmNcascadingCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNcascadingCallback** resource specifies the list of callback routines that is called just prior to the mapping of the submenu associated with the **CascadeButton** widget. The callback reason is the **XmCR_CASCADING** value.

XmNmappingDelay

Class **XmCMappingDelay**

Type **int**

Default **100**

Access CSG

Description

The **XmNmappingDelay** resource specifies the amount of time, in milliseconds, between when a **CascadeButton** widget becomes armed and when it maps its submenu. This delay is only used when the widget is within an **Popup** menu or a **Pulldown MenuPane** menu.

XmNsubMenuId

Class **XmCMenuWidget**

Type **Widget**

Default **0**

Access CSG

Description

The **XmNsubMenuId** resource specifies the widget ID for the **Pulldown MenuPane** menu to be associated with this **CascadeButton** widget. The specified **MenuPane** widget is displayed when the **CascadeButton** widget is armed. The **MenuPane** widget must be created with the appropriate parentage depending on the type of menu used. See the **XmCreatePulldownMenu** subroutine, the **XmCreatePopupMenu** subroutine, and the **XmCreateOptionMenu** subroutine for more information on the menu systems.

XmCascadeButtonGadget Resource Set

XmNactivateCallback
XmNcascadingCallback
XmNsubMenuId
XmNcascadePixmap
XmNmappingDelay

XmNactivateCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNactivateCallback** resource specifies the list of callbacks that is called when the user activates the **CascadeButtonGadget** gadget, and there is no submenu attached to pop up. The activation occurs by releasing a mouse button or by typing the mnemonic associated with the gadget. The specific mouse button depends on information in the **RowColumn** widget parent. The callback reason is the **XmCR_ACTIVATE** value.

XmNcascadePixmap

Class	**XmCPixmap**
Type	**Pixmap**
Default	**"menu_cascade"**
Access	CSG

Description

The **XmNcascadePixmap** resource specifies the cascade pixmap displayed on the right end of the gadget when an **CascadeButtonGadget** gadget is used within an **Popup** widget or **Pulldown MenuPane** widget and a submenu is attached. The **XmLabelGadget** widget class resources: **XmNmarginRight**, **XmNmarginTop**, and **XmNmarginBottom**, can be modified to ensure room is left for the cascade pixmap. The default cascade pixmap is an arrow pointing to the right.

XmNcascadingCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNcascadingCallback** resource specifies the list of callbacks that is called just prior to the mapping of the submenu associated with the **CascadeButtonGadget** gadget. The callback reason is the **XmCR_CASCADING** value.

XmNmappingDelay

Class	**XmCMappingDelay**
Type	**int**
Default	**100**
Access	CSG

Description

The **XmNmappingDelay** resource specifies the amount of time, in milliseconds, between when an **CascadeButtonGadget** gadget becomes armed and when it maps its submenu. This delay is only used when the gadget is within an **XmPopup** menu or an **XmPulldown MenuPane** menu.

XmNsubMenuId

Class	**XmCMenuWidget**
Type	**Widget**
Default	**0**
Access	CSG

Description

The **XmNsubMenuId** resource specifies the widget ID for the **XmPulldown MenuPane** menu to be associated with this **CascadeButtonGadget** gadget. The specified **MenuPane** widget is displayed when the **CascadeButtonGadget** gadget is armed. The **MenuPane** widget must be created with the appropriate parentage depending on the type of menu used. See the **XmCreatePulldownMenu** subroutine, the **XmCreatePopupMenu** subroutine, and the **XmCreateOptionMenu** subroutine for more information on the menu systems.

XmCommand Resource Set

XmNcommand
XmNcommandChangedCallback
XmNcommandEnteredCallback
XmNhistoryItems
XmNhistoryItemCount
XmNhistoryMaxItems
XmNhistoryVisibleItemCount
XmNpromptString

XmNcommand

Class	**XmCTextString**
Type	**XmString**
Default	**NULL**
Access	CSG

Description

The **XmNcommand** resource contains the current command line text. This is the **XmNtextString** resource in the **SelectionBox** widget, renamed for the **Command** widget. This resource can also be modified using the **XmCommandSetValue** and the **XmCommandAppendValue** subroutines. The command area is a **Text** widget.

XmNcommandChangedCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNcommandChangedCallback** resource specifies the list of callbacks that is called when the value of the command changes. The callback reason is the **XmCR_COMMAND_CHANGED** value. This is equivalent to the **XmNvalueChangedCallback** resource of the **Text** widget, except that an **XmCommandCallback** structure is returned, loaded with the **String** widget.

XmNcommandEnteredCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNcommandEnteredCallback** resource specifies the list of callbacks that is called when a command is entered in the **Command** widget. The callback reason is the **XmCR_COMMAND_ENTERED** value. An **XmCommandCallback** structure is returned.

XmNhistoryItems

Class	**XmCItems**
Type	**XmStringTable**
Default	**NULL**
Access	CSG

Description

The **XmNhistoryItems** resource lists the **String** widget items that make up the contents of the history list. This is the **XmNlistItems** resource in the **SelectionBox** widget, renamed for the **Command** widget.

XmNhistoryItemCount

Class	**XmCItemCount**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNhistoryItemCount** resource specifies the number of the **XmStrings** value items in the **XmNhistoryItems** resource. This is the **XmNlistItemCount** resource in the **SelectionBox** widget, renamed for the **Command** widget.

XmNhistoryMaxItems

Class	**XmCMaxItems**
Type	**int**
Default	**100**
Access	CSG

Description

The **XmNhistoryMaxItems** resource specifies the maximum number of items allowed in the history list. Once this number is reached, the first list item is removed from the list for each new item added to the list, that is, for each new command entered.

XmNhistoryVisibleItemCount

Class	**XmCVisibleItemCount**
Type	**int**
Default	**8**
Access	CSG

Description

The **XmNhistoryVisibleItemCount** resource specifies the number of items in the history list that are visible at one time. In effect, it sets the height (in lines) of the history list window. This is the **XmNvisibleItemCount** resource in the **XmSelectionBox** resource set, renamed for the **Command** widget.

XmNpromptString

Class	**XmCString**
Type	**XmString**
Default	**">"**
Access	CSG

Description

The **XmNpromptString** resource prompts for the command line. This is the **XmNselectionLabelString** resource in the **SelectionBox** widget, renamed for the **Command** widget.

XmDrawingArea Resource Set

XmNexposeCallback	**XmNinputCallback**
XmNmarginHeight	**XmNmarginWidth**
XmNresizeCallback	**XmNresizePolicy**

XmNexposeCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNexposeCallback** resource invokes the callback list when the DrawingArea receives an exposure event The callback reason is the **XmCR_EXPOSE** value The callback structure also includes the exposure event and the exposure region.

XmNinputCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNinputCallback** resource invokes the callback list when the **DrawingArea** widget receives a keyboard or mouse event (key or button, up or down). The callback reason is **XmCR_INPUT**. The callback structure also includes the input event.

XmNmarginHeight

Class	**XmCMarginHeight**
Type	**short**
Default	**10**
Access	CSG

Description

The **XmNmarginHeight** resource specifies the minimum spacing in pixels between the top or bottom edge of the **DrawingArea** widget and any child widget.

XmNmarginWidth

Class	**XmCMarginWidth**
Type	**short**
Default	**10**
Access	CSG

Description

The **XmNmarginWidth** resource specifies the minimum spacing in pixels between the left or right edge of the **DrawingArea** widget and any child widget.

XmNresizeCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNresizeCallback** resource invokes the callback list when the **DrawingArea** widget is realized The callback reason is the **XmCR_RESIZE** value.

XmNresizePolicy

Class	**XmCResizePolicy**
Type	**unsigned char**
Default	**XmRESIZE_ANY**
Access	CSG

Description

The **XmNresizePolicy** resource controls the policy for resizing the **DrawingArea** widget Possible values for this resource include the following:

XmRESIZE_NONE – fixed size.

XmRESIZE_ANY – shrink or grow as needed,

XmRESIZE_GROW – grow only.

XmDrawnButton Resource Set

XmNactivateCallback
XmNarmCallback
XmNdisarmCallback
XmNexposeCallback
XmNpushButtonEnabled
XmNresizeCallback
XmNshadowType

XmNactivateCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNactivateCallback** resource specifies the list of callbacks that is called when the widget is selected. The callback reason is the **XmCR_ACTIVATE** value.

XmNarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNarmCallback** resource specifies the list of callbacks that is called when the widget is armed. The callback reason is the **XmCR_ARM** value.

XmNdisarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdisarmCallback** resource specifies the list of callbacks that is called when the widget is disarmed. The callback reason is the **XmCR_DISARM** value.

XmNexposeCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNexposeCallback** resource specifies the list of callbacks that is called when the widget receives an exposure event. The callback reason is the **XmCR_EXPOSE** value.

XmNpushButtonEnabled

Class	**XmCPushButtonEnabled**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNpushButtonEnabled** resource enables or disables the three–dimensional shadow drawing, as in the **PushButton** widget.

XmNresizeCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNresizeCallback** resource specifies the list of callbacks that is called when the widget receives a resize event. The callback reason is the **XmCR_RESIZE** value. The event returned for this callback is the **NULL** value.

XmNshadowType

Class	**XmCShadowType**
Type	**unsigned_char**
Default	**XmSHADOW_ETCHED_IN**
Return Values	**XmSHADOW_IN** **XmSHADOW_OUT** **XmSHADOW_ETCHED_IN** **XmSHADOW_ETCHED_OUT**
Access	CSG

Description

The **XmNshadowType** resource describes the drawing style for the **XmDrawnButton** widget. This resource can have the following values:

- **XmSHADOW_IN** – draws the **XmDrawnButton** widget such that the shadow appears inset. This means that the bottom shadow visuals and top shadow visuals are reversed.
- **XmSHADOW_OUT** – draws the **XmDrawnButton** widget such that the shadow appears outset.
- **XmSHADOW_ETCHED_IN** – draws the **XmDrawnButton** widget using a double line. This gives the effect of a line etched into the window. The thickness of the double line is equal to the value of the **XmNshadowThickness** resource.

- **XmSHADOW_ETCHED_OUT** – draws the **XmDrawnButton** widget using a double line. This gives the effect of a line coming out of the window. The thickness of the double line is equal to the value of the **XmNshadowThickness** resource.

XmFileSelectionBox Resource Set

XmNdirMask **XmNdirSpec**
XmNfileSearchProc **XmNfilterLabelString**
XmNlistUpdated

XmNdirMask

Class	**XmCDirMask**
Type	**XmString**
Default	"*"
Access	CSG

Description

The **XmNdirMask** resource specifies the directory mask used in determining the files to be displayed in the box.

XmNdirSpec

Class	**XmCDirSpec**
Type	**XmString**
Default	**NULL**
Access	CSG

Description

The **XmNdirSpec** resource specifies the full file specification. This resource overrides the **XmNtextString** resource in the **XmSelectionBox** widget.

XmNfileSearchProc

Class	**XmCFileSearchProc**
Type	**XtProc**
Default	See Below
Access	CSG

Description

The **XmNfileSearchProc** resource specifies a directory search procedure to replace the default file selection search procedure. The **XmFileSelectionBox** widget default file search procedure fills the needs of most applications. Since it is impossible to cover the requirements of all applications; the default search procedure can be replaced.

The file search procedure is called with two arguments: the **XmFileSelectionBox** widget and the **XmFileSelectionCallbackStruct** widget structure. The callback structure contains all required information to conduct a directory search, including the current file search mask. Once called, it is up to the search routine to generate a new list of files and to update the file selection widget by using the **XtSetValues** subroutine.

The following resources must be set: the **XmNitems**, **XmNitemsCount**, **XmNlistUpdated**, and **XmNdirSpec** resources. Set the **XmNitems** resource to the new list of files. If there are no files, set this resource to the **NULL** value. This sets the **XmNitems** resource, which is associated with the **XmSelectionBox** widget.

If there are no files, set the **XmNitemsCount** resource to zero. This sets the **XmNitemsCount** resource, which is associated with the **XmSelectionBox** widget. Always set the **XmNlistUpdated** resource to the **True** value when updating the file list using a search procedure, even if there are no files. Setting the **XmNdirSpec** resource is optional, but recommended. Set this resource to the full file specification of the directory searched. The directory specification is displayed above the list box.

XmNfilterLabelString

Class	**XmCXmString**
Type	**XmString**
Default	**"File Filter"**
Access	CSG

Description

The **XmNfilterLabelString** resource specifies the string value for the label located above the **DIR_MASK** text entry field.

XmNlistUpdated

Class	**XmCListUpdated**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNlistUpdated** resource specifies an resource that is set only by the file search procedure. This resource is set to the **True** value, if the file list has been updated.

XmForm Constraint Resource Set

XmNbottomAttachment
XmNbottomOffset
XmNbottomPosition
XmNbottomWidget
XmNleftAttachment
XmNleftOffset
XmNleftPosition
XmNleftWidget
XmNresizable
XmNrightAttachment
XmNrightOffset
XmNrightPosition
XmNrightWidget
XmNtopAttachment
XmNtopOffset
XmNtopPosition
XmNtopWidget

XmNbottomAttachment

Class **XmCAttachment**

Type **unsigned char**

Default **XmATTACH_NONE**

Access CSG

Description

The **XmNbottomAttachment** resource specifies the attachment of the bottom side of the child widget. This resource can have the following values:

- **XmATTACH_NONE** – do not attach this side.
- **XmATTACH_FORM** – attach the bottom side of the child widget to the bottom side of the **Form** widget.
- **XmATTACH_OPPOSITE_FORM** – attach the bottom side of the child widget to the top side fo the **Form** widget.
- **XmATTACH_WIDGET** – attach the bottom side of the child widget to the top side of the widget or gadget specified in the **XmNbottomWidget** resource.
- **XmATTACH_OPPOSITE_WIDGET** – attach the bottom side of the child widget to the bottom side of the widget or gadget specified in the **XmNbottomWidget** resource.
- **XmATTACH_POSITION** – attach the bottom side of the child widget to a relative position in the **Form** widget. This position is specified by the **XmNbottomPosition** resource.
- **XmATTACH_SELF** – attach the bottom of the child widget to its initial position in the **Form** widget.

XmNbottomOffset

Class **XmCOffset**

Type **int**

Default **0**

Access CSG

Description

The **XmNbottomOffset** resource specifies the constant offset between the bottom side of the child widget and the object to which it is attached. This resource is ignored if the **XmNbottomAttachment** resource is set to the **XmATTACH_POSITION** value. The relationship established remains, regardless of any resizing operations that occur.

XmNbottomPosition

Class **XmCAttachment**

Type **int**

Default **0**

Access CSG

Description

The **XmNbottomPosition** resource determines the relative position of the bottom side of the child widget. The relative position is a fractional value of the height of the **Form** widget. The fractional value is equal to the value of this resource divided by the value of the **XmNfractionBase** resource of the **Form** widget. This resource is only used if the **XmNbottomAttachment** resource is set to the **XmATTACH_POSITION** value.

XmNbottomWidget

Class **XmCWidget**

Type **Widget**

Default **NULL**

Access CSG

Description

The **XmNbottomWidget** resource specifies the widget or gadget to which the bottom side of the child widget is attached. This resource is used if the **XmNbottomAttachment** resource is set to either the **XmATTACH_WIDGET** value or the **XmATTACH_OPPOSITE_WIDGET** value.

XmNleftAttachment

Class **XmCAttachment**

Type **unsigned char**

Default **XmATTACH_NONE**

Access CSG

Description

The **XmNleftAttachment** resource specifies the attachment of the left side of the child widget. This resource can have the following values:

- **XmATTACH_NONE** – do not attach this side.
- **XmATTACH_FORM** – attach the left side of the child widget to the left side of the **Form** widget.

- **XmATTACH_OPPOSITE_FORM** – attach the left side of the child widget to the right side of the **Form** widget.
- **XmATTACH_WIDGET** – attach the left side of the child widget to the right side of the widget or gadget specified in the **XmNleftWidget** resource.
- **XmATTACH_OPPOSITE_WIDGET** – attach the left side of the child widget to the left side of the widget or gadget specified in the **XmNleftWidget** resource.
- **XmATTACH_POSITION** – attach the left side of the child widget to a relative position in the **Form** widget. This position is specified by the **XmNleftPosition** resource.
- **XmATTACH_SELF** – attach the left side of the child widget to its initial position in the **Form** widget.

XmNleftOffset

Class **XmCOffset**

Type **int**

Default **0**

Access CSG

Description

The **XmNleftOffset** resource specifies the constant offset between the left side of the child and the object to which it is attached. This resource is ignored if the **XmNleftAttachment** resource is set to the **XmATTACH_POSITION** value. The relationship established remains, regardless of any resizing operations that occur.

XmNleftPosition

Class **XmCAttachment**

Type **int**

Default **0**

Access CSG

Description

The **XmNleftPosition** resource determines the relative position of the left side of the child widget. The relative position is a fractional value of the width of the **Form** widget. The fractional value is equal to the value of this resource divided by the value of the **XmNfractionBase** resource of the **Form** widget. This resource is only used if the **XmNleftAttachment** resource is set to the **XmATTACH_POSITION** value.

XmNleftWidget

Class **XmCWidget**

Type **Widget**

Default **NULL**

Access CSG

Description

The **XmNleftWidget** resource specifies the widget or gadget to which the left side of the child widget is attached. This resource is used if the **XmNleftAttachment** resource is set to either the **XmATTACH_WIDGET** value or the **XmATTACH_OPPOSITE_WIDGET** value.

XmNresizable

Class **XmCBoolean**

Type **Boolean**

Default **True**

Access CSG

Description

The **XmNresizable** resource specifies whether a child widget can be resized by the **Form** widget. The default value is the **True** value.

XmNrightAttachment

Class **XmCAttachment**

Type **unsigned char**

Default **XmATTACH_NONE**

Access CSG

Description

The **XmNrightAttachment** resource specifies the attachment of the right side of the child widget. This resource can have the following values:

- **XmATTACH_NONE** – do not attach this side.
- **XmATTACH_FORM** – attach the right side of the child widget to the right side of the **Form** widget.
- **XmATTACH_OPPOSITE_FORM** – attach the right side of the child widget to the left side of the **Form** widget.
- **XmATTACH_WIDGET** – attach the right side of the child widget to the left side of the widget or gadget specified in the **XmNrightWidget** resource.
- **XmATTACH_OPPOSITE_WIDGET** – attach the right side of the child widget to the right side of the widget or gadget specified in the **XmNrightWidget** resource.
- **XmATTACH_POSITION** – attach the right side of the child widget to a relative position in the **Form** widget. This position is specified by the **XmNrightPosition** resource.
- **XmATTACH_SELF** – attach the right side of the child widget to its initial position in the **Form** widget.

XmNrightOffset

Class **XmCOffset**

Type **int**

Default **0**

Access CSG

Description

The **XmNrightOffset** resource specifies the constant offset between the right side of the child and the object to which it is attached. This resource is ignored if the **XmNrightAttachment** resource is set to the **XmATTACH_POSITION** value. The relationship established remains, regardless of any resizing operations that occur.

XmNrightPosition

Class **XmCAttachment**

Type **int**

Default **0**

Access CSG

Description

The **XmNrightPosition** resource determines the relative position of the right side of the child widget. The relative position is a fractional value of the width of the **Form** widget. The fractional value is equal to the value of this resource divided by the value of the **XmNfractionBase** resource of the **Form** widget. This resource is only used if the **XmNrightAttachment** resource is set to the **XmATTACH_POSITION** value.

XmNrightWidget

Class **XmCWidget**

Type **Widget**

Default **NULL**

Access CSG

Description

The **XmNrightWidget** resource specifies the widget or gadget to which the right side of the child widget is attached. This resource is used if the **XmNrightAttachment** resource is set to either the **XmATTACH_WIDGET** value or the **XmATTACH_OPPOSITE_WIDGET** value.

XmNtopAttachment

Class **XmCAttachment**

Type **unsigned char**

Default **XmATTACH_NONE**

Access CSG

Description

The **XmNtopAttachment** resource specifies the attachment of the top side of the child widget. This resource can have the following values:

- **XmATTACH_NONE** – do not attach this side.
- **XmATTACH_FORM** – attach the top side of the child widget to the top side of the **Form** widget.
- **XmATTACH_OPPOSITE_FORM** – attach the top side of the child widget to the bottom side fo the **Form** widget.
- **XmATTACH_WIDGET** – attach the top side of the child widget to the bottom side of the widget or gadget specified in the **XmNtopWidget** resource.
- **XmATTACH_OPPOSITE_WIDGET** – attach the top side of the child widget to the top side of the widget or gadget specified in the **XmNtopWidget** resource.
- **XmATTACH_POSITION** – attach the top side of the child widget to a relative position in the **Form** widget. This position is specified by the **XmNtopPosition** resource.
- **XmATTACH_SELF** – attach the top of the child widget to its initial position in the **Form** widget.

XmNtopOffset

Class **XmCOffset**

Type **int**

Default **0**

Access CSG

Description

The **XmNtopOffset** resource specifies the constant offset between the top side of the child widget and the object to which it is attached. This resource is ignored if the **XmNtopAttachment** resource is set to the **XmATTACH_POSITION** value. The relationship established remains, regardless of any resizing operations that occur.

XmNtopPosition

Class **XmCAttachment**

Type **int**

Default **0**

Access CSG

Description

The **XmNtopPosition** resource determines the relative position of the top side of the child widget. The relative position is a fractional value of the height of the **Form** widget. The fractional value is equal to the value of this resource divided by the value of the **XmNfractionBase** resource of the **Form** widget. This resource is only used if the **XmNtopAttachment** resource is set to the **XmATTACH_POSITION** value.

XmNtopWidget

Class	**XmCWidget**
Type	**Widget**
Default	**NULL**
Access	CSG

Description

The **XmNtopWidget** resource specifies the widget or gadget to which the top side of the child widget is attached. This resource is used if the **XmNtopAttachment** resource is set to either the **XmATTACH_WIDGET** value or the **XmATTACH_OPPOSITE_WIDGET** value.

XmForm Resource Set

XmNfractionBase
XmNhorizontalSpacing
XmNrubberPositioning
XmNverticalSpacing

XmNfractionBase

Class	**XmCMaxValue**
Type	**int**
Default	**100**
Access	CSG

Description

The **XmNfractionBase** resource specifies the denominator used in calculating the relative position of a child widget using the **XmATTACH_POSITION** value constraints.

XmNhorizontalSpacing

Class	**XmCSpacing**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNhorizontalSpacing** resource specifies the offset for right and left attachments.

XmNrubberPositioning

Class	**XmCRubberPositioning**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNrubberPositioning** resource indicates the default attachment for a child of the **Form** widget. If this Boolean resource is set to a **False** value, the left and top of the child defaults to being attached to the left and top side of the **Form** widget. If this resource is set to a **True** value, then the child defaults to being attached to its initial position in the **Form** widget.

XmNverticalSpacing

Class	**XmCSpacing**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNverticalSpacing** resource specifies the offset for top and bottom attachments.

XmFrame Resource Set

XmNmarginWidth **XmNshadowType**

XmNmarginHeight

XmNmarginWidth

Class	**XmCMarginWidth**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNmarginWidth** resource specifies the padding space on the left and right sides between the child of the **Frame** widget and its shadow drawing.

XmNmarginHeight

Class	**XmCMarginHeight**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNmarginHeight** resource specifies the padding space on the top and bottom sides between the child of the **Frame** widget and its shadow drawing.

XmNshadowType

Class	**XmCShadowType**
Type	**unsigned char**
Default	**XmSHADOW_ETCHED_IN**
Access	CSG

Description

The **XmNshadowType** resource describes the drawing style for the **Frame** widget. This resource can have the following values:

- **XmSHADOW_IN** Draw the **Frame** widget such that is appears inset. This means that the bottom shadow visuals and top shadow visuals are reversed.
- **XmSHADOW_OUT** Draw the **Frame** widget such that it appears outset.
- **XmSHADOW_ETCHED_IN** Draw the **Frame** widget using a double line which gives the effect of a line etched into the window similar to the **Separator** widget.
- **XmSHADOW_ETCHED_OUT** Draw the **Frame** widget using a double line which gives the effect of a line coming out of the window similar to the **Separator** widget.

XmGadget Resource Set

XmNhelpCallback
XmNhighlightThickness
XmNtraversalOn
XmNuserData
XmNhighlightOnEnter
XmNshadowThickness
XmNunitType

XmNhelpCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNhelpCallback** resource specifies the list of callbacks that is called when the help key sequence is pressed. The callback reason is the **XmCR_HELP** value.

XmNhighlightOnEnter

Class	**XmCHighlightOnEnter**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNhighlightOnEnter** resource specifies whether draw border highlighting. This resource is ignored if the **XmNtraversalOn** resource is the **True** value.

XmNhighlightThickness

Class	**XmCHighlightThickness**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNhighlightThickness** resource specifies the thickness of the highlighting rectangle.

XmNshadowThickness

Class	**XmCShadowThickness**
Type	**short**
Default	**0**; **2** if inherited for the **ArrowButtonGadget**, **CascadeButtonGadget**, **PushButtonGadget**, or **SeparatorGadget** gadget
Access	CSG

Description

The **XmNshadowThickness** resource specifies the size of the drawn border shadow.

XmNtraversalOn

Class	**XmCTraversalOn**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNtraversalOn** resource specifies if traversal is activated for this gadget.

XmNunitType

Class	**XmCUnitType**
Type	**unsigned char**
Default	**XmPIXELS**
Return Values	**XmPIXELS** **Xm100TH_MILLIMETERS** **Xm1000TH_INCHES** **Xm100TH_POINTS** **Xm100TH_FONT_UNITS**
Access	CSG

Description

The **XmNunitType** resource provides the basic support for resolution independence. It defines the type of units a widget uses with sizing and positioning resources. Unless the **XmNunitType** resource is explicitly set, it defaults to the unit type of the parent widget. If the parent has a unit type of the **Xm100TH_POINTS** value, any of its children whose **XmNunitType** resource is not set also has a unit type of the **Xm100TH_POINTS** value. This feature applies only to widgets whose parents are a subclass of the **Manager** widget. Widgets whose parents are not subclasses of **Manager** have a unit type of the **XmPIXELS** value.

The **XmNunitType** resource can have the following values:

- **XmPIXELS** – All values provided to the widget are treated as normal pixel values. This is the default for the resource.
- **Xm100TH_MILLIMETERS** – All values provided to the widget are treated as 1/100 millimeter.
- **Xm1000TH_INCHES** – All values provided to the widget are treated as 1/1000 inch.
- **Xm100TH_POINTS** – All values provided to the widget are treated as 1/100 point. A point is a unit typically used in text processing applications and is defined as 1/72 inch.
- **Xm100TH_FONT_UNITS** – All values provided to the widget are treated as 1/100th font unit. The value to be used for the font unit is determined in one of two ways. The **XmNfont** resource can be used in a defaults file or on the command line. The standard command line options of the **–fn** and **–font** flags can also be used. The font unit value is taken as the **QUAD_WIDTH** value of the font. The **XmSetFontUnits** subroutine allows applications to specify the font unit values.

XmNuserData

Class	**XmCUserData**
Type	**caddr_t**
Default	**NULL**
Access	CSG

Description

The **XmNuserData** resource allows the application to attach any necessary specific data to the gadget. This is an internally unused resource.

XmLabel Resource Set

XmNaccelerator
XmNalignment
XmNlabelInsensitivePixmap
XmNlabelString
XmNmarginBottom
XmNmarginLeft
XmNmarginTop
XmNmnemonic
XmNstringDirection
XmNacceleratorText
XmNfontList
XmNlabelPixmap
XmNlabelType
XmNmarginHeight
XmNmarginRight
XmNmarginWidth
XmNrecomputeSize

XmNaccelerator

Class	**XmCAccelerator**
Type	**String**
Default	**NULL**
Access	CSG

Description

The **XmNaccelerator** resource sets the accelerator on a button gadget in a menu. This resource is only used in menu gadgets. It contains the left–hand side of a translation, which is a character string. This string must be a single key event.

Note: Accelerators for buttons are supported only for certain buttons, namely for the **PushButton** widget and the **ToggleButton** widget in **pulldown**, **popup** and **workarea** menus and only in certain menu gadgets.

XmNacceleratorText

Class	**XmCAcceleratorText**
Type	**XmString**
Default	**NULL**
Access	CSG

Description

The **XmNacceleratorText** specifies the text displayed for the accelerator.

XmNalignment

Class	**XmCAlignment**
Type	**unsigned char**
Default	**XmALIGNMENT_CENTER**
Access	CSG

Description

The **XmNalignment** resource specifies the label alignment for text or pixmap style as follows:

- **XmALIGNMENT_CENTER** (center alignment) causes the centers of the lines to be vertically aligned in the center of the parent window. For a pixmap, its center is vertically aligned with the center of the widget window.
- **XmALIGNMENT_END** (right alignment) causes the right sides of the lines to be vertically aligned with the right edge of the parent window. For a pixmap, its right side is vertically aligned with the right edge of the widget window.
- **XmALIGNMENT_BEGINNING** (left alignment) causes the left sides of the lines to be vertically aligned with the left edge of the parent window. For a pixmap, its left side is vertically aligned with the left edge of the widget window.

  The above descriptions for text are correct when **XmNstringDirection** is **XmSTRING_DIRECTION_L_TO_R**. When that resource is **XmSTRING_DIRECTION_R_TO_L**, the descriptions for **XmALIGNMENT_BEGINNING** and **XmALIGNMENT_END** are switched.

XmNfontList

Class	**XmCFontLIst**
Type	**XmFontLIst**
Default	**"Fixed"**
Access	CSG

Description

The **XmNfontList** resource specifies the font of the text used in the widget. Refer to **XmFontListCreate** for more information on the creation and the structure of a font list.

XmNlabelInsensitivePixmap

Class	**XmCLabelInsensitivePixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNlabelInsensitivePixmap** resource specifies the pixmap used as the button face if the **XmNlabelType** resource is the **XmPIXMAP** value and the button is insensitive. Refer to **XmStringCreate** or **XmStringCreateLtoR** for more information on the creation and the structure of compound strings.

XmNlabelPixmap

Class	**XmCPixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNlabelPixmap** resource specifies the pixmap when the **LabelType** widget is the **XmPIXMAP** value.

XmNlabelString

Class **XmCXmString**

Type **XmString**

Default **NULL;**
'\0' if inherited for the **DrawnButton** widget

Access CSG

Description

The **XmNlabelString** resource specifies the label when the **XmNlabelType** resource is the **XmSTRING** value.

XmNlabelType

Class **XmCLabelType**

Type **unsigned char**

Default **XmSTRING**

Access CSG

Description

The **XmNlabelType** resource specifies the label type as follows:

- **XmSTRING** (text)
- **XmPIXMAP** (icon data in pixmap)

XmNmarginBottom

Class **XmCMarginBottom**

Type **short**

Default **0;**
dynamic if inherited for the **CascadeButton** or **PushButton** widget

Access CSG

Description

The **XmNmarginBottom** resource specifies the amount of spacing that is to be left, after the bottom margin (the **XmNmarginHeight** resource) of the widget, before the label is drawn.

XmNmarginHeight

Class **XmCMarginHeight**

Type **short**

Default 2;
dynamic if inherited for the **DrawnButton** widget

Access CSG

Description

The **XmNmarginHeight** resource specifies the amount of blank space between the bottom edge of the top shadow and the label, and between the top edge of the bottom shadow and the label.

XmNmarginLeft

Class **XmCMarginLeft**

Type **short**

Default **0**;
dynamic if inherited for the **ToggleButton** or **PushButton** widget

Access CSG

Description

The **XmNmarginLeft** resource specifies the amount of spacing that is to be left, after the left margin (the **XmNmarginWidth** resource) of the widget, before the label is drawn.

XmNmarginRight

Class **XmCMarginRight**

Type **short**

Default **0**;
dynamic if inherited for the **CascadeButton** or **PushButton** widget

Access CSG

Description

The **XmNmarginRight** resource specifies the amount of spacing that is to be left, after the right margin (the **XmNmarginWidth** resource) of the widget, before the label is drawn.

XmNmarginTop

Class **XmCMarginTop**

Type **short**

Default **0**;
dynamic if inherited for the **CascadeButton** or **PushButton** widget

Access CSG

Description

The **XmNmarginTop** resource specifies the amount of spacing that is to be left, after the top margin (the **XmNmarginHeight** resource) of the widget, before the label is drawn.

XmNmarginWidth

Class	**XmCMarginWidth**
Type	**short**
Default	**2**; **dynamic** if inherited for the **CascadeButton** or **PushButton** widget
Access	CSG

Description

The **XmNmarginWidth** resource specifies the amount of blank space between the right edge of the top shadow and the label, and between the left edge of the bottom shadow and the label.

XmNmnemonic

Class	**XmCMnemonic**
Type	**char**
Default	**'\0'**
Access	CSG

Description

The **XmNmnemonic** resource provides the user with alternate means for selection a button. The buttons must be visible for mnemonics to work. Buttons, which are in either a **menubar**, a **popup menupane**, a **pulldown menupane**, are allowed to have a mnemonic.

This resource contains a single character. The first character in the label string that exactly matches the mnemonic is underlined when the button is displayed.

When a mnemonic is specified for a **menubar** button, the user activates the mnemonic by pressing the **meta** key and the specified mnemonic key simultaneously. All other mnemonics are activated by pressing the specified mnemonic. Mnemonics are case–sensitive; the character underlined can be a modified key, but the key pressed should always be unmodified.

XmNrecomputeSize

Class	**XmCRecomputeSize**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNrecomputeSize** resource specifies a Boolean value that indicates whether the widget always attempts to be big enough to contain the label. If the **True** value, an **XtSetValues** subroutine with a new label string or pixmap, causes the widget to attempt to

shrink or to expand to exactly fit (accounting for margins) the new label string or pixmap. If the **False** value, the widget never attempts to change size on its own.

XmNstringDirection

Class	**XmCStringDirection**
Type	**unsigned char**; **XmStringDirection** if inherited for the **Label**, **PushButton**, or **ToggleButton** widget
Default	**XmSTRING_DIRECTION_L_TO_R**
Access	CSG

Description

The **XmNstringDirection** resource specifies the direction in which the string is to be drawn. The values are the **XmSTRING_DIRECTION_L_TO_R** and **XmSTRING_DIRECTION_R_TO_L** values.

XmLabelGadget Resource Set

XmNaccelerator
XmNacceleratorText
XmNalignment
XmNfontList
XmNlabelInsensitivePixmap
XmNlabelPixmap
XmNlabelString
XmNlabelType
XmNmarginBottom
XmNmarginHeight
XmNmarginLeft
XmNmarginRight
XmNmarginTop
XmNmarginWidth
XmNmnemonic
XmNrecomputeSize
XmNstringDirection

XmNaccelerator

Exception: This resource does not apply to XmCascadeButtonGadget.

Class	**XmCAccelerator**
Type	**String**
Default	**NULL**
Access	CSG

Description

The **XmNaccelerator** resource sets the accelerator on a button widget in a menu and is used only in menu widgets. This resource contains the left–hand side of a translation, which is a character string. This string should be a single key event.

Note: Accelerators for buttons are supported only in certain menu widgets, and only for certain buttons, namely for the **PushButton** widget and the **ToggleButton** widget in **Pulldown**, **Popup** and **Work Area** menus.

XmNacceleratorText

Exception: This resource does not apply to XmCascadeButtonGadget.

Class	**XmCAcceleratorText**
Type	**XmString**
Default	**NULL**
Access	CSG

Description

The **XmNacceleratorText** resource specifies the text displayed for the accelerator.

XmNalignment

Class	**XmCAlignment**
Type	**unsigned char**
Default	**XmALIGNMENT_CENTER**

Return Values **XmALIGNMENT_CENTER** (center alignment)
XmALIGNMENT_END (right alignment)
XmALIGNMENT_BEGINNING (left alignment)

Access CSG

Description

The **XmNalignment** resource specifies the label alignment for text style, as in the following:

- **XmALIGNMENT_CENTER** (center alignment) Causes the centers of the lines to be vertically aligned in the center of the widget window.
- **XmALIGNMENT_END** (right alignment) Causes the right sides of the lines to be vertically aligned with the right edge of the widget window.
- **XmALIGNMENT_BEGINNING** (left alignment) Causes the left sides of the lines to be vertically aligned with the left edge of the widget window.

XmNfontList

Class **XmCFontList**

Type **XmFontList**

Default **"Fixed"**

Access CSG

Description

The **XmNfontList** resource specifies the font of the text used in the widget.

XmNlabelInsensitivePixmap

Class **XmCLabelInsensitivePixmap**

Type **Pixmap**

Default **XmUNSPECIFIED_PIXMAP**

Access CSG

Description

The **XmNlabelInsensitivePixmap** resource specifies the pixmap when the **XmNlabelType** resource is the **XmPIXMAP** value.

XmNlabelPixmap

Class **XmCPixmap**

Type **Pixmap**

Default **XmUNSPECIFIED_PIXMAP**

Access CSG

Description

The **XmNlabelPixmap** resource specifies the pixmap when the **XmNlabelType** resource is the **XmPIXMAP** value.

XmNlabelString

Class	**XmCXmString**
Type	**XmString**
Default	**NULL**
Access	CSG

Description

The **XmNlabelString** resource specifies the label when the **XmNlabelType** resource is the **XmString** value.

XmNlabelType

Class	**XmCLabelType**
Type	**unsigned char**
Default	**XmSTRING**
Return Values	**XmSTRING** (text) **XmPIXMAP** (icon data in pixmap)
Access	CSG

Description

The **XmNlabelType** resource specifies the label type. This resource can have the following values:

- **XmSTRING** – text displays **XmNlabelstring**.
- **XmPIXMAP** – icon data in pixmap displays **XmNlabelpixmap** or **XmNlabelInsensitivePixmap**.

XmNmarginBottom

Class	**XmCMarginBottom**
Type	**short**
Default	**0**; **dynamic** if inherited for the **CascadeButtonGadget** gadget
Access	CSG

Description

The **XmNmarginBottom** resource specifies the amount of spacing that is to be left, after the bottom margin (the **XmNmarginHeight** resource) of the widget, before the label is drawn.

XmNmarginHeight

Class	**XmCMarginHeight**
Type	**short**
Default	**2**
Access	CSG

Description

The **XmNmarginHeight** resource specifies the amount of blank space between the bottom edge of the top shadow and the label and between the top edge of the bottom shadow and the label.

XmNmarginLeft

Class **XmCMarginLeft**

Type **short**

Default **0**; **dynamic** if inherited for the **ToggleButtonGadget** gadget

Access CSG

Description

The **XmNmarginLeft** resource specifies the amount of spacing that is to be left, after the left margin (the **XmNmarginWidth** resource) of the widget, before the label is drawn.

XmNmarginRight

Class **XmCMarginRight**

Type **short**

Default **0**; **dynamic** if inherited for the **CascadeButtonGadget** gadget

Access CSG

Description

The **XmNmarginRight** resource specifies the amount of spacing that is to be left, after the right margin (the **XmNmarginWidth** resource) of the widget, before the label is drawn.

XmNmarginTop

Class **XmCMarginTop**

Type **short**

Default **0**; **dynamic** if inherited for the **CascadeButtonGadget** gadget

Access CSG

Description

The **XmNmarginTop** resource specifies the amount of spacing that is to be left, after the top margin (the **XmNmarginHeight** resource) of the widget, before the label is drawn.

XmNmarginWidth

Class **XmCMarginWidth**

Type **short**

Default **2**

Access CSG

Description

The **XmNmarginWidth** resource specifies the amount of blank space between the right edge of the top shadow and the label and between the left edge of the bottom shadow and the label.

XmNmnemonic

Class	**XmCMnemonic**
Type	**char**
Default	**'\0'**
Access	CSG

Description

The **XmNmnemonic** resource supports buttons that are in either a **Menubar**, a **Popup Menupane**, a **Pulldown Menupane**, or an **option menu**. When a mnemonic is specified for a **menubar button**, the user activates the mnemonic by pressing the Extend key and the specified mnemonic key simultaneously. All other mnemonics are activated by pressing the specified mnemonic. Mnemonics are not case–sensitive.

XmNrecomputeSize

Class	**XmCRecomputeSize**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNrecomputeSize** resource specifies a Boolean value that indicates whether the widget always attempts to be big enough to contain the label. If it is the **True** value, the **XtSetValues** subroutine changes value with a new label string or pixmap, causing the widget to attempt to shrink or expand to fit exactly (accounting for margins) the new label string or pixmap. If **False**, the widget never attempts to change size on its own.

XmNstringDirection

Class	**XmCStringDirection**
Type	**XmStringDirection**
Default	**XmSTRING_DIRECTION_L_TO_R**
Return Values	**XmSTRING_DIRECTION_L_TO_R** (the default) **XmSTRING_DIRECTION_R_TO_L**
Access	CSG

Description

The **XmNstringDirection** resource specifies the direction in which the string is to be drawn.

XmList Resource Set

XmNautomaticSelection
XmNlistSpacing
XmNmultipleSelectionCallback
XmNselectedItemCount
XmNselectedItems
XmNselectionPolicy
XmNsingleSelectionCallback
XmNstringDirection
XmNvisibleItemCount
XmNbrowseSelectionCallback
XmNdefaultActionCallback
XmNdoubleClickInterval
XmNextendedSelectionCallback
XmNfontList
XmNitemCount
XmNitems
XmNlistMarginHeight
XmNlistMarginWidth

XmNautomaticSelection

Class	**XmCAutomaticSelection**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNautomaticSelection** resource specifies the **XmNsingleSelectionCallback** resource when the user moves into a new item if the value is a **True** value and the selection mode is either a **XmBROWSE_SELECT** value or a **XmEXTENDED_SELECT** value. If a **False** value, no selection callbacks invoke until the user releases the mouse button.

XmNbrowseSelectionCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNbrowseSelectionCallback** resource specifies a list of callbacks that is called when an item is selected in the browse selection mode. The callback reason is the **XmCR_BROWSE_SELECT** value.

XmNdefaultActionCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdefaultActionCallback** resource specifies a list of callbacks that is called when an item is double clicked. The callback reason is the **XmCR_ DEFAULT_ACTION** value.

XmNdoubleClickInterval

Class	**XmCDoubleClickInterval**
Type	**int**
Default	**250**
Access	CSG

Description

The **XmNdoubleClickInterval** resource specifies the time (in milliseconds) that two consecutive clicks must occur to be considered a double–click action, rather than two single–click actions.

XmNextendedSelectionCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNextendedSelectionCallback** resource specifies a list of callbacks that is called when items are selected using the extended selection mode. The callback reason is the **XmCR_EXTENDED_SELECT** value.

XmNfontList

Class	**XmCFontList**
Type	**XmFontList**
Default	**"fixed"**
Access	CSG

Description

The **XmNfontList** resource specifies the font list associated with the list items. This is used in conjunction with the **XmNvisibleItemsCount** resource to determine the height of the **List** widget.

XmNitemCount

Class	**XmCItemCount**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNitemCount** resource specifies the total number of items. This number must match the **XmNitems** resource set. It is automatically updated by the list whenever an element is added to or deleted from the list.

XmNitems

Class	**XmCItems**
Type	**XmStringTable**
Default	**NULL**
Access	CSG

Description

The **XmNitems** resource points to an array of compound strings that are to be displayed as the list items.

XmNlistMarginHeight

Class	**XmCListMarginHeight**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNlistMarginHeight** resource specifies the height of margin between the **List** widget border and the items.

XmNlistMarginWidth

Class	**XmCListMarginWidth**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNlistMarginWidth** resource specifies the width of the margin between the **List** widget border and the items.

XmNlistSpacing

Class	**XmCListSpacing**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNlistSpacing** resource specifies spacing between list items. When keyboard traversal is enabled, this spacing increases by the value of the **XmNhighlightThickness** resource in **XmPrimitive**.

XmNmultipleSelectionCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNmultipleSelectionCallback** resource specifies a list of callbacks that is called when an item is selected in multiple selection mode. The callback reason is the **XmCR_MULTIPLE_SELECT** value.

XmNselectedItemCount

Class	**XmCSelectedItemCount**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNselectedItemCount** resource specifies the number of strings in the selected items list.

XmNselectedItems

Class	**XmCSelectedItems**
Type	**XmStringTable**
Default	**NULL**
Access	CSG

Description

The **XmNselectedItems** resource points to an array of compound strings that represent the list items that are currently selected, either by the user or by the application.

XmNselectionPolicy

Class	**XmCSelectionPolicy**
Type	**unsigned char**
Default	**XmBROWSE_SELECT**
Return Values	**XmSINGLE_SELECT** **XmMULTIPLE_SELECT** **XmXTENDED_SELECT** **XmBROWSE_SELECT**
Access	CSG

Description

The **XmNselectionPolicy** resource defines the interpretation of the selection action. This resource can be one of following values:

- **XmSINGLE_SELECT** – Only a single selection is allowed.
- **XmMULTIPLE_SELECT** – Multiple selections are allowed.
- **XmEXTENDED_SELECT** – Extended selections are allowed.
- **XmBROWSE_SELECT** – PM drag and browse subroutinely.

XmNsingleSelectionCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNsingleSelectionCallback** resource specifies a list of callbacks that is called when an item is selected in a single selection mode. The callback reason is the **XmCR_SINGLE_SELECT** value.

XmNstringDirection

Class	**XmCStringDirection**
Type	**XmStringDirection**
Default	**XmSTRING_DIRECTION_L_TO_R**
Access	CSG

Description

The **XmNstringDirection** resource specifies the direction to draw the string. The values are the **XmSTRING_DIRECTION_L_TO_R** and **XmSTRING_DIRECTION_R_TO_L** values.

XmNvisibleItemCount

Class	**XmCVisibleItemCount**
Type	**int**
Default	**1**
Access	CSG

Description

The **XmNvisibleItemCount** resource specifies the number of items that can fit in the visible space of the **List** widget work area. The list uses this value to determine its height.

XmMainWindow Resource Set

XmNcommandWindow
XmNmainWindowMarginHeight
XmNmainWindowMarginWidth
XmNmenuBar
XmNshowSeparator

XmNcommandWindow

Class **XmCCommandWindow**

Type **Widget**

Default **NULL**

Access CSG

Description

The **XmNcommandWindow** resource specifies the widget to be laid out as the **XmCommandWindow** widget. This widget must have been previously created and managed as a child of the **MainWindow** widget.

XmNmainWindowMarginHeight

Class **XmCMainWindowMarginHeight**

Type **Dimension**

Default **0**

Access CSG

Description

The **XmNmainWindowMarginHeight** resource specifies the margin height on the top and bottom of the **MainWindow** widget. This resource overrides any setting of the **XmNscrolledWindowMarginHeight** resource of the **ScrolledWindow** widget.

XmNmainWindowMarginWidth

Class **XmCMainWindowMarginWidth**

Type **Dimension**

Default **0**

Access CSG

Description

The **XmNmainWindowMarginWidth** resource specifies the margin width on the right and left sides of the **XmMainWindow** widget. This resource overrides any setting of the **XmNscrolledWindowMarginWidth** resource of the **ScrolledWindow** widget.

XmNmenuBar

Class **XmCMenuBar**

Type **Widget**

Default **NULL**

Access CSG

Description

The **XmNmenuBar** resource specifies the widget to be laid out as the **MenuBar** widget. This widget must have been previously created and managed as a child of the **MainWindow** widget.

XmNshowSeparator

Class	**XmCShowSeparator**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNshowSeparator** resource displays separators between the components of the **MainWindow** widget when set to a **True** value. If set to a **False** value, no separators are displayed.

XmManager Resource Set

XmNbottomShadowColor
XmNbottomShadowPixmap
XmNforeground
XmNhelpCallback
XmNhighlightColor
XmNhighlightPixmap
XmNshadowThickness
XmNtopShadowColor
XmNtopShadowPixmap
XmNunitType
XmNuserData

XmNbottomShadowColor

Class	**XmCForeground**
Type	**Pixel**
Default	**dynamic**
Access	CSG

Description

The **XmNbottomShadowColor** resource specifies the color to use to draw the bottom and right sides of the border shadow. This color is used if the **XmNbottomShadowPixmap** resource is a **NULL** value.

XmNbottomShadowPixmap

Class	**XmCBottomShadowPixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNbottomShadowPixmap** resource specifies the pixmap to use to draw the bottom and right sides of the border shadow.

XmNforeground

Class	**XmCForeground**
Type	**Pixel**
Default	**dynamic**
Access	CSG

Description

The **XmNforeground** resource specifies the foreground drawing color used by manager widgets.

XmNhelpCallback

Class	**XmCCallback**
Type	**XtCallback List**

Default	**NULL**
Access	C

Description

The **XmNhelpCallback** resource specifies the list of callbacks that are called when the help key sequence is pressed. The callback reason is the **XmCR_HELP** value. There is not a translation bound to this resource. It is up to the application to install a translation for help.

XmNhighlightColor

Class	**XmCForeground**
Type	**Pixel**
Default	**Black**
Access	CSG

Description

The **XmNhighlightColor** resource specifies the color of the highlighting rectangle. This color is used if the **XmNhighlightPixmap** resource is the **XmUNSPECIFIED_PIXMAP** value.

XmNhighlightPixmap

Class	**XmCHighlightPixmap**
Type	**Pixmap**
Default	**dynamic**
Access	CSG

Description

The **XmNhighlightPixmap** resource specifies the pixmap to use to draw the highlighting rectangle.

XmNshadowThickness

Class	**XmCShadowThickness**
Type	**short**
Default	**0**; **dynamic** if inherited for the **BulletinBoard**, **Command**, **FileSelectionBox**, **Frame**, **MessageBox**, or **SelectionBox** widget
Access	CSG

Description

The **XmNshadowThickness** resource specifies the thickness of the drawn border shadow.

XmNtopShadowColor

Class	**XmCBackground**
Type	**Pixel**
Default	**dynamic**
Access	CSG; N/A if inherited for the **PanedWindow** or **Scale** widget

Description

The **XmNtopShadowColor** resource specifies the color to use to draw the top and the left sides of the border shadow. This color is used if the **XmNtopShadowPixmap** resource is the **NULL** value.

XmNtopShadowPixmap

Class	**XmCTopShadowPixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNtopShadowPixmap** resource specifies the pixmap to use to draw the top and left sides of the border shadow.

XmNunitType

Class	**XmCUnitType**
Type	**unsigned char**
Default	**XmPIXELS**
Access	CSG

Description

The **XmNunitType** resource provides the basic support for resolution independence. It defines the type of units a widget uses with size and positioning resources. Unless the **XmNunitType** resource is explicitly set, it defaults to the unit type of the parent widget. If the parent type has a unit type of **Xm100th_POINTS**, any of its children whose **XmNunitType** resource is not set also has a unit type of the **Xm100th_POINTS** value. This feature applies only to widgets whose parents are a subclass of the **XmManager** resource. Widgets whose parents are not subclasses of the **XmManager** resource have a unit type of the **XmPIXELS** value.

The **XmNunitType** resource can have the following values:

- **XmPIXELS** All values provided to the widget are treated as normal pixel values. This is the default for the resource.
- **Xm100TH_MILLIMETERS** All values provided to the widget are treated as 1/100 of a millimeter.
- **Xm1000TH_INCHES** All values provided to the widget are treated as 1/1000 of an inch.
- **Xm100th_POINTS** All values provided to the widget are treated as 1/100 of a point. A point is a unit typically used in text processing applications and is defined as 1/72 of an inch.
- **Xm100TH_FONT_UNITS** All values provided to the widget are treated as 1/100 of a font unit. The value to be used for the font unit is gathered in one of two ways. The **XmNfont** resource can be used in a defaults file or on the command line. The standard command line options of the **–fn** and **–font** flags can also be used. The font unit value is the **XmQUAD_WIDTH** property value of the font. The **XmSetFontUnits** resource allows applications to specify the font unit values.

XmNuserData

Class	**XmCUserData**
Type	**caddr_t**
Default	**NULL**
Access	CSG

Description

The **XmNuserData** resource allows the application to attach any necessary specific data to the widget. This is an internally unused resource.

XmMessageBox Resource Set

XmNcancelCallback
XmNcancelLabelString
XmNdefaultButtonType
XmNokCallback
XmNokLabelString
XmNhelpLabelString
XmNmessageAlignment
XmNmessageString
XmNminimizeButtons
XmNdialogType
XmNsymbolPixmap

XmNcancelCallback

Class	**XtCallbackList**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNcancelCallback** resource specifies the list of callbacks that is called when the user clicks on the cancel button. The callback reason is the **XmCR_CANCEL** value.

XmNcancelLabelString

Class	**XmCXmString**
Type	**XmString**
Default	**"Cancel"**
Access	CSG

Description

The **XmNcancelLabelString** resource specifies the string label for the cancel button.

XmNdefaultButtonType

Class	**XmCDefaultButtonType**
Type	**unsigned char**
Default	**XmDIALOG_OK_BUTTON**
Access	CSG

Description

The **XmNdefaultButtonType** resource specifies the default of the **PushButton** widget. The valid types are as follows:

- **XmDIALOG_CANCEL_BUTTON.**
- **XmDIALOG_OK_BUTTON.**
- **XmDIALOG_HELP_BUTTON.**

XmNdialogType

Class	**XmCDialogType**
Type	**unsigned char**
Default	**XmDIALOG_MESSAGE**
Return Values	**XmDIALOG_ERROR** **XmDIALOG_INFORMATION** **XmDIALOG_MESSAGE** **XmDIALOG_QUESTION** **XmDIALOG_WARNING** **XmDIALOG_WORKING**
Access	CSG

Description

The **XmNdialogType** resource specifies the **XmMessageBox** dialog, which determines the default message symbol. The possible values for this resource are as follows:

- **XmDIALOG_ERROR_BUTTON** indicates an **ErrorDialog**.
- **XmDIALOG_INFORMATION** indicates an **InformationDialog**.
- **XmDIALOG_MESSAGE** indicates a **MessageDialog**. This is the default value of the **XmMessageBox** widget dialog type. The default message symbol is a **NULL** value.
- **XmDIALOG_QUESTION** indicates a **QuestionDialog**.
- **XmDIALOG_WARNING** indicates a **WarningDialog**.
- **XmDIALOG_WORKING** indicates a **WorkingDialog**.

XmNhelpLabelString

Class	**XmCXmString**
Type	**XmString**
Default	**"Help"**
Access	CSG

Description

The **XmNhelpLabelString** resource specifies the string label for the help button.

XmNmessageAlignment

Class	**XmCAlignment**
Type	**unsigned char**
Default	**XmALIGNMENT_BEGINNING**
Return Values	**XmALIGNMENT_BEGINNING (the default)** **XmALIGNMENT_CENTER** **XmALIGNMENT_END**
Access	CSG

Description

The **XmNmessageAlignment** resource controls the alignment of the message label. Possible values are as follows:

- **XmALIGNMENT_BEGINNING** indicates the default.
- **XmALIGNMENT_CENTER.**
- **XmALIGNMENT_END.**

XmNmessageString

Class	**XmCXmString**
Type	**XmString**
Default	**NULL**
Access	CSG

Description

The **XmNmessageString** resource specifies the string to be used as the message.

XmNminimizeButtons

Class	**XmCMinimizeButtons**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNminimizeButtons** resource sets the buttons to the width of the widest button and to the height of the tallest button if a **False** value. If a **True** value, the button width and height are set to the preferred size of each button.

XmNokCallback

Class	**XtCallbackList**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNokCallback** resource specifies the list of callbacks that is called when the user clicks on the **OK** button. The callback reason is the **XmCR_OK** value.

XmNokLabelString

Class	**XmCXmString**
Type	**XmString**
Default	**"OK"**
Access	CSG

Description

The **XmNokLabelString** resource specifies the string label for the **OK** button.

XmNsymbolPixmap

Class	**XmCPixmap**
Type	**Pixmap**
Default	**dynamic**
Access	CSG

Description

The **XmNsymbolPixmap** resource specifies the pixmap label to be used as the message symbol.

XmPanedWindow Constraint Resource Set

XmNallowResize **XmNmaximum**

XmNminimum **XmNskipAdjust**

XmNallowResize

Class **XmCBoolean**

Type **Boolean**

Default **False**

Access CSG

Description

The **XmNallowResize** resource allows an application to specify whether the **PanedWindow** widget should allow a pane to request to be resized. This resource only has an effect after the **PanedWindow** widget and its children have been realized. If this resource is set to a **True** value, the **PanedWindow** widget tries to honor requests to alter the height of the pane. If this resource is set to a **False** value, the **PanedWindow** widget always denies pane requests to resize.

XmNpaneMaximum

Class **XmCPaneMaximum**

Type **int**

Default **1000**

Access CSG

Description

The **XmNmaximum** resource allows an application to specify the maximum size to which a pane can be resized. This value must be greater than the specified minimum.

XmNpaneMinimum

Class **XmCPaneMinimum**

Type **int**

Default **1**

Access CSG

Description

The **XmNminumum** resource allows an application to specify the minimum size to which a pane can be resized. This value must be greater than zero.

XmNskipAdjust

Class	**XmCBoolean**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNskipAdjust** resource, when set to a **True** value, allows an application to specify that the **PanedWindow** widget should not automatically resize this pane.

XmPanedWindow Resource Set

XmNmarginHeight
XmNmarginWidth
XmNrefigureMode
XmNsashHeight
XmNsashIndent
XmNsashShadowThickness
XmNsashWidth
XmNseparatorOn
XmNspacing

XmNmarginHeight

Class	**XmCMarginHeight**
Type	**short**
Default	**3**
Access	CSG

Description

The **XmNmarginHeight** resource specifies the distance between the top and bottom edges of the **PanedWindow** widget and its children.

XmNmarginWidth

Class	**XmCMarginWidth**
Type	**short**
Default	**3**
Access	CSG

Description

The **XmNmarginWidth** resource specifies the distance between the left and right edges of the **PanedWindow** widget and its children.

XmNrefigureMode

Class	**XmCBoolean**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNrefigureMode** resource determines whether the positions of the panes are recomputed and repositioned when programmatic changes are being made to the **PanedWindow** widget. Setting this resource to the **True** value resets the children to their appropriate positions.

XmNsashHeight

Class	**XmCSashHeight**
Type	**Dimension**

Default	**10**
Access	CSG

Description

The **XmNsashHeight** resource specifies the height of the sash.

XmNsashIndent

Class	**XmCSashIndent**
Type	**Position**
Default	**–10**
Access	CSG

Description

The **XmNsashIndent** resource specifies the horizontal placement of the sash along each pane. A positive value causes the sash to be offset from the left side of the **PanedWindow** widget, and a negative value causes the sash to be offset from the right side of the **PanedWindow** widget. If the offset is greater than the width of the **PanedWindow** widget minus the width of the sash, the sash is placed flush against the left–hand side of the **PanedWindow** widget.

XmNsashShadowThickness

Class	**XmCShadowThickness**
Type	**int**
Default	**2**
Access	CSG

Description

The **XmNsashShadowThickness** resource specifies the thickness of the sash shadows.

XmNsashWidth

Class	**XmCSashWidth**
Type	**Dimension**
Default	**10**
Access	CSG

Description

The **XmNsashWidth** resource specifies the width of the sash.

XmNseparatorOn

Class	**XmCSeparatorOn**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNseparatorOn** resource determines whether a separator is created between each of the panes. Setting this resource to the **True** value creates a separator at the midpoint between each of the panes.

XmNspacing

Class	**XmCSpacing**
Type	**int**
Default	**8**
Access	CSG

Description

The **XmNspacing** resource specifies the distance between each child pane.

XmPrimitive Resource Set

XmNbottomShadowColor	XmNbottomShadowPixmap
XmNforeground	XmNhelpCallback
XmNhighlightColor	XmNhighlightOnEnter
XmNhighlightPixmap	XmNhighlightThickness
XmNshadowThickness	XmNtopShadowColor
XmNtopShadowPixmap	XmNtraversalOn
XmNunitType	XmNuserData

XmNbottomShadowColor

Class	XmCForeground
Type	Pixel
Default	dynamic
Access	CSG

Description

The **XmNbottomShadowColor** resource specifies the pixmap to use to draw the top and left sides of the border shadow.

XmNbottomShadowPixmap

Class	XmCBottomShadowPixmap
Type	Pixmap
Default	XmUNSPECIFIED_PIXMAP
Access	CSG

Description

The **XmNbottomShadowPixmap** resource specifies the pixmap to use to draw the bottom and right sides of the border shadow.

XmNforeground

Class	XmCForeground
Type	Pixel
Default	dynamic
Access	CSG

Description

The **XmNforeground** resource specifies the foreground drawing color used by the **Primitive** widget.

XmNhelpCallback

Class	XmCCallback
Type	XtCallbackList

Default **NULL**

Access C

Description

The **XmNhelpCallback** resource requests the help key sequence being pressed. The callback reason is the **XmCR_HELP** value.

XmNhighlightColor

Class **XmCForeground**

Type **Pixel**

Default **Black**

Access CSG

Description

The **XmNhighlightColor** resource specifies the color of the enter window and traversal highlight rectangle. This color is used if the highlight pixmap resource is the **NULL** value.

XmNhighlightOnEnter

Class **XmCHighlightOnEnter**

Type **Boolean**

Default **False**

Access CSG

Description

The **XmNhighlightOnEnter** resource specifies whether to draw the border highlight on enter window events. This resource is ignored if the **XmNtraversalOn** resource is set to the **True** value.

XmNhighlightPixmap

Class **XmCHighlightPixmap**

Type **Pixmap**

Default **dynamic**

Access CSG

Description

The **XmNhighlightPixmap** resource specifies the pixmap to use to draw the enter window or traversal highlight rectangle.

XmNhighlightThickness

Class **XmCHighlightThickness**

Type **short**

Default **0**

Access CSG

Description

The **XmNhighlightThickness** resource specifies the size of the border–drawing rectangle used for the enter window and traversal highlight drawing.

XmNshadowThickness

Class	**XmCShadowThickness**
Type	**short**
Default	**0**; **2** if inherited for the **ArrowButton**, **CascadeButton**, **DrawnButton**, **List**, **Primitive**, **PushButton**, **ScrollBar**, **Separator**, or **Text** widget
Access	CSG

Description

The **XmNshadowThickness** resource specifies the size of the drawn border shadow.

XmNtopShadowColor

Class	**XmCBackground**
Type	**Pixel**
Default	**dynamic**
Access	CSG

Description

The **XmNtopShadowColor** resource specifies the pixmap to use to draw the top and left sides of the border shadow. The specified color is used if the **XmNtopShadowPixmap** resource is the **NULL** value.

XmNtopShadowPixmap

Class	**XmCTopShadowPixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNtopShadowPixmap** resource specifies the pixmap to use to draw the top and left sides of the border shadow.

XmNtraversalOn

Class	**XmCTraversalOn**
Type	**Boolean**
Default	**False**; **True** if inherited for the **Text** widget
Access	CSG

Description

The **XmNtraversalOn** resource specifies if the traversal is activated for this widget.

XmNunitType

Class	**XmCUnitType**
Type	**unsigned char**
Default	**XmPIXELS**
Access	CSG

Description

The **XmNunitType** resource provides the basic support for resolution independence. It defines the widget unit types for operation. Possible values include the following:

XmPIXELS – All values provided to the widget are treated as normal pixel values. This is the default for the resource.

Xm100TH_MILLIMETERS – All values provided to the widget are treated as 1/100 of a millimeter.

Xm1000TH_INCHES – All values provided to the widget are treated as 1/1000 of a point.

Xm100TH_POINTS –All values provided to the widget are treated as 1/100 of a point. A point is a unit used in text processing applications and is defined as 1/72 of an inch.

Xm100TH_FONT_UNITS – All values provided to the widget are treated as 1/100 of a font unit. The value used for the font unit is gathered in one of two ways. A global application **XmNbaseFont** resource can be set from which the unit data is extracted. This resource provides a global font to the application, which is then used to calculate the font units. The font unit value becomes the **XmQUAD_WIDTH** property value of the font. A subroutine is also provided to the application that allows it to specify the font unit values.

XmNuserData

Class	**XmCUserData**
Type	**caddr_t**
Default	**NULL**
Access	CSG

Description

The **XmNuserData** resource allows the application to attach any necessary specific data to the widget. It is an internally unused resource.

XmPushButton Resource Set

XmNactivateCallback
XmNarmCallback
XmNarmColor
XmNarmPixmap
XmNdisarmCallback
XmNfillOnArm
XmNshowAsDefault

XmNactivateCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNactivateCallback** resource specifies a callback subroutine that is called when the **PushButton** widget is activated. The **PushButton** widget is activated when the user presses and releases the parent–determined active mouse button while the pointer is inside that widget. Activating the **PushButton** widget also disarms it. The callback reason is the **XmCR_ACTIVATE** value.

XmNarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNarmCallback** resource specifies a callback subroutine that is called when the **PushButton** widget is armed. The **PushButton** widget is armed when the user presses the parent–determined active mouse button while the pointer is inside that widget. The callback reason is the **XmCR_ARM** value.

XmNarmColor

Class	**XmCArmColor**
Type	**Pixel**
Default	**dynamic**
Access	CSG

Description

The **XmNarmColor** resource specifies the color used to fill the armed button. The **XmNfillOnArm** resource must be set to the **True** value for this resource to have an effect. The default for a color display is a color between the background and the bottom shadow color. For a monochrome display, the default is set to the foreground color, and any text in the label appears in the background color when the button is armed.

XmNarmPixmap

Class	**XmCArmPixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNarmPixmap** resource specifies the pixmap to be used as the button face if the **XmNlabeltype** resource is the **XmPIXMAP** value and the **XmPushButton** widget is armed. This resource is disabled when the **PushButton** widget is in a menu.

XmNdisarmCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdisarmCallback** resource specifies a list of callback subroutines that is called when the **PushButton** widget is disarmed. The **PushButton** widget is disarmed when the user presses and releases the parent–determined active mouse button while the pointer is inside that widget. The callback reason is the **XmCR_DISARM** value.

XmNfillOnArm

Class	**XmCFillOnArm**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNfillOnArm** resource forces the **PushButton** widget to fill the background of the button with the color specified by the **XmNarmColor** resource when the button is armed and when this resource is set to a **True** value. If a **False** value, it switches only the top and bottom shadow colors. When the **PushButton** widget is in a menu, this resource is forced to a **False** value.

XmNshowAsDefault

Class	**XmCShowAsDefault**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNshowAsDefault** resource specifies a shadow thickness for a second shadow to be drawn around the **PushButton** widget to visually mark it as a default button. The default value is zero. When this value is not zero, the **Label** widget resources **XmNmarginLeft**, **XmNmarginRight**, **XmNmarginTop**, and **XmNmarginBottom** may be modified to accomodate the second shadow. This resource is disabled when the **PushButton** widget is in a menu.

XmPushButtonGadget Resource Set

XmNactivateCallback
XmNarmCallback
XmNarmColor
XmNarmPixmap
XmNdisarmCallback
XmNfillOnArm
XmNshowAsDefault

XmNactivateCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNactivateCallback** resource specifies a callback subroutine that is called when the **PushButtonGadget** gadget is activated. It is activated when the user presses and releases the parent–determined active mouse button while the pointer is inside the **PushButtonGadget** gadget. Activating the **PushButtonGadget** gadget also disarms it. The callback reason is the **XmCR_ACTIVATE** value.

XmNarmCallback

Class	**XmCCallback**
Type	**caddr_t**
Default	**NULL**
Access	C

Description

The **XmNarmCallback** resource specifies a callback subroutine that is called when the **PushButtonGadget** gadget is armed. It is armed when the user presses the parent–determined active mouse button while the pointer is inside the **PushButtonGadget** gadget. The callback reason is the **XmCR_ARM** value.

XmNarmColor

Class	**XmCArmColor**
Type	**Pixel**
Default	**dynamic**
Access	CSG

Description

The **XmNarmColor** resource specifies the color used to fill the armed button. The **XmNfillOnArm** resource must be set to a **True** value for this resource to have an effect. The default for a color display is a color between the background and the bottom shadow color. For a monochrome display, the default is set to the background color.

XmNarmPixmap

Class	**XmCArmPixmap**
Type	**Pixmap**
Default	**UNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNarmPixmap** resource specifies the pixmap to be used as the button face if the **XmNlabeltype** resource is the **XmPIXMAP** value and the **PushButtonGadget** gadget is armed. This resource is disabled when the **PushButtonGadget** gadget is in a menu.

XmNdisarmCallback

Class	**XmCCallback**
Type	**caddr_t**
Default	**NULL**
Access	C

Description

The **XmNdisarmCallback** resource specifies a callback subroutine that is called when the **PushButtonGadget** gadget is disarmed. The **PushButtonGadget** gadget is disarmed when the user presses and releases the parent–determined active mouse button while the pointer is inside that widget. The callback reason is the **XmCR_DISARM** value.

XmNfillOnArm

Class	**XmCFillOnArm**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNfillOnArm** resource forces the **PushButtonGadget** gadget to fill the background of the button with the color specified by the **XmNarmColor** resource when the button is armed and when this resource is set to a **True** value. If a **False** value, it switches only the top and bottom shadow colors. When the **PushButtonGadget** gadget is in a menu, this resource is forced to a **False** value.

XmNshowAsDefault

Class	**XmCShowAsDefault**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNshowAsDefault** resource specifies a shadow thickness for a second shadow to be drawn around the **PushButtonGadget** gadget to visually mark it as a default button. The default value is zero.

XmRowColumn Resource Set

XmNadjustLast
XmNadjustMargin
XmNentryAlignment
XmNentryBorder
XmNentryCallback
XmNentryClass
XmNisAligned
XmNisHomogeneous
XmNlabelString
XmNmapCallback
XmNmarginHeight
XmNmarginWidth
XmNmenuAccelerator
XmNmenuHelpWidget
XmNmenuHistory
XmNmnemonic
XmNnumColumns
XmNorientation
XmNpacking
XmNpopupEnabled
XmNradioAlwaysOne
XmNradioBehavior
XmNresizeHeight
XmNresizeWidth
XmNrowColumnType
XmNshadowThickness
XmNspacing
XmNsubMenuId
XmNunmapCallback
XmNwhichButton

XmNadjustLast

Class	**XmCAdjustLast**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNadjustLast** resource extends the last row of children to the bottom edge of the **RowColumn** widget (when the **XmOrientation** widget is the **XmHORIZONTAL** value) or extends the last column to the right edge of the **RowColumn** widget (when the **XmOrientation** widget is the **XmVERTICAL** value). This feature is disabled by setting the **XmNadjustLast** resource to the **False** value.

XmNadjustMargin

Class	**XmCAdjustMargin**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNadjustMargin** resource specifies whether the inner minor margins of all items contained within the **RowColumn** widget are forced to the same value. The inner minor margin corresponds to the **XmNmarginLeft**, **XmNmarginRight**, **XmNmarginTop** and **XmNmarginBottom** resources supported by the **XmLabel** widget and the **XmLabelGadget** gadget.

A horizontal orientation forces the **XmNmarginTop** resource and the **XmNmarginBottom** resource for all items in a particular row to the same value; the value is the largest margin specified for one of the **Label** items.

A vertical orientation forces the **XmNmarginLeft** resource and the **XmNmarginRight** resource for all items in a particular column to the same value; the value is the largest margin specified for one of the **Label** items.

This keeps all text within each row or column lined up with all other text in its row or column. If the **XmNrowColumnType** resource is either the **XmMENU_POPUP** value or the **XmMENU_PULLDOWN** value and this resource is the **True** value, only button children have their margins adjusted.

XmNentryAlignment

Class	**XmCAlignment**
Type	**unsigned char**
Default	**dynamic**
Access	CSG

Description

The **XmNentryAlignment** resource specifies the alignment type for **Label** widget or **LabelGadget** gadget children when the **XmNisAligned** resource is enabled. Textual alignment types are as follows:

- **XmALIGNMENT_BEGINNING**, the default
- **XmALIGNMENT_CENTER**
- **XmALIGNMENT_END**

XmNentryBorder

Class	**XmCEntryBorder**
Type	**short**
Default	**dynamic**
Access	CSG

Description

The **XmNentryBorder** resource imposes a uniform border width upon all the children of the **RowColumn** widget. The default value is 0, which disables the feature.

XmNentryCallback

Class	**XtCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNentryCallback** resource disables the activation callbacks for all **ToggleButton**, **CascadeButton**, and **PushButton** items contained within the **RowColumn** widget if the

application supplies this resource; they are then revectored to this callback. This allows an application to supply a single callback routine for handling all items contained in an **RowColumn** widget. The application must supply this resource when this widget is created.

If the application does not supply this resource, the activation callbacks for each item in the **RowColumn** widget will work as normal. The callback reason is the **XmCR_ACTIVATE** value and the default value is the **NULL** value. Changing this resource using the **XtSetValues()** subroutine is not supported.

XmNentryClass

Class	**XmCEntryClass**
Type	**WidgetClass**
Default	**dynamic**
Access	CSG

Description

The **XmNentryClass** resource specifies the only widget class that can be added to the **RowColumn** widget; this resource is meaningful only when the **XmNisHomogeneous** resource is set to the **True** value.

When the **XmNrowColumnType** resource is set to the **XmWORK_AREA** value and the **XmNradioBehavior** resource is the **True** value, then the default value for the **XmNentryClass** resource is the **ToggleButtonGadgetClass** widget.

When the **XmNrowColumnType** resource is set to the **XmMENU_BAR** value, the value of the **XmNentryClass** resource is forced to the **XmCascadeButtonWidgetClass** widget.

XmNisAligned

Class	**XmCIsAligned**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNisAligned** resource specifies text alignment for each item within the **RowColumn** widget; this only applies to items which are a subclass of the **XmLabel** widget class or the **XmLabelGadget** gadget class. If the item is a **Label** widget or gadget and its parent is either a **Popup MenuPane** or a **Pulldown MenuPane**, then alignment is not to be performed; the Label is treated as the title within the MenuPane, and the alignment set by the application is not overridden. The **XmNentryAlignment** controls the type of textual alignment.

XmNisHomogeneous

Class	**XmCIsHomogeneous**
Type	**Boolean**
Default	**dynamic**
Access	CSG

Description

The **XmNisHomogeneous** resource indicates whether the **RowColumn** widget should enforce exact homogeneity among the items it contains; if a **True** value, only the widgets that are of the class indicated by the **XmNentryClass** resource are allowed as children of the **RowColumn** widget. This is most often used when creating a **MenuBar** or a **RadioBox** widget.

Attempting to insert a child which is not a member of the specified class generates a warning message. The default value is the **False** value, except when creating a **MenuBar** or a **RadioBox** widget, when the default is the **True** value.

XmNlabelString

Class	**XtCString**
Type	**XmString**
Default	**NULL**
Access	C

Description

The **XmNlabelString** resource points to a text string that displays the label to the left of the selection area when the **XmNrowColumnType** resource is set to the **XmMENU_OPTION** value. This resource is not meaningful for all other **XmRowColumn** types. If the application wishes to change the label after creation, it must get the **LabelGadget** gadget ID (**XmOptionLabelGadget**) and call the **XtSetValues** subroutine on the **LabelGadget** gadget directly.The default value is no label.

XmNmapCallback

Class	**XtCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNmapCallback** resource specifies a widget–specific callback routine that is invoked when the window associated with the **RowColumn** widget is about to be mapped. The callback reason is the **XmCRMap** resource.

XmNmarginHeight

Class	**XmCMarginHeight**
Type	**Dimension**
Default	**dynamic**
Access	CSG

Description

The **XmNmarginHeight** resource specifies the amount of blank space between the top edge of the **RowColumn** widget and the first item in each column, and the bottom edge of the **RowColumn** widget and the last item in each column. The default value is three pixels.

XmNmarginWidth

Class	**XmCMarginWidth**
Type	**Dimension**
Default	**3**
Access	CSG

Description

The **XmNmarginWidth** resource specifies the amount of blank space between the left edge of the **RowColumn** widget and the first item in each row, and the right edge of the **RowColumn** widget and the last item in each row. The default value is three pixels.

XmNmenuAccelerator

Class	**XmCAccelerators**
Type	**String**
Default	**dynamic**
Access	CSG

Description

The **XmNmenuAccelerator** resource is only useful when the **RowColumn** widget is configured to operate as a Popup MenuPane or a MenuBar. The format of this resource is similar to the left–side specification of a translation string, with the limitation that it must specify a key event. For a Popup MenuPane, when the accelerator is typed by the user, the Popup MenuPane is posted. For a MenuBar, when the accelerator is typed by the user, the first item in the MenuBar is highlighted, and a traversal is enabled in the MenuBar. The default for a Popup MenuPane is the **<Key>F4** value. The default for a MenuBar is the **<Key>F10** value. The accelerator can be disabled by setting the **XmNpopupEnabled** resource to the **False** value.

XmNmenuHelpWidget

Class	**XmCMenuWidget**
Type	**Widget**
Default	**NULL**
Access	CSG

Description

The **XmNmenuHelpWidget** resource specifies the widget ID for the **CascadeButton** widget, which is treated as the Help widget if the **XmNrowColumnType** resource is set to the **XmMENU_BAR** value. The MenuBar always places the Help widget at the lower right corner. If the **RowColumn** widget is any type other than the **XmMENU_BAR** value, this resource is not meaningful.

XmNmenuHistory

Class	**XmCMenuWidget**
Type	**Widget**
Default	**NULL**

Access CSG

Description

The **XmNmenuHistory** resource specifies the widget ID of the last menu entry to be activated. It is also useful for specifying the current selection for an **OptionMenu** label. If the **XmNrowColumnType** resource is set to the **XmMENU_OPTION** value, the specified menu item is positioned under the cursor when the menu is displayed.

If the **RowColumn** widget has the **XmNradioBehavior** resource set to a **True** value, then the widget field associated with this resource contains the widget ID of the last **ToggleButton** widget or **ToggleButtonGadget** gadget to change from unselected to selected. The default value is the widget ID of the first child in the widget.

XmNmnemonic

Class **XmCMnemonic**

Type **char**

Default **dynamic**

Access CSG

Description

The **XmNmnemonic** resource is only useful when the **XmNrowColumnType** resource is set to the **XmMENU_OPTION** value. It specifies a single character that, when typed by the user, posts the associated **Pulldown MenuPane**. The character is underlined if it appears in the **OptionMenu** label, giving the user a visual cue that the character has special functionality associated with it. The default is no mnemonic.

XmNnumColumns

Class **XmCNumColumns**

Type **short**

Default **dynamic**

Access CSG

Description

The **XmNnumColumns** resource specifies the number of minor dimension extensions that are made to accommodate the entries; this resource is only meaningful when the **XmNpacking** resource is set to the **XmPACK_COLUMN** value.

For vertically–oriented **XmRowColumn** widgets, this resource indicates how many columns are built; the number of entries per column is adjusted to maintain this number of columns, if possible.

For horizontally–oriented **XmRowColumn** widgets, this resource indicates how many rows will be built.

The default value is 1.

XmNorientation

Class **XmCOrientation**

Type **unsigned char**

Default	**dynamic**
Access	CSG

Description

The **XmNorientation** resource determines whether **XmRowColumn** layouts are row major or column major. In a column major layout, the children of the **XmRowColumn** are laid out in columns top to bottom within the widget. In a row major layout the children of the **XmRowColumn** are laid out in rows. The **XmVERTICAL** resource value selects a column major layout. The **XmHORIZONTAL** resource value selects a row major layout.

The default value is the **XmVERTICAL** value, except when creating a MenuBar, when the default is the **XmHORIZONTAL** value.

XmNpacking

Class	**XmCPacking**
Type	**unsigned char**
Default	**dynamic**
Access	CSG

Description

The **XmNpacking** resource specifies how to pack the items contained within an **RowColumn** widget. This can be set to the **XmPACK_TIGHT** value, the **XmPACK_COLUMN** value or the **XmPACK_NONE** value. When an **RowColumn** widget packs the items it contains, it determines its major dimension using the value of the **XmNorientation** resource.

The **XmPACK_TIGHT** value indicates that given the current major dimension (for example, vertical if the **XmNorientation** resource is the **XmVERTICAL** value), entries are placed one after the other until the **RowColumn** widget must wrap. The **RowColumn** widget wraps when there is no room left for a complete child in that dimension. Wrapping occurs by beginning a new row or column in the next available space. Wrapping continues, as often as necessary, until all of the children are laid out. In the vertical dimension (columns), boxes are set to the same width; in the horizontal dimension (rows), boxes are set to the same depth. Each entry's position in the major dimension is left unaltered (for example, the **XmNy** resource is unchanged when the **XmNorientation** resource is the **XmVERTICAL** value); its position in the minor dimension is set to the same value as the greatest entry in that particular row or column. The position in the minor dimension of any particular row or column is independent of all other rows or columns.

The **XmPACK_COLUMN** value indicates that all entries are placed in identically sized boxes. The box is based on the largest height and width values of all the children widgets. The value of the **XmNnumColumns** resource determines how many boxes are placed in the major dimension before extending in the minor dimension.

The **XmPACK_NONE** value indicates that no packing is performed. The x and y resources of each entry are left alone, and the **XmRowColumn** widget attempts to become large enough to enclose all entries.

The default value is the **XmPACK_TIGHT** value, except when building an OptionMenu or a **RadioBox** widget, where the default is the **XmPACK_COLUMN** value.

XmNpopupEnabled

Class	**XmCPopupEnabled**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNpopupEnabled** resource allows the menu system to enable keyboard input (accelerators and mnemonics) defined for the **Popup MenuPane** and any of its submenus. The **Popup MenuPane** needs to be informed whenever its accessibility to the user changes because posting of the **Popup MenuPane** is controlled by the application. The default value for this resource is the **True** value (keyboard input—accelerators and mnemonics—defined for the **Popup MenuPane** and any of its submenus is enabled).

XmNradioAlwaysOne

Class	**XmCRadioAlwaysOne**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNradioAlwaysOne** resource forces the active ToggleButton or ToggleButtonGadget to be automatically selected after having been unselected (if no other toggle was activated), if a **True** value. If a **False** value, the active toggle may be unselected. The default value is the **True** value. This resource is only important when the **XmNradioBehavior** resource is a **True** value.

The application always has the freedom to add and subtract toggles from the **RowColumn** widget regardless of the selected/unselected state of the toggle. The application also has the freedom to manage and unmanage toggle children of **RowColumn** widget at any time regardless of state. Because of these freedoms, there are cases in which it is possible for the application to create a **RowColumn** widget that has the **XmNradioAlwaysOne** resource set to the **True** value and none of the toggle children selected.

XmNradioBehavior

Class	**XmCRadioBehavior**
Type	**Boolean**
Default	**False**
Access	CSG

Description

When the **XmNradioBehavior** resource specifies a Boolean value that is the **True** value, it indicates that the **RowColumn** widget should enforce a radio box type behavior on all of its children which are **ToggleButton** widgets or **ToggleButtonGadget** gadgets.

Two resources in the **ToggleButton** widget and the **ToggleButtonGadget** gadget are forced to specified values at creation time. The **XmNindicator** resource is forced to the

XmONE_OF_MANY value, and the **XmNinvisibleWhenOff** resource is forced to the **True** value.

Radio box behavior dictates that when one toggle is selected and another toggle is selected, the first toggle is unselected automatically. The default value is the **False** value, except when creating a **RadioBox** widget, when the default is the **True** value.

XmNresizeHeight

Class	**XmCResizeHeight**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNresizeHeight** resource requests a new height if necessary, when set to a **True** value. When set to a **False** value, the widget does not request a new height regardless of any changes to the widget or its children.

XmNresizeWidth

Class	**XmCResizeWidth**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNresizeWidth** resource requests a new width if necessary, when set to a **True** value. When set to a **False** value, the widget does not request a new width regardless of any changes to the widget or its children.

XmNrowColumnType

Class	**XmCRowColumnType**
Type	**unsigned char**
Default	**XmWORK_AREA**
Access	CG

Description

The **XmNrowColumnType** resource specifies the type of the **RowColumn** widget that is to be created. It is a non–standard resource that cannot be changed after it is set. If an application uses any of the convenience routines except its **CreateRowColumn** subroutine, this resource is automatically forced to the appropriate value by the convenience routine. If an application uses the **XtIntrinsics** API to create its **RowColumn** widgets, it must specify this resource. The set of possible settings for this resource are:

- **XmWORK_AREA**
- **XmMENU_BAR**
- **XmMENU_PULLDOWN**

- **XmMENU_POPUP**
- **XmMENU_OPTION**

The default value is **XmWORK_AREA**.

This resource cannot be changed after the **RowColumn** widget is created. Any changes attempted through the **XtSetValues** subroutine will be ignored.

XmNshadowThickness

Class **XmCShadowThickness**

Type **int**

Default **dynamic**

Access CSG

Description

The **XmNshadowThickness** resource specifies the width of the shadow drawn just inside the borders of the **BulletinBoard** widget. If the parent is a **DialogShell** widget, then the default is 1 pixel; otherwise, it is zero.

XmNspacing

Class **XmCSpacing**

Type **short**

Default **dynamic**

Access CSG

Description

The **XmNspacing** resource specifies the horizontal and vertical spacing between items contained within the **XmRowColumn** widget. The default value is 1 pixel, except for a horizontal MenuBar, which defaults to zero pixels.

XmNsubMenuId

Class **XmCMenuWidget**

Type **Widget**

Default **NULL**

Access CG

Description

The **XmNsubMenuId** resource specifies the widget ID for the Pulldown MenuPane associated with an OptionMenu. This resource is only useful when the **XmNrowColumnType** resource is set to the **XmMENU_OPTION** value. This resource is unused for all other **XmRowColumn** resource set types. The default value is the **NULL** value.

XmNunmapCallback

Class **XtCCallback**

Type **XtCallbackList**

Default	**NULL**
Access	C

Description

The **XmNunmapCallback** resource specifies a widget–specific callback routine that is invoked after the window associated with the **XmRowColumn** widget is unmapped. The callback reason is the **XmCR_Unmap** value. The default value is **NULL**.

XmNwhichButton

Class	**XmCWhichButton**
Type	**unsigned int**
Default	**dynamic**
Access	CSG

Description

The **XmNwhichButton** resource specifies the mouse button to which a menu system is sensitive. The default for the **XmMENU_POPUP** menu system is the right mouse button (Button 3) value. The default for the **XmMENU_OPTION** and **XmMENU_BAR** menu systems is the left mouse button (Button 1) value. This resource is not useful for **RowColumn** widgets of the **XmWORK_AREA** and **XmMENU_PULLDOWN** types.

XmRowColumn Special Menu Resource Set

XmNmenuCursor

XmNmenuCursor

Class	**XmCCursor**
Type	**String**
Default	**arrow**
Access	C

Description

The **XmNmenuCursor** resource sets a variable that controls the cursor used whenever this application posts a menu. This resource can only be specified once at application start up time, either by placing it within a defaults file or by using the **–xrm** command line argument (for example, `myProg –xrm "*menuCursor: arrow"`).

The menu cursor can also be selected programmatically by using the **XmSetMenuCursor** subroutine. The following list contains the acceptable cursor names. If the application does not specify a cursor or if an invalid name is supplied, the default cursor (an arrow pointing up and to the right) is used.

X_cursor
arrow
based_arrow_down
based_arrow_up
boat
bogosity
bottom_left_corner
bottom_right_corner
bottom_side
bottom_tee
box_spiral
center_ptr
circle
clock
coffee_mug
cross
cross_reverse
crosshair
diamond_cross
dot
dotbox
double_arrow
draft_large
lr_angle
man
middlebutton
mouse
pencil
pirate
plus
question_arrow
right_ptr
right_side
right_tee
rightbutton
rtl_logo
sailboat
sb_down_arrow
sb_h_double_arrow
sb_left_arrow
sb_right_arrow
sb_up_arrow
sb_v_double_arrow
shuttle
sizing
spider

draft_small
draped_box
exchange
fleur
gobbler
gumby
hand1
hand2
heart
icon
iron_cross
left_ptr
left_side
left_tee
leftbutton
ll_angle
spraycan
star
target
tcross
top_left_arrow
top_left_corner
top_right_corner
top_side
top_tee
trek
ul_angle
umbrella
ur_angle
watch
xterm

XmScale Resource Set

XmNdecimalPoints
XmNdragCallback
XmNfontList
XmNhighlightOnEnter
XmNhighlightThickness
XmNmaximum
XmNminimum
XmNorientation
XmNprocessingDirection
XmNscaleHeight
XmNscaleWidth
XmNshowValue
XmNtitleString
XmNtraversalOn
XmNvalue
XmNvalueChangedCallback

XmNdecimalPoints

Class	**XmCDecimalPoints**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNdecimalPoints** resource specifies the number of decimal points to shift the slider value while displaying it. For example, a slider value of 2,350 and an **XmdecimalPoints** resource set value of 2 results in a display value of 23.50.

XmNdragCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdragCallback** resource specifies the list of callbacks that is called when the slider position changes when the slider is dragged. The callback reason is the **XmCR_DRAG** value.

XmNfontList

Class	**XmCFontList**
Type	**XmFontList**
Default	**"Fixed"**
Access	CSG

Description

The **XmNfontList** resource specifies the font list to use for the title text string specified by the **XmNtitleString** resource.

XmNhighlightOnEnter

Class	**XmCHighlightOnEnter**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNhighlightOnEnter** resource specifies whether to draw the border highlight on enter window events. This resource is ignored if the **XmNtraversalOn** resource is set to a a **True** value.

XmNhighlightThickness

Class	**XmCHighlightThickness**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNhighlightThickness** resource specifies the size of the border drawing rectangle used for enter window and traversal highlight drawing.

XmNmaximum

Class	**XmCMaximum**
Type	**int**
Default	**100**
Access	CSG

Description

The **XmNmaximum** resource specifies the slider maximum value.

XmNminimum

Class	**XmCMinimum**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNminimum** resource specifies the slider minimum value.

XmNorientation

Class	**XmCOrientation**
Type	**unsigned char**
Default	**XmVERTICAL**

Access CSG

Description

The **XmNorientation** resource displays scale vertically or horizontally. This resource can have the **XmVERTICAL** value or the **XmHORIZONTAL** value.

XmNprocessingDirection

Class **XmCProcessingDirection**

Type **unsigned char**

Default **XmMAX_ON_TOP**

Access CSG

Description

The **XmNprocessingDirection** resource specifies whether the value for the **XmNmaximum** resource is on the right or the left side of the **XmNminimum** resource for horizontal scales or above or below the **XmNminimum** resource for vertical scales. This resource can have the following values:

- **XmMAX_ON_TOP**
- **XmMAX_ON_BOTTOM**
- **XmMAX_ON_LEFT**
- **XmMAX_ON_RIGHT**

XmNscaleHeight

Class **XmCScaleHeight**

Type **Dimension**

Default **0**

Access CSG

Description

The **XmNscaleHeight** resource specifies the height of the slider area. The value is in the specified unit type. The default value is pixels.

XmNscaleWidth

Class **XmCScaleWidth**

Type **Dimension**

Default **0**

Access CSG

Description

The **XmNscaleWidth** resource specifies the width of the slider area. The value is in the specified unit type. The default value is pixels.

XmNshowValue

Class **XmCShowValue**

Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNshowValue** resource specifies whether a label for the current slider value is displayed next to the slider. If it is a **True** value, the current slider value is displayed.

XmNtitleString

Class	**XmCTitleString**
Type	**XmString**
Default	**NULL**
Access	CSG

Description

The **XmNtitleString** resource specifies the title text string that is displayed in the scale widget window.

XmNtraversalOn

Class	**XmCTraversalOn**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNtraversalOn** resource specifies whether the scale slider is to have traversal on for it.

XmNvalue

Class	**XmCValue**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNvalue** resource specifies the current position of the slider along the scale, between minimum and maximum values.

XmNvalueChangedCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNvalueChangedCallback** resource specifies a list of callback subroutines that is called when the value of the slider has changed. The callback reason is the **XmCR_VALUE_CHANGED** value.

XmScrollBar Resource Set

XmNdecrementCallback
XmNdragCallback
XmNincrement
XmNincrementCallback
XmNinitialDelay
XmNmaximum
XmNminimum
XmNorientation
XmNpageDecrementCallback
XmNpageIncrement
XmNprocessingDirection
XmNrepeatDelay
XmNshowArrows
XmNsliderSize
XmNtoBottomCallback
XmNtoTopCallback
XmNvalue
XmNvalueChangedCallback

XmNdecrementCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdecrementCallback** resource specifies the list of callbacks that is called when an arrow is selected that decreases the slider value by one increment. The callback reason is the **XmCR_DECREMENT** value.

XmNdragCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdragCallback** resource specifies the list of callbacks that is called on each incremental change of position when the slider is being dragged. The callback reason is the **XmCR_DRAG** value.

XmNincrement

Class	**XmCIncrement**
Type	**int**
Default	**1**
Access	CSG

Description

The **XmNincrement** resource specifies the amount to move the slider when the corresponding arrow is selected.

XmNincrementCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNincrementCallback** resource specifies the list of callbacks that is called when an arrow is selected that increases the slider value by one increment. The callback reason is the **XmCR_INCREMENT** value.

XmNinitialDelay

Class	**XmCInitialDelay**
Type	**int**
Default	**250**
Access	CSG

Description

The **XmNinitialDelay** resource specifies the amount of time to wait (in milliseconds) before starting continuous slider movement while an arrow or the scroll region is pressed.

XmNmaximum

Class	**XmCMaximum**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNmaximum** resource specifies the slider maximum value.

XmNminimum

Class	**XmCMinimum**
Type	**int**
Default	**0**
Access	CSG

Description

The **XmNminimum** resource specifies the slider minimum value.

XmNorientation

Class	**XmCOrientation**
Type	**unsigned char**
Default	**XmVERTICAL**

Access CSG

Description

The **XmNorientation** resource specifies whether the ScrollBar is displayed vertically or horizontally. This resource can have the **XmVERTICAL** value or the **XmHORIZONTAL** value.

XmNpageDecrementCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access CSG

Description

The **XmNpageDecrementCallback** resource specifies the list of callbacks that is called when the slider area is selected and the slider value is decreased by a one–page increment. The the callback reason is the **XmCR_PAGE_DECREMENT** value.

XmNpageIncrement

Class **XmCPageIncrement**

Type **int**

Default **10**

Access C

Description

The **XmNpageIncrement** resource specifies the amount to move the slider when selection occurs on the slide area.

XmNpageIncrementCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNpageIncrementCallback** resource specifies the list of callbacks that is called when the slider area is selected and the slider value is increased by one page increment. The callback reason is the **XmCR_PAGE_DECREMENT** value.

XmNprocessingDirection

Class **XmCProcessingDirection**

Type **unsigned char**

Default **XmMAX_ON_BOTTOM**

Access CSG

Description

The **XmNprocessingDirection** resource specifies whether the value for the **XmNmaximum** resource should be on the right or the left side of the **XmNminimum** resource for horizontal ScrollBars or above or below the **XmNminimum** resource for vertical ScrollBars. This resource can have the following values:

- **XmMAX_ON_TOP**
- **XmMAX_ON_BOTTOM**
- **XmMAX_ON_LEFT**
- **XmMAX_ON_RIGHT**

XmNrepeatDelay

Class	**XmCRepeatDelay**
Type	**int**
Default	**50**
Access	CSG

Description

The **XmNrepeatDelay** resource specifies the amount of time to wait (in milliseconds) between subsequent slider movements after the **XmNinitialDelay** resource is processed.

XmNshowArrows

Class	**XmCShowArrows**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNshowArrows** resource specifies whether the arrows are displayed.

XmNsliderSize

Class	**XmCSliderSize**
Type	**int**
Default	**10**
Access	CSG

Description

The **XmNsliderSize** resource specifies the size of the slider between the values of zero and maximum – minimum.

XmNtoBottomCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**

Access C

Description

The **XmNtoBottomCallback** resource specifies the list of callbacks that are called when the user selects **<Shift>** mouse button one down. This callback sends as a value the maximum ScrollBar value minus the ScrollBar slider size. The slider location is not automatically repositioned. The callback reason is the **XmCR_TO_TOP** value.

XmNtoTopCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNtoTopCallback** resource specifies the list of callbacks that are called when the user selects **<Shift>** left mouse button down in the top arrow button. This callback sends as a value the minimum ScrollBar slider value. The slider location is not automatically repositioned. The callback reason is the **XmCR_TO_TOP** value.

XmNvalue

Class **XmCValue**

Type **int**

Default **0**

Access CSG

Description

The **XmNvalue** resource specifies the slider position between the minimum and maximum.

XmNvalueChangedCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNvalueChangedCallback** resource specifies the list of callbacks that is called when the slider is released while being dragged; this is in place of the **XmNincrementCallback**, the **XmNdecrementCallback**, the **XmNpageIncrementCallback** or the **XmNpageDecrementCallback** resource when they do not have any callbacks attached. The callback reason is the **XmCR_VALUE_CHANGED** value.

XmScrolledList Resource Set

XmNhorizontalScrollBar
XmNlistSizePolicy
XmNscrollBarDisplayPolicy
XmNscrollBarPlacement
XmNscrolledWindowMarginHeight
XmNscrolledWindowMarginWidth
XmNspacing
XmNverticalScrollBar

XmNhorizontalScrollBar

Class	**XmCHorizontalScrollBar**
Type	**Widget**
Default	**NULL**
Access	CSG

Description

The **XmNhorizontalScrollBar** resource specifies the widget ID of the horizontal ScrollBar. This widget is created automatically by the **XmCreateScrolledList** convenience subroutine.

XmNlistSizePolicy

Class	**XmCListSizePolicy**
Type	**unsigned char**
Default	**XmVARIABLE**
Access	CG

Description

The **XmNlistSizePolicy** resource controls the reaction of the **List** widget when an item grows horizontally beyond the current size of the **List** widget work area. If the value is the **XmCONSTANT** value, the **List** widget viewing area does not grow, and a horizontal ScrollBar is added. If this resource is set to the **XmVARIABLE** value, the **List** widget grows to match the size of the longest item, and no horizontal ScrollBar is displayed.

When the value of this resource is the **XmRESIZE_IF_POSSIBLE** value, the **List** widget attempts to grow or shrink to match the width of the widest item. If it cannot grow to match the widest size, a horizontal ScrollBar is added if the longest item is wider than the **XmList** viewing area.

The size policy must be set at the time the **List** widget is created. It cannot be changed at a later time through the **XtSetValues** subroutine.

XmNscrollBarDisplayPolicy

Class	**XmCScrollBarDisplayPolicy**
Type	**unsigned char**
Default	**XmAS_NEEDED**
Access	CSG

Description

The **XmNscrollBarDisplayPolicy** resource specifies the ScrollBar display policy. When this resource is set to the **XmAS_NEEDED** value, the vertical ScrollBar is displayed only when the number of items in the list exceeds the number of visible items. If the **XmNlistSizePolicy** resource is the **XmCONSTANT** value or the **XmRESIZE_IF_POSSIBLE** value, the horizontal ScrollBar is displayed only if there is an item that is wider than the current width of the list. When this resource is set to the **XmSTATIC** value, the vertical ScrollBar is always displayed. The horizontal ScrollBar is always displayed if the **XmNlistSizePolicy** resource is set to the **XmCONSTANT** value or the **XmRESIZE_IF_POSSIBLE** value.

XmNscrollBarPlacement

Class	**XmCScrollBarPlacement**
Type	**unsigned char**
Default	**XmBOTTOM_RIGHT**
Access	CSG

Description

The **XmNscrollBarPlacement** resource specifies the positioning of the ScrollBars in relation to the visible items. The following are the values:

- **XmTOP_LEFT** – The horizontal ScrollBar is placed above the visible items and the vertical ScrollBar is placed to the left of the visible items.
- **XmBOTTOM_LEFT** – The horizontal ScrollBar is placed below the visible items and the vertical ScrollBar is placed to the left of the visible items.
- **XmTOP_RIGHT** – The horizontal ScrollBar is placed above the visible items and the vertical ScrollBar is placed to the right of the visible items.
- **XmBOTTOM_RIGHT** – The horizontal ScrollBar is placed below the visible items and the vertical ScrollBar is placed to the right of the visible items.

XmNscrolledWindowMarginHeight

Class	**XmCScrolledWindowMarginHeight**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNscrolledWindowMarginHeight** resource specifies the margin height on the top and the bottom of the scrolled window.

XmNscrolledWindowMarginWidth

Class	**XmCScrolledWindowMarginWidth**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNscrolledWindowMarginWidth** resource specifies the margin width on the right and the left sides of the scrolled window.

XmNspacing

Class **XmCSpacing**

Type **Dimension**

Default **4**

Access CSG

Description

The **XmNspacing** resource specifies the distance that separates the ScrollBars from the visible items.

XmNverticalScrollBar

Class **XmCVerticalScrollBar**

Type **Widget**

Default **NULL**

Access CSG

Description

The **XmNverticalScrollBar** resource specifies the widget ID of the vertical ScrollBar. This widget is created automatically by the **XmCreateScrolledList** convenience subroutine.

XmScrolledWindow Resource Set

XmNclipWindow
XmNhorizontalScrollBar
XmNscrollBarDisplayPolicy
XmNscrollBarPlacement
XmNscrolledWindowMarginHeight
XmNscrolledWindowMarginWidth
XmNscrollingPolicy
XmNspacing
XmNverticalScrollBar
XmNvisualPolicy
XmNworkWindow

XmNclipWindow

Class	**XmCClipWindow**
Type	**Widget**
Default	**NULL**
Access	G

Description

The **XmNclipWindow** resource specifies the widget ID of the clipping area. This is automatically created by the **ScrolledWindow** widget when the **XmNvisualPolicy** resource is set to the **XmCONSTANT** value and can only be read by the application. Any attempt to set this resource to a new value causes a warning message to be printed by the scrolled window. If the **XmNvisualPolicy** resource is set to the **XmVARIABLE** value, this resource is set to the **NULL** value, and no clipping window is created.

XmNhorizontalScrollBar

Class	**XmCHorizontalScrollBar**
Type	**Widget**
Default	**NULL**
Access	CSG

Description

The **XmNhorizontalScrollBar** resource specifies the widget ID of the horizontal **ScrollBar** widget.

XmNscrollBarDisplayPolicy

Class	**XmCScrollBarDisplayPolicy**
Type	**unsigned char**
Default	**XmSTATIC**
Return Value	**XmAS_NEEDED**
Access	CG

Description

The **XmNscrollBarDisplayPolicy** resource controls the automatic placement of the ScrollBars. If it is set to the **XmAS_NEEDED** value and if the **XmNscrollingPolicy** resource is set to the **XmAUTOMATIC** value, ScrollBars are only displayed if the workspace exceeds

the clip area in one or both dimensions. An resource value of the **XmSTATIC** value causes the **ScrolledWindow** widget to display the ScrollBars whenever they are managed, regardless of the relationship between the clip window and the work area. This resource must be the **XmSTATIC** value when the **XmNscrollingPolicy** resource is the **XmAPPLICATION_DEFINED** value.

XmNscrollBarPlacement

Class	**XmCScrollBarPlacement**
Type	**unsigned char**
Default	**XmBOTTOM_RIGHT**
Return Values	**XmTOP_LEFT** **XmBOTTOM_LEFT** **XmTOP_RIGHT** **XmBOTTOM_RIGHT**
Access	CSG

Description

The **XmNscrollBarPlacement** resource specifies the positioning of the ScrollBars in relation to the work window. The values are as follows:

- **XmTOP_LEFT** – The horizontal ScrollBar is placed above the work window, and the vertical ScrollBar to the left of the work window.
- **XmBOTTOM_LEFT** – The horizontal ScrollBar is placed below the work window, and the vertical ScrollBar to the left of the work window.
- **XmTOP_RIGHT** – The horizontal ScrollBar is placed above the work window, and the vertical ScrollBar to the right of the work window.
- **XmBOTTOM_RIGHT** – The horizontal ScrollBar is placed below the work window, and the vertical ScrollBar to the right of the work window.

XmNscrolledWindowMarginHeight

Class	**XmCScrolledWindowMarginHeight**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNscrolledWindowMarginHeight** resource specifies the margin height on the top and bottom of the **ScrolledWindow** widget.

XmNscrolledWindowMarginWidth

Class	**XmCScrolledWindowMarginWidth**
Type	**Dimension**
Default	**0**
Access	CSG

Description

The **XmNscrolledWindowMarginWidth** resource specifies the margin width on the right and left sides of the **ScrolledWindow** widget.

XmNscrollingPolicy

Class	**XmCScrollingPolicy**
Type	**unsigned char**
Default	**XmAPPLICATION_DEFINED**
Return Values	**XmAUTOMATIC** **XmCONSTANT** **XmAS_NEEDED** **XmAPPLICATION_DEFINED**
Access	CG

Description

The **XmNscrollingPolicy** resource performs automatic scrolling of the work area with no application interaction. If the value of this resource is the **XmAUTOMATIC** value, the **ScrolledWindow** widget automatically creates the Scrollbars, attaches callbacks to the Scrollbars, sets the visual policy to the **XmCONSTANT** value, sets the Scrollbar display policy to the **XmAS_NEEDED** value, and automatically moves the work area through the clip window in response to any user interaction with the Scrollbars. An application can also add its own callbacks to the Scrollbars. This allows the application to be notified of a scroll event without having to perform any layout procedures.

NOTE: Since the **ScrolledWindow** widget adds callbacks to the Scrollbars, an application should not perform an **XtRemoveAllCallbacks** subroutine on any of the **XmScrollbar** widgets.

When the **XmNscrollingPolicy** resource is set to the **XmAPPLICATION_DEFINED** default value, the application is responsible for all aspects of scrolling. The Scrollbars must be created by the application, and it is responsible for performing any visual changes in the work area in response to user input.

This resource must be set to the desired policy at the time the **ScrolledWindow** widget is created. It cannot be changed through the **SetValues()** facilitator.

XmNspacing

Class	**XmCSpacing**
Type	**Dimension**; **int** if inherited for the **MainWindow** widget
Default	**4**
Access	CSG

Description

The **XmNspacing** resource specifies the distance that separates the ScrollBars from the work window.

XmNverticalScrollBar

Class	**XmCVerticalScroll bar**
Type	**Widget**
Default	**NULL**
Access	CSG

Description

The **XmNverticalScrollBar** resource specifies the widget ID of the vertical ScrollBar.

XmNvisualPolicy

Class	**XmCVisualPolicy**
Type	**unsigned char**
Default	**XmVARIABLE**
Return Values	**XmVARIABLE** **XmCONSTANT**
Access	CG

Description

The **XmNvisualPolicy** resource increases the size of the **ScrolledWindow** widget to match the size of the work area, or it is used as a static viewport onto a larger data space. If the visual policy is the **XmVARIABLE** value, the **ScrolledWindow** widget forces the ScrollBar display policy to the **XmSTATIC** value, allows the work area to grow or shrink at any time, and adjusts its layout to accommodate the new size. When the policy is the **XmCONSTANT** value, the work area is allowed to grow or shrink as requested, but a clipping window forces the size of the visible portion to remain constant. The only time the viewing area can grow is in response to a resize operation from the parent of the **ScrolledWindow** widget.

NOTE: This resource must be set to the desired policy at the time the **ScrolledWindow** widget is created. It cannot be changed through the **SetValues()** facilitator.

XmNworkWindow

Class	**XmCWorkWindow**
Type	**Widget**
Default	**NULL**
Access	CSG

Description

The **XmNworkWindow** resource specifies the widget ID of the viewing area.

XmSelectionBox Resource Set

XmNapplyCallback
XmNcancelCallback
XmNdialogType
XmNlistItemCount
XmNlistLabelString
XmNminimizeButtons
XmNnoMatchCallback
XmNokLabelString
XmNtextAccelerators
XmNtextString
XmNapplyLabelString
XmNcancelLabelString
XmNhelpLabelString
XmNlistItems
XmNlistVisibleItemCount
XmNmustMatch
XmNokCallback
XmNselectionLabelString
XmNtextColumns

XmNapplyCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C; N/A if inherited for the **Command** widget

Description

The **XmNapplyCallback** resource specifies the list of callback subroutines that is called when the user clicks on the **Apply** button. The callback reason is the **XmCR_APPLY** value.

XmNapplyLabelString

Class	**XmCApplyLabelString**
Type	**XmString**
Default	**"Apply"**; **"Filter"** if inherited for the **FileSelectionBox** widget
Access	CSG; N/A if inherited for the **Command** widget

Description

The **XmNapplyLabelString** resource specifies the string label for the **Apply** button.

XmNcancelCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	CSG; N/A if inherited for the **Command** widget

Description

The **XmNcancelCallback** resource invokes this **Callback** subroutine when the user clicks on the cancel button. The callback reason is the **XmCR_CANCEL** value.

XmNcancelLabelString

Class **XmCXmString**

Type **XmString**

Default **"Cancel"**

Access CSG; N/A if inherited for the **Command** widget

Description

The **XmNcancelLabelString** resource specifies the string label for the **Cancel** button.

XmNdialogType

Class **XmCDialogType**

Type **unsigned char**

Default **dynamic**; **XmDIALOG_COMMAND** if inherited for the **Command** widget; **XmDIALOG_FILE_SELECTION** if inherited for the **FileSelectionBox** widget

Access CG; G if inherited for the **Command** widget

Description

The **XmNdialogType** resource determines the set of **SelectionBox** children widgets which are created and managed at initialization. The following are possible values:

- **XmDIALOG_PROMPT** – the list and list labels are not created, and the **Apply** button is unmanaged.
- **XmDIALOG_SELECTION** – all standard children are created and managed except the **Apply** button.
- **XmDIALOG_WORK_AREA** – all standard children are created and managed.

If the parent of the **SelectionBox** widget is a **DialogShell** widget, the default is the **XmDIALOG_SELECTION** value; otherwise, the default is the **XmDIALOG_WORK_AREA** value. The **XmCreatePromptDialog** and **XmCreateSelectionDialog** subroutines set and append this resource to the creation *ArgumentList* supplied by the application. This resource cannot be modified after creation.

XmNhelpLabelString

Class **XmCXmString**

Type **XmString**

Default **"Help"**

Access CSG; N/A if inherited for the **Command** widget

Description

The **XmNhelpLabelString** resource specifies the string label for the **Help** button.

XmNlistItemCount

Class	**XmCItemCount**
Type	**int**
Default	**0**
Access	CSG; N/A if inherited for the **Command** widget

Description

The **XmNlistItemCount** resource specifies the number of items in the **SelectionBox** list.

XmNlistItems

Class	**XmCItems**
Type	**XmStringList**
Default	**NULL**
Access	CSG; N/A if inherited for the **Command** widget

Description

The **XmNlistItems** resource specifies the items in the **SelectionBox** list.

XmNlistLabelString

Class	**XmCXmString**
Type	**XmString**
Default	**NULL**; **"Files"** if inherited for the **FileSelectionBox** widget
Access	CSG; N/A if inherited for the **Command** widget

Description

The **XmNlistLabelString** resource specifies the string label to appear above the **SelectionBox** list containing the selection items.

XmNlistVisibleItemCount

Class	**XmCVisibleItemCount**
Type	**int**
Default	**8**
Access	CSG; N/A if inherited for the **Command** widget

Description

The **XmNlistVisibleItemCount** resource specifies the number of items displayed in the **SelectionBox** list.

XmNminimizeButtons

Class **XmCMinimizeButtons**

Type **Boolean**

Default **False**

Access CSG; N/A if inherited for the **Command** widget

Description

The **XmNminimizeButtons** resource sets the buttons to the width of the widest button and the height of the tallest button if it is the **False** value. If this resource is the **True** value, button width and height are not modified.

XmNmustMatch

Class **XmCMustMatch**

Type **Boolean**

Default **False**

Access CSG; N/A if inherited for the **Command** widget

Description

The **XmNmustMatch** resource specifies whether the selection widget should check if the user selection in the text edit field has an exact match in the **SelectionBox** list. If the selection does not have an exact match, and if the **XmNmustMatch** resource is a **True** value, the **XmNnoMatchCallback** resource is activated. If the selection does have an exact match, either the **XmNapplyCallback** resource or the **XmNokCallback** resource is activated.

XmNnoMatchCallback

Class **XmCCallback**

Type **XtCallbackList**

Default **NULL**

Access C; N/A if inherited for the **Command** widget

Description

The **XmNnoMatchCallback** resource specifies the list of callback subroutines that is called when the user makes a selection from the text edit field that does not have an exact match with any of the items in the list box. The callback reason is the **XmCR_NO_MATCH** value. Callback subroutines in this list are called only if the **XmNmustMatch** resource has a **True** value.

XmNokCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	**C**; N/A if inherited for the **Command** widget

Description

The **XmNokCallback** resource specifies the list of callback subroutines that is called when the user clicks the **OK** button. The callback reason is the **XmCR_OK** value.

XmNokLabelString

Class	**XmCXmString**
Type	**XmString**
Default	**"OK"**
Access	CSG; N/A if inherited for the **Command** widget

Description

The **XmNokLabelString** resource specifies the string label for the **OK** button.

XmNselectionLabelString

Class	**XmCXmString**
Type	**XmString**
Default	**"Selection"**
Access	CSG

Description

The **XmNselectionLabelString** resource specifies the string label for the selection text edit field.

XmNtextAccelerators

Class	**XmCTextAccelerators**
Type	**XtTranslations**
Default	(see below)
Access	C

Description

The **XmNtextAccelerators** resource specifies translations added to the **Text** widget child of the **SelectionBox** widget. The default includes bindings for the up and down keys for auto selection of list items; it also includes the normal accelerator translations defined by the **BulletinBoard** widget for dialog components.

XmNtextColumns

Class **XmCTextColumns**

Type **int**

Default **20; 31** if inherited for the **FileSelectionBox** widget

Access CSG

Description

The **XmNtextColumns** resource specifies the number of columns in the **Text** widget.

XmNtextString

Class **XmCTextString**

Type **XmString**

Default **NULL**

Access CSG; N/A if inherited for the **Command** widget

Description

The **XmNtextString** specifies the text in the text edit selection field.

XmSeparator Resource Set

XmNmargin **XmNseparatorType**

XmNorientation

XmNmargin

Class	**XmCMargin**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNmargin** resource specifies the space on the left and right sides between the border of the **Separator** widget and the line drawn for horizontal orientation. For vertical orientation, this resource specifies the space on the top and bottom between the border of the **Separator** widget and the line drawn.

XmNorientation

Class	**XmCOrientation**
Type	**unsigned_char**
Default	**XmHORIZONTAL**
Return Values	**XmHORIZONTAL** **XmVERTICAL**
Access	CSG

Description

The **XmNorientation** resource displays the **Separator** widget vertically or horizontally. This resource can have the **XmVERTICAL** and **XmHORIZONTAL** values.

XmNseparatorType

Class	**XmCSeparatorType**
Type	**unsigned_char**
Default	**XmSHADOW_ETCHED_IN**
Return Values	**XmSINGLE_LINE** **XmDOUBLE_LINE** **XmSINGLE_DASHED_LINE** **XmDOUBLE_DASHED_LINE** **XmNO_LINE** **XmSHADOW_ETCHED_IN** **XmSHADOW_ETCHED_OUT**
Access	CSG

Description

The **XmNseparatorType** resource specifies the type of line drawing to be drawn in the **Separator** widget. The values are as follows:

- **XmSINGLE_LINE** – single line.
- **XmDOUBLE_LINE** – double line.
- **XmSINGLE_DASHED_LINE** – single dashed line.
- **XmDOUBLE_DASHED_LINE** – double dashed line.
- **XmNO_LINE** – no line.
- **XmSHADOW_ETCHED_IN** – double line giving the effect of a line etched into the window. For horizontal orientation, the top line is drawn in the **XmNtopShadowColor** resource, and the bottom line is drawn in the **XmNbottomShadowColor** resource. For vertical orientation, the left line is drawn in the **XmNtopShadowColor** resource, and the right line is drawn in the **XmNbottomShadowColor** resource.
- **XmSHADOW_ETCHED_OUT** – double line giving the effect of an etched line coming out from the window. For horizontal orientation, the top line is drawn in the **XmNbottomShadowColor** resource, and the bottom line is drawn in the **XmNtopShadowColor** resource. For vertical orientation, the left line is drawn in the **XmNbottomShadowColor** resource, and the right line is drawn in the **XmNtopShadowColor** resource.

XmSeparatorGadget Resource Set

XmNmargin **XmNorientation**

XmNseparatorType

XmNmargin

Class	**XmCMargin**
Type	**short**
Default	**0**
Access	CSG

Description

The **XmNmargin** resource specifies space, on the left and right sides, between the border of the **SeparatorGadget** gadget and the line drawn for horizontal orientation. For vertical orientation, this resource specifies space, on the top and bottom, between the border of the **SeparatorGadget** gadget and the line drawn.

XmNorientation

Class	**XmCOrientation**
Type	**unsigned_char**
Default	**XmHORIZONTAL**
Return Values	**XmVERTICAL** **XmHORIZONTAL**
Access	CSG

Description

The **XmNorientation** resource specifies whether the **SeparatorGadget** gadget is displayed vertically or horizontally. This resource can have the **XmVERTICAL** or **XmHORIZONTAL** value. The **XmHORIZONTAL** value is the default orientation.

XmNseparatorType

Class	**XmCSeparatorType**
Type	**unsigned_char**
Default	**XmSHADOW_ETCHED_IN**
Return Values	**XmSINGLE_LINE** **XmDOUBLE_LINE** **XmSINGLE_DASHED_LINE** **XmDOUBLE_DASHED_LINE** **XmNO_LINE** **XmSHADOW_ETCHED_IN** **XmSHADOW_ENCHED_OUT**
Access	CSG

Description

The **XmNseparatorType** resource specifies types of line drawings to be drawn in the **SeparatorGadget** gadget. The types of line drawings available are as follows:

- **XmSINGLE_LINE** – single line.
- **XmDOUBLE_LINE** – double line.
- **XmSINGLE_DASHED_LINE** – single dashed line.
- **XmDOUBLE_DASHED_LINE** – double dashed line.
- **XmNO_LINE** – no line.
- **XmSHADOW_ETCHED_IN** – double line giving the effect of a line etched into the window. The thickness of the double line is equal to the value of the **XmNshadowThickness** resource. For horizontal orientation, the top line is drawn in the **XmNtopShadowColor** resource, and the bottom line is drawn in the **XmNbottomShadowColor** resource. For vertical orientation, the left line is drawn in the **XmNtopShadowColor** resource, and the right line is drawn in the **XmNbottomShadowColor** resource.
- **XmSHADOW_ETCHED_OUT** – double line giving the effect of an etched line coming out from the window. The thickness of the double line is equal to the value of the **XmNshadowThickness** resource. For horizontal orientation, the top line is drawn in the **XmNbottomShadowColor** resource, and the bottom line is drawn in the **XmNtopShadowColor** resource. For vertical orientation, the left line is drawn in the **XmNbottomShadowColor** resource, and the right line is drawn in the **XmNtopShadowColor** resource.

XmText Input Resource Set

XmNpendingDelete **XmNselectThreshold**

XmNselectionArray

XmNpendingDelete

Class	**XmCPendingDelete**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNpendingDelete** resource indicates that pending delete mode is on when the Boolean value is **True**. The pending deletion is defined as deletion of the selected text when an insertion is made.

XmNselectionArray

Class	**XmCSelectionArray**
Type	**Pointer**
Default	**sarray**
Access	CSG

Description

The **XmNselectionArray** resource defines the actions for multiple mouse clicks. Each mouse click performed within a half of a second of the previous mouse click increments the index into to this array and perform the defined action for that index. The possible actions are as follows:

- **XmSELECT_POSITIONS** resets the insert cursor position.
- **XmSELECT_WORD** selects a word.
- **XmSELECT_LINE** selects a line of text.
- **XmSELECT_ALL** selects all of the text.

XmNselectThreshold

Class	**XmCSelectThreshold**
Type	**int**
Default	**5**
Access	CSG

Description

The **XmNselectThreshold** resource specifies the number of pixels of motion that are required to select the next character when selection is performed using the click–drag mode of selection.

XmText Output Resource Set

XmNblinkRate
XmNcolumns
XmNcursorPositionVisible
XmNfontList
XmNresizeHeight
XmNresizeWidth
XmNrows
XmNwordWrap

XmNblinkRate

Class	**XmCBlinkRate**
Type	**int**
Default	**500**
Access	CSG

Description

The **XmNblinkRate** resource specifies the blink rate of the text cursor in milliseconds. The time indicated in the blink rate relates to the length of time the cursor is visible and the time the cursor is invisible (in other words, the time it will take to blink the insertion cursor on and off is be two times the blink rate). The cursor does not blink when the blink rate is set to a value of zero.

XmNcolumns

Class	**XmCColumns**
Type	**short**
Default	**20**
Access	CSG

Description

The **XmNcolumns** resource specifies the initial width of the text window measured in character spaces.

XmNcursorPositionVisible

Class	**XmCCursorPositionVisible**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNcursorPositionVisible** resource indicates that the insert cursor position is marked by a blinking text cursor when the Boolean value is **True**.

XmNfontList

Class	**XmCFontList**
Type	**XmFontLIst**
Default	**"fixed"**
Access	CSG

Description

The **XmNfontList** resource specifies the font list to be used for the **Text** widget.

XmNresizeHeight

Class	**XmCResizeHeight**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNresizeHeight** resource indicates that the **Text** widget attempts to resize its height to accommodate all the text contained in the widget when the Boolean value is **True**. If set to the **True** value, the text is always displayed starting from the first position in the source, even if instructed otherwise. This resource is ignored when using a **ScrolledText** widget and when the **XmNscrollVertical** resource is the **True** value.

XmNresizeWidth

Class	**XmCResizeWidth**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNresizeWidth** resource indicates that the **Text** widget attempts to resize its width to accommodate all the text contained in the widget when the Boolean value is **True**. This resource is ignored if the **XmNwordWrap** resource is the **True** value.

XmNrows

Class	**XmCRows**
Type	**short**
Default	**1**
Access	CSG

Description

The **XmNrows** resource specifies the initial height of the text window measured in character heights. This resource is ignored if the **XmNeditMode** text widget resource is the **XmSINGLE_LINE_EDIT** value.

XmNwordWrap

Class	**XmCWordWrap**

Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNwordWrap** resource indicates that lines are to be broken at word breaks (in other words, the text does not go off the right edge of the window) when the Boolean value is **True**. Words are defined as a sequence of characters separated by white space. White space is defined as a space, tab, or new–line. This resource is ignored if the **XmNeditMode** text widget resource is the **XmSINGLE_LINE_EDIT** value.

XmText Resource Set

XmNactivateCallback
XmNautoShowCursorPosition
XmNcursorPosition
XmNeditable
XmNeditMode
XmNfocusCallback
XmNlosingFocusCallback
XmNmarginHeight
XmNmarginWidth
XmNmaxLength
XmNmodifyVerifyCallback
XmNmotionVerifyCallback
XmNtopPosition
XmNvalue
XmNvalueChangedCallback

XmNactivateCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNactivateCallback** resource specifies a callback subroutine that is called when the widget becomes active. The initial state of the **Text** widget is inactive, shown visibly by a stippled insert cursor. The callback reason is the **XmCR_ACTIVATE** value. The structure returned by this callback subroutine is **XmAnyCallbackStruct**.

XmNautoShowCursorPosition

Class	**XmCAutoShowCursorPosition**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNautoShowCursorPosition** resource ensures that the visible text contains the insert cursor when set to the **True** value. If the insert cursor changes, the contents of the **Text** widget may scroll in order to bring the insertion point into the window.

XmNcursorPosition

Class	**XmCCursorPosition**
Type	**XmTextPosition**
Default	**0**
Access	CSG

Description

The **XmNcursorPosition** resource indicates the position in the text where the current insert cursor is to be located. The position is determined by the number of characters from the beginning of the text.

XmNeditable

Class	**XmCEditable**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNeditable** resource indicates that the user can edit the text string in the **Text** widget when the Boolean value is **True**. A **False** value prohibits the user from editing the text.

XmNeditMode

Class	**XmCEditMode**
Type	**int**
Default	**XmSINGLE_LINE_EDIT**
Access	CSG

Description

The **XmNeditMode** resource specifies a set of keyboard bindings used in the **Text** widget. The default keyboard bindings (such as the **XmSINGLE_LINE_EDIT** value) provides the set of key bindings to be used in editing the multiline text in the **Text** widget. The multi–line bindings (such as the **XmMULTI_LINE_EDIT** value) provide the set of key bindings to be used in editing single–line text in the **Text** widget.

XmNfocusCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNfocusCallback** resource specifies a callback subroutine that is called before the **Text** widget has accepted input focus. The callback reason is the **XmCR_FOCUS** value. The structure returned by this callback subroutine is **XmAnyCallbackStruct**.

XmNlosingFocusCallback

Class	**XmCCallback**
Type	**XtCallbackLIst**
Default	**NULL**
Access	C

Description

The **XmNlosingFocusCallback** resource specifies a callback subroutine that is called before the **Text** widget loses input focus. The callback reason is the **XmCR_LOSING_FOCUS** value. The structure returned by this callback subroutine is **XmTextVerifyCallbackStruct**.

XmNmarginHeight

Class	**XmCMarginHeight**
Type	**short**
Default	**5**
Access	CSG

Description

The **XmNmarginHeight** resource specifies the distance between the top edge of the widget window and the text and between the bottom edge of the widget window and the text. This resource is forced to a **True** value when the **Text** widget is placed in a **ScrolledWindow** with the **XmNscrollingPolicy** resource set to the **XmAUTOMATIC** value.

XmNmarginWidth

Class	**XmCMarginWidth**
Type	**short**
Default	**5**
Access	CSG

Description

The **XmNmarginWidth** resource specifies the distance between the left edge of the widget window and the text and between the right edge of the widget window and the text. This resource is forced to a **True** value when the **Text** widget is placed in a **ScrolledWindow** with the **XmNscrollingPolicy** resource set to the **XmAUTOMATIC** value.

XmNmaxLength

Class	**XmCMaxLength**
Type	**int**
Default	**MAXINT**
Access	CSG

Description

The **XmNmaxLength** resource specifies the maximum length of the text string that can be entered into the **Text** widget from the keyboard. Strings that are entered using the **XmNvalue** resource or the **XmTextString** subroutine ignore this resource.

XmNmodifyVerifyCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNmodifyVerifyCallback** resource specifies a callback subroutine that is called before text is deleted from or inserted into the **Text** widget. The callback reason is the

XmCR_MODIFYING_TEXT_VALUE value. The structure returned by this callback subroutine is **XmTextVerifyCallbackStruct**.

XmNmotionVerifyCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNmotionVerifyCallback** resource calls a callback list before the insert cursor is moved to a new position. The structure returned by this callback is the **XmTextVerifyCallbackStruct** value. The callback reason is the **XmCR_MOVING_INSERT_CURSOR** value.

XmNtopCharacter

Class	**XmCTextPosition**
Type	**XmTextPosition**
Default	**0**
Access	CSG

Description

The **XmNtopCharacter** resource shows the relative position of text currently located at the top of the window. Position is determined by the number of characters from the beginning of the text.

XmNvalue

Class	**XmCValue**
Type	**String**
Default	""
Access	CSG

Description

The **XmNvalue** resource displays the string value. The **XtGetValues** subroutine returns the value of the internal buffer and the **XtSetValues** subroutine copies the string values into the internal buffer.

XmNvalueChangedCallback

Class	**XmCCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNvalueChangedCallback** resource specifies a callback subroutine that is called after text is deleted from or inserted into the **Text** widget. The callback reason is the **XmCR_VALUE_CHANGED** value. The structure returned by this callback subroutine is **XmAnyCallbackStruct**.

XmTextScrolled Text Resource Set

XmNscrollHorizontal
XmNscrollLeftSide
XmNscrollTopSide
XmNscrollVertical

XmNscrollHorizontal

Class **XmCScroll**

Type **Boolean**

Default **True**

Access CG

Description

The **XmNscrollHorizontal** resource adds a scrollbar that allows the user to scroll horizontally through the text when the Boolean value is **True**. This resource is ignored if the **XmNeditMode** resource of the **Text** widget is the **XmSINGLE_LINE_EDIT** value. This resource is forced to a **False** value when the **Text** widget is placed in a **ScrolledWindow** with the **XmNscrollingPolicy** resource set to the **XmAUTOMATIC** value.

XmNscrollLeftSide

Class **XmCScrollSide**

Type **Boolean**

Default **False**

Access CG

Description

The **XmNscrollLeftSide** resource indicates that the vertical scrollbar should be placed on the left side of the scrolled text window when the Boolean value is **True**. This resource is ignored if the **XmNscrollVertical** resource is the **False** value or the **XmNeditMode** text widget resource is the **XmSINGLE_LINE_EDIT** value.

XmNscrollTopSide

Class **XmCScrollSide**

Type **Boolean**

Default **False**

Access CG

Description

The **XmNscrollTopSide** resource indicates that the horizontal scrollbar should be placed on the top side of the scrolled text window when the Boolean value is **True**. This resource is ignored if the **XmNscrollHorizontal** resource is the **False** value.

XmNscrollVertical

Class **XmCScroll**

Type **Boolean**

Default	**True**
Access	CG

Description

The **XmNscrollVertical** adds a scrollbar that allows the user to scroll vertically through text when the Boolean value is **True**. This resource is ignored if the **XmNeditMode** text widget resource is the **XmSINGLE_LINE_EDIT** value. This resource is forced to a **False** value when the **Text** widget is placed in a **ScrolledWindow** with the **XmNscrollingPolicy** resource set to the **XmAUTOMATIC** value.

XmToggleButton Resource Set

XmNarmCallback
XmNdisarmCallback
XmNfillOnSelect
XmNindicatorOn
XmNindicatorType
XmNselectColor
XmNselectInsensitivePixmap
XmNselectPixmap
XmNset
XmNspacing
XmNvalueChangedCallback
XmNvisibleWhenOff

XmNarmCallback

Class	**XmCArmCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNarmCallback** resource specifies a callback subroutine called when the **ToggleButton** widget is armed. The **ToggleButton** widget is armed when the user presses while the pointer is inside the **ToggleButton** widget. The callback reason is the **XmCR_ARM** value.

XmNdisarmCallback

Class	**XmCDisarmCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNdisarmCallback** resource specifies a callback subroutine called when the **ToggleButton** widget is disarmed. When the user activates the **ToggleButton** widget (presses and releases while the pointer is inside the widget), the **ToggleButton** widget is disarmed. The **ToggleButton** widget is also disarmed when the user presses mouse button one when the pointer is inside the **ToggleButton** widget, moves out of the widget, and releases the when the pointer is outside the widget. The callback reason is the **XmCR_DISARM** value.

XmNfillOnSelect

Class	**XmCFillOnSelect**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNfillOnSelect** resource fills the indicator with the foreground color when set to the **True** value, otherwise it switches only the top and bottom shadow colors.

XmNindicatorOn

Class	**XmCIndicatorOn**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNindicatorOn** resource specifies a Boolean value that when the **True** value signifies that the indicator is present in the **ToggleButton** widget, and vice versa.

XmNindicatorType

Class	**XmCIndicatorType**
Type	**unsigned char**
Default	**XmN_OF_MANY**
Access	CSG

Description

The **XmNindicatorType** resource specifies whether the indicator is a **1–of** or **N–of** indicator. For the **1–of** indicator the value is the **XmONE_OF_MANY** value, and for the **N–of** indicator, the value is the **XmN_OF_MANY** value. The **N–of–many** indicator is square–shaped and the **1–of–many** indicator is diamond–shaped. This resource specifies the visuals, but does not enforce the behavior.

XmNselectColor

Class	**XmCSelectColor**
Type	**Pixel**
Default	**dynamic**
Access	CSG

Description

The **XmNselectColor** resource allows the application to specify what color fills the center of the square or diamond indicator when it is set. If this color is the same as either the top or bottom shadow color of the indicator, a one–pixel–wide margin is left between the shadows and the fill; otherwise, it is completely filled. The default of this resource for a color screen is a color between the background and bottom shadow color. For a monochrome display, the default is set to the foreground color.

XmNselectInsensitivePixmap

Class	**XmCSelectInsensitivePixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**

Access CSG

Description

The **XmNselectInsensitivePixmap** resource specifies the pixmap used as the button face when the **ToggleButton** widget is selected. The button is insensitive if the **XmLabel** resource **XmNlabelType** is the **XmPIXMAP** value. If the **ToggleButton** widget is not selected and the button is insensitive, the pixmap in the **XmNlabelInsensitivePixmap** resource is used as the button face.

XmNselectPixmap

Class **XmCSelectPixmap**

Type **Pixmap**

Default **XmUNSPECIFIED_PIXMAP**

Access CSG

Description

The **XmNselectPixmap** resource specifies the pixmap to be used as the button face if the **XmNlabelType** resource is the **XmPIXMAP** value and the **ToggleButton** widget is in the on position.

XmNset

Class **XmCSet**

Type **Boolean**

Default **False**

Access CSG

Description

The **XmNset** resource displays the button in its selected state if set to the **True** value. This is useful for showing some conditions as active when a set of buttons first appears.

XmNspacing

Class **XmCspacing**

Type **short**

Default **4**

Access CSG

Description

The **XmNspacing** resource specifies the amount of spacing between the toggle indicator and the toggle label (text or pixmap).

XmNvalueChangedCallback

Class **XmCValueChangedCallback**

Type **XtCallbackList**

Default **NULL**

Access C

Description

The **XmNvalueChangedCallback** resource specifies a callback subroutine that is called when the **ToggleButton** widget value is changed. The value is changed when the user presses and releases while the pointer is inside the **ToggleButton** widget. This action also causes the **XmToggleButton** widget to be disarmed. The callback reason is the **XmCR_VALUE_CHANGED** value.

XmNvisibleWhenOff

Class	**XmCVisibleWhenOff**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNvisibleWhenOff** resource specifies that the toggle indicator is visible in the off position when the Boolean value is **True**.

XmToggleButtonGadget Resource Set

XmNarmCallback	XmNdisarmCallback
XmNfillOnSelect	XmNindicatorOn
XmNindicatorType	XmNselectColor
XmNselectInsensitivePixmap	XmNselectPixmap
XmNset	XmNspacing
XmNvalueChangedCallback	XmNvisibleWhenOff

XmNarmCallback

Class	XmCArmCallback
Type	XtCallbackList
Default	NULL
Access	C

Description

The **XmNarmCallback** resource specifies a list of callbacks that is called when the **ToggleButtonGadget** gadget is armed. To arm this gadget, press the active mouse button while the pointer is inside the **ToggleButtonGadget** gadget. The callback reason is the **XmCR_ARM** value.

XmNdisarmCallback

Class	XmCDisarmCallback
Type	XtCallbackList
Default	NULL
Access	C

Description

The **XmNdisarmCallback** resource specifies a list of callbacks that is called when the **ToggleButtonGadget** gadget is disarmed. To disarm this gadget, press and release the active mouse button while the pointer is inside the **ToggleButtonGadget** gadget. The gadget is also disarmed when the user moves out of the gadget and releases the mouse button when the pointer is outside the gadget. The callback reason is the **XmCR_DISARM** value.

XmNfillOnSelect

Class	XmCFillOnSelect
Type	Boolean
Default	True
Access	CSG

Description

The **XmNfillOnSelect** resource fills the indicator with the color specified in the **XmNselectColor** resource and switches the top and bottom shadow when set to the **True** value. If set to the **False** value, it switches only the top and bottom shadow colors.

XmNindicatorOn

Class	**XmCIndicatorOn**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNindicatorOn** resource specifies that a toggle indicator is drawn to the left of the toggle text or pixmap when set to the **True** value. When set to the **False** value, no space is allocated for the indicator, and it is not displayed. If the **XmNindicatorOn** resource is the **True** value, the indicator shadows are switched when the button is selected or unselected, but any shadows around the entire gadget are not switched. However, if the **XmNindicatorOn** resource is the **False** value, any shadows around the entire gadget are switched when the toggle is selected or unselected.

XmNindicatorType

Class	**XmCIndicatorType**
Type	**unsigned char**
Default	**XmN_OF_MANY**
Access	CSG

Description

The **XmNindicatorType** resource specifies if the indicator is a **1–of** or **N–of** indicator. For the **1–of** indicator, the value is **XmONE_OF_MANY**. For the **N–of** indicator, the value is **XmN_OF_MANY**. The **N–of–many** indicator is square–shaped. The **1–of–many** indicator is diamond–shaped. This resource only specifies the visuals and does not enforce the behavior. When the **ToggleButtonGadget** gadget is in a **RadioBox** widget, the parent forces this resource to the **XmONE_OF_MANY** value.

XmNselectColor

Class	**XmCSelectColor**
Type	**Pixel**
Default	**dynamic**
Access	CSG

Description

The **NselectColor** resource allows the application to specify what color fills the center of the square or diamond indicator when it is set. If this color is the same as either the top or bottom shadow color of the indicator, a one–pixel–wide margin is left between the shadows and the fill; otherwise, it is completely filled . The default of this resource for a color display is a color between the background and the bottom shadow color. For a monochrome display device, the default is set to the background color.

XmNselectInsensitivePixmap

Class	**XmCSelectInsensitivePixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNselectInsensitivePixmap** resource specifies a pixmap used as the button face when the **ToggleButtonGadget** gadget is selected, and the button is insensitive if the **XmLabelGadget** widget **XmNlabelType** resource is the **XmPIXMAP** value. If the **ToggleButtonGadget** gadget is unselected and the button is insensitive, the pixmap in the **XmNlabelInsensitivePixmap** resource is used as the button face.

XmNselectPixmap

Class	**XmCSelectPixmap**
Type	**Pixmap**
Default	**XmUNSPECIFIED_PIXMAP**
Access	CSG

Description

The **XmNselectPixmap** resource specifies the pixmap to be used as the button face if the **XmNlabelType** resource is the **XmPIXMAP** value and the **ToggleButtonGadget** gadget is selected. When the **ToggleButtonGadget** gadget is unselected, the pixmap specified in the **XmLabel** widget **XmNlabelPixmap** resource is used.

XmNset

Class	**XmCSet**
Type	**Boolean**
Default	**False**
Access	CSG

Description

The **XmNset** resource displays the button in its selected state if set to the **True** value. This shows some conditions as active when a set of buttons first appear.

XmNspacing

Class	**XmCSpacing**
Type	**short**
Default	**4**
Access	CSG

Description

The **XmNspacing** resource specifies the amount of spacing between the toggle indicator and the toggle label (text or pixmap).

XmNvalueChangedCallback

Class	**XmCValueChangedCallback**
Type	**XtCallbackList**
Default	**NULL**
Access	C

Description

The **XmNvalueChangedCallback** resource specifies a list of callbacks that is called when the **ToggleButtonGadget** value is changed. To change the value, press and release the parent–determined active mouse button while the pointer is inside the **ToggleButtonGadget** value. This action also causes the gadget to be disarmed. The callback reason is the **XmCR_VALUE_CHANGED** value.

XmNvisibleWhenOff

Class	**XmCVisibleWhenOff**
Type	**Boolean**
Default	**True**
Access	CSG

Description

The **XmNvisibleWhenOff** resource indicates that the toggle indicator is visible in the unselected state when the Boolean value is **True**. When the **ToggleButtonGadget** gadget is in a menu, the **RowColumn** parent widget forces this resource to the **False** value. When the **ToggleButtonGadget** gadget is in an **RadioBox** widget, the parent widget forces this resource to the **True** value.

AIXwindows Desktop Resource Sets

AIXwindows Desktop Resource Sets

The following AIXwindows resources can be used to determine the appearance and behavior of the AIXwindows Desktop. These resources can be found and edited within the **.Xdefaults** file in your home directory or in the **app–defaults/Xdesktop** file.

Text

Xdesktop*font
Xdesktop*text.margin

Icon layout

Xdesktop*directory.aisleWidth
Xdesktop*iconGrid.spacing
Xdesktop*iconsAsBitmaps
Xdesktop*rename.width
Xdesktop*fileIcon.pixmap
Xdesktop*fileIcon.background
Xdesktop*fileIcon.foreground
Xdesktop*newicon.pixmap
Xdesktop*newicon.background
Xdesktop*newicon.foreground

File Defaults

Xdesktop*initialEnvironmentRuleFile
Xdesktop*isWindowManager
Xdesktop*name
Xdesktop*userRuleFile
Xdesktop*systemRuleFile
Xdesktop*directoryRuleFIle
Xdesktop*windowlessIcons
Xdesktop*pictureDirectory

Triggers

Xdesktop*triggers.mapping
Xdesktop*triggers.maxMotion
Xdesktop*triggers.maxUpTime

Cursor Shapes

Xdesktop.busy.data
Xdesktop.busy.mask
Xdesktop.drag.data
Xdesktop.drag.mask
Xdesktop.idle.data
Xdesktop.idle.mask
Xdesktop.menu.data
Xdesktop.menu.mask
Xdesktop.multiDrag.data
Xdesktop.multiDrag.mask
Xdesktop.none.data
Xdesktop.none.mask
Xdesktop*textIn.data
Xdesktop*textIn.mask

Desktop Appearance

Xdesktop*desktop.height
Xdesktop*desktop.width
Xdesktop*desktop.x
Xdesktop*desktop.y
Xdesktop*title
Xdesktop*iconName
Xdesktop*directoryGroups
Xdesktop*desktopIcon.pixmap
Xdesktop*desktopIcon.background
Xdesktop*desktopIcon.foreground
Xdesktop*desktop.geometry
Xdesktop*directory.enableStatusBar

Menus

Xdesktop*desktop.menuMapping
Xdesktop*directory.menuMapping

Message Windows

Xdesktop*normal.foreground
Xdesktop*normal.background
Xdesktop*message.fatal.pixmap
Xdesktop*message.fatal.background
Xdesktop*message.fatal.foreground
Xdesktop*message.alert.pixmap
Xdesktop*message.alert.background
Xdesktop*message.alert.foreground
Xdesktop*message.fyi.pixmap
Xdesktop*message.fyi.background
Xdesktop*message.fyi.foreground
Xdesktop*message.greeting.pixmap
Xdesktop*message.greeting.background
Xdesktop*message.greeting.foreground
Xdesktop*textButton.pixmap
Xdesktop*textButton.background
Xdesktop*textButton.foreground
Xdesktop*message.timeout

Launching Programs

Xdesktop*processLaunch.data
Xdesktop*processLaunch.mask
Xdesktop*process.launch.show
Xdesktop*process.launch.time
Xdesktop*process.showBorder
Xdesktop*process.titleBorder

Related Information

How to configure the AIXwindows Desktop through resources.

AIXwindows Desktop Text Appearance Resource Set

Xdesktop*font

Class **Font**

Type **string**

Default **"fixed"**

Description

Specifies the name of the font that the desktop uses for text. The **/usr/lpp/X11/defaults/fonts** file contains a list of valid fonts names, along with the information that AIXwindows uses in displaying them.

Xdesktop*textMargin

Class **TextMargin**

Type **Integer**

Default **2**

Description

Specifies the amount of space, in pixels, that appears around all text displayed by the desktop.

AIXwindows Desktop Icon Layout Resource Set

Xdesktop*directory.aisleWidth

Class **Directory.AisleWidth**

Type **Integer**

Default **8**

Description

Specifies the minimum number of pixels to be left between each icon in a directory window when it is first opened and whenever it is tidied.

Xdesktop*iconGrid.spacing

Class **IconGrid.Spacing**

Type **Integer**

Default **100**

Description

Specifies how far apart icons are when the desktop is tidied. The value is indicated in pixels from the center of each icon.

Xdesktop*iconsAsBitmaps

Class **IconsAsBitmaps**

Type **Boolean**

Default **TRUE**

Description

Specifies whether icons are saved with only two colors. This uses less memory, but takes longer to display the icon.

Xdesktop*rename.width

Class **Rename.Width**

Type **Integer**

Default **14**

Description

Specifies the width, in characters, of the title of an icon being renamed.

Xdesktop*fileIcon.pixmap

Class **Icon.Pixmap**

Type **pixmap**

Default **#6c889c**

Description

The pixmap used for files for which no picture is specified in the rules.

Xdesktop*fileIcon.background

Class **Icon.Background**

Type **color**

Default **#e0414**

Description

The background color used for files for which no background color is specified in the rules. The default is light blue grey.

Xdesktop*fileIcon.foreground

Class **Icon.Foreground**

Type **color**

Default **black**

Description

The foreground color used for files for which no picture is specified in the rules. The default is dark blue grey.

Xdesktop*newicon.pixmap

Class **Icon.pixmap**

Type **pixmap**

Description

The pixmap used for icons for new files or directories.

Xdesktop*newicon.background

Class **Icon.background**

Type **color**

Description

The background color used for icons for new files or directories.

Xdesktop*newicon.foreground

Class **Icon.foreground**

Type **color**

Description

The foreground color used for icons for new files or directories.

AIXwindows Desktop Files Resource Set

Xdesktop*initialEnvironmentRuleFile

Class **InitialEnvironmentRuleFile**

Type *FileName*

Default **xdtinitial.xde**

Description

Specifies the name of the initial environment rule file.

Xdesktop*isWindowManager

Class **IsWindowManager**

Type **Boolean**

Default **FALSE**

Description

Determines whether the desktop runs as a window manager or as an ordinary program.

Xdesktop*name

Class **Name**

Type **string**

Default **"xdt"**

Description

Specifies the name used for reading resources. This is always read with the first element "xdt", but all other resources are looked up with this as the first element.

Xdesktop*userRuleFile

Class **UserRuleFile**

Type *FileName*

Default **.xdtuserinfo**

Description

Specifies the name of the user's rule file. If the user's home directory does not contain a file of this name, the Desktop looks for a file named **.xdtuserinfo**.

Xdesktop*systemRuleFile

Class **SystemRuleFile**

Type *FileName*

Default **Set upon installation.**

Description

Specifies the name of the system rule file. It should only be changed when it is impossible to install the system rule file in its default location, and should not be set in users' **.Xdefaults** files. This value is set upon installation to **/usr/lpp/xdt/lpp.sysinfo/***prime* where *prime* represents the first language for which National Language Support had been installed for the Desktop.

Xdesktop*directoryRuleFile

Class **DirectoryRuleFile**

Type *FileName*

Default **.Set upon installation.**

Description

Specifies the name of the directory rule files. If a directory does not contain a file of this name, the Desktop looks for a file named **xdtdir.$LANG** where **$LANG** represent the value of the user's **LANG** environment variable.

Xdesktop*windowlessIcons

Class **WindowlessIcons**

Type **Boolean**

Default **FALSE**

Description

Specifies whether icons are displayed by painting on the background (TRUE) or by using windows (FALSE).

Xdesktop*pictureDirectory

Class **PictureDirectory**

Type **String**

Default none

Description

The absolute path name of the directory containing the bit mapped pictures that the desktop uses.

AIXwindows Desktop Triggers Resource Set

Xdesktop*triggers.mapping

Class **Triggers.Mapping**

Type **String**

Default

```
1=!s      ; 2=+s     ; 3=-s      ;4=~s     ; \
1,1=s1    ; 2,2=s3   ; 3,3=s2    ; 4,4=s4    ;5,5=s5   ; \
1=d1      ; 2=d3     ; 3=d2      ; 4=d4     ; 5=d5
```

Description

Converts triggers to trigger IDs.

Xdesktop*triggers.maxMotion

Class **Triggers.MaxMotion**

Type **Integer**

Default **3**

Description

Specifies the amount of motion (in pixels) allowed within a mouse click. If the mouse moves by more than this amount in any direction, its motion is considered a drag.

Xdesktop*triggers.maxUpTime

Class **Triggers.Time**

Type **Integer**

Default **700 ms**

Description

Determines the maximum time, in milliseconds, between clicks of a double click. If more time than this passes, the clicks are treated as two single clicks.

AIXwindows Desktop Cursors Resource Set

Xdesktop.busy.data

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the data pixmap for the icon that appears when the system is busy.

Xdesktop.busy.mask

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the mask pixmap for the icon that appears when the system is busy.

Xdesktop.drag.data

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the data pixmap for the icon that appears when the user is dragging an icon.

Xdesktop.drag.mask

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the mask pixmap for the icon that appears when the user is dragging an icon.

Xdesktop.multiDrag.data

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the data pixmap for the icon that appears when the user is dragging multiple icons.

Xdesktop.multiDrag.mask

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the mask pixmap for the icon that appears when the user is dragging multiple icons.

Xdesktop.idle.data

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the data pixmap for the icon that appears when the desktop is not busy.

Xdesktop.idle.mask

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the mask pixmap for the icon that appears when the desktop is not busy.

Xdesktop.menu.data

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the data pixmap for the icon that points to menu selections.

Xdesktop.menu.mask

Class **Cursor.Bitmap**

Type **Pixmap**

Description

The name of the file containing the mask pixmap for the icon that points to menu selections.

Xdesktop*none.data

Class **Cursor.Bitmap**

Type **Pixmap**

Description

Specifies the data pixmap for the cursor when a window is being resized or moved.

Xdesktop*none.mask

Class **Cursor.Bitmap**

Type **Pixmap**

Description

Specifies the mask pixmap for the cursor when a window is being resized or moved.

Xdesktop*textIn.data

Class **Cursor.Bitmap**

Type **Pixmap**

Description

Specifies the data pixmap for the cursor when the desktop is waiting for text input.

Xdesktop*textIn.mask

Class **Cursor.Bitmap**

Type **Pixmap**

Description

Specifies the mask pixmap for the cursor when the desktop is waiting for text input.

AIXwindows Desktop Appearance Resource Set

Xdesktop*desktop.height

Class **Desktop.Height**

Type **Integer**

Default 3/4 of screen height

Description

Specifies the height of the desktop when it is first displayed.

Xdesktop*desktop.width

Class **Desktop.Width**

Type **Integer**

Default 3/4 of screen width

Description

Specifies the width of the desktop when it is first displayed.

Xdesktop*desktop.x

Class **Desktop.X**

Type **Integer**

Default 1/8 of screen width

Description

Specifies the x coordinate, in pixels, of the top left corner of the desktop when it is first displayed.

Xdesktop*desktop.y

Class **Desktop.Y**

Type **Integer**

Default 1/8 of screen height

Description

Specifies the y coordinate, in pixels, of the top left corner of the desktop when it is first displayed.

Xdesktop*title

Class **Title**

Type **string**

Default Taken from NLS

Description

The name which the desktop provides to the Window Manager.

Xdesktop*iconName

Class **IconName**

Type **string**

Default Taken from NLS

Description

The name which the desktop provides to the Window Manager for its icon.

Xdesktop*directoryGroups

Class **DirectoryGroups**

Type **Boolean**

Default **FALSE**

Description

Specifies whether directory windows are transient (FALSE) or belong to a window group (TRUE).

Xdesktop*desktopIcon.pixmap

Class **Icon.Pixmap**

Type **pixmap**

Description

Specifies the icon which identifies the desktop within the window manager icon box.

Xdesktop*desktopIcon.background

Class **Icon.Background**

Type **color**

Description

Specifies the background color for the icon which identifies the desktop within the window manager icon box. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*desktopIcon.foreground

Class **Icon.Foreground**

Type **color**

Description

Specifies the foreground color for the icon which identifies the desktop within the window manager icon box. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*desktop.geometry

Class **Desktop.Geometry**

Type **string**

Description

Specifies the dimensions and placement of the desktop window.

Xdesktop*directory.enableStatusBar

Class **directory.enableStatusBar**

Type **Boolean**

Default **TRUE**

Description

Specifies whether directories have status bars.

Xdesktop*IconButton.background

Class **IconButton.Background**

Type **color**

Default **white**

Description

Specifies the background color of icon buttons. The default is extra dark blue grey.

Xdesktop*IconButton.foreground

Class **IconButton.Foreground**

Type **color**

Default **black**

Description

Specifies the foreground color of icon buttons.

Xdesktop*directory.XmScrollBar.foreground

Class **Directory.XmScrollBar.Foreground**

Type **color**

Default **black**

Description

Specifies the foreground color of directory window scroll bars.

Xdesktop*XmMessageBox.Background

Class Directory.XmMessageBox.Background

Type color

Default white

Description

Specifies the background color of desktop message boxes. The default is medium blue grey.

Xdesktop*XmFrame.Background

Class Directory.XmFrame.Background

Type color

Default white

Description

Specifies the background color of desktop message boxes. The default is medium blue grey.

Xdesktop*desktop,base.background

Class desktop,base.background

Type color

Default white

Description

The background color of the main desktop window. The default is light blue grey.

Xdesktop*directory.background

Class desktop,base.background

Type color

Default white

Description

The background color of the main desktop window. The default is light blue grey.

AIXwindows Desktop Menus Resource Set

Xdesktop*desktop.menuMapping

Class **Desktop.MenuMapping**

Type **string**

Default **"3=Mainxdt"**

Description

Specifies the mapping of trigger actions to menu options on the desktop.

The values of this resource are in the following format:

trigger[:*key*]=*menuname*

where

trigger Specifies the number of a mouse button

key Specifies a single letter indicating a key that is pressed at the same time as the mouse button. Valid values include the following:

c Ctrl

s Shift

a Alt

menuname Specifies a menu to be opened when the trigger and key are pressed. The desktop searches the rules files for a menu of that name.

For example, the default value, "3=Mainxdt" specifies that the **Mainxdt** menu appears when the user presses mouse button 3 (both mouse buttons).

This resource can include several mapping strings, separated by a semicolon.

Xdesktop*directory.menuMapping

Class **Directory.MenuMapping**

Type **string**

Default **"3=Maindir"**

Description

Specifies the mapping of trigger actions to menu options within directory windows.

The values of this resource are in the following format:

trigger[:*key*]=*menuname*

where

trigger Specifies the number of a mouse button

key Specifies a single letter indicating a key that is pressed at the same time as the mouse button. Valid values include the following:

c Ctrl

s Shift

a Alt

menuname Specifies a menu to be opened when the trigger and key are pressed. The desktop searches the rules files for a menu of that name.

For example, the default value, "3=Maindir" specifies that the Maindir menu appears when the user presses mouse button 3 (both mouse buttons).

This resource can include several mapping strings, separated by semicolons.

Xdesktop*nevermapped.MainXdtMenu.background

Class **Nevermapped.MainXdtMenu.Background**

Type **color**

Default **#d4d8e8**

Description

The background color of the main desktop menu. The default color is a medium blue grey.

Xdesktop*nevermapped.MainDirMenu.background

Class **Nevermapped.MainDirMenu.Background**

Type **color**

Default **#d4d8e8**

Description

The background color of the main directory menu. The default color is a medium blue grey.

Xdesktop*nevermapped*pop_mainsub_app_menu*background

Class Nevermapped.pop_mainsub_app_menu.Background

Type color

Default #c8ccde

Description

The background color of the application menu that pops up from the main desktop menu. The default color is a medium light blue grey.

Xdesktop*nevermapped*pop_mainsub_misc_menu*background

Class Nevermapped.pop_mainsub_misc_menu.Background

Type color

Default #c8ccde

Description

The background color of the miscellaneous menu that pops up from the main desktop menu. The default color is a medium light blue grey.

Xdesktop*nevermapped*pop_dir_file_menu*background

Class Nevermapped.pop_dir_file_menu.Background

Type color

Default #c8ccde

Description

The background color of the file menu that pops up from the main directory menu. The default color is a medium light blue grey.

Xdesktop*nevermapped*pop_dir_sort_menu*background

Class Nevermapped.pop_dir_sort_menu.Background

Type color

Default #c8ccde

Description

The background color of the sort menu that pops up from the main directory menu. The default color is a medium light blue grey.

Xdesktop*nevermapped*pop_dir_view_menu*background

Class Nevermapped.pop_dir_view_menu.Background

Type color

Default #c8ccde

Description

The background color of the view menu that pops up from the main directory menu. The default color is a medium light blue grey.

AIXwindows Desktop Message Windows Resource Set

Xdesktop*normal.foreground

Class **Normal.Foreground**

Type **Color**

Default **"black"**

Description

Specifies the foreground color for text windows. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*normal.background

Class **Normal.Background**

Type **Color**

Default **#e0e4f4**

Description

Specifies the background color for text windows. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen. The default is light blue grey.

Xdesktop*message.fatal.pixmap

Class **Message.Logo.Pixmap**

Type **pixmap**

Description

Specifies the pixmap for messages that appear when programs end unexpectedly.

Xdesktop*message.fatal.background

Class **Message.Logo.Background**

Type **color**

Default **"white"**

Description

Specifies the background color for messages that appear when programs end unexpectedly. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.fatal.foreground

Class **Message.Logo.Foreground**

Type **color**

Default **"red"**

Description

Specifies the foreground color for messages that appear when programs end unexpectedly. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.alert.pixmap

Class **Message.Logo.Pixmap**

Type **pixmap**

Description

Specifies the pixmap for warning messages.

Xdesktop*message.alert.background

Class **Message.Logo.Background**

Type **color**

Default **"white"**

Description

Specifies the background color for warning messages. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.alert.foreground

Class **Message.Logo.Foreground**

Type **color**

Default **"red"**

Description

Specifies the foreground color for warning messages. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.fyi.pixmap

Class **Message.Logo.Pixmap**

Type **pixmap**

Description

Specifies the pixmap for informative messages.

Xdesktop*message.fyi.background

Class **Message.Logo.Background**

Type **color**

Default **"white"**

Description

Specifies the background color for informative messages. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.fyi.foreground

Class **Message.Logo.Foreground**

Type **color**

Default **"blue"**

Description

Specifies the foreground color for informative messages. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.greeting.pixmap

Class **Message.Logo.Pixmap**

Type **pixmap**

Description

Specifies the pixmap for the message that appears when the Desktop begins.

Xdesktop*message.greeting.background

Class **Message.Logo.Background**

Type **color**

Default **"white"**

Description

Specifies the background color for the message that appears when the Desktop begins. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.greeting.foreground

Class **Message.Logo.Foreground**

Type **color**

Default **"black"**

Description

Specifies the foreground color for the message that appears when the Desktop begins. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*textButton.pixmap

Class **textButton.Pixmap**

Type **pixmap**

Description

Specifies the pixmap for the button which appears next to the names of text files when the Desktop displays files in name mode.

Xdesktop*textButton.background

Class textButton.Background

Type Color

Default "white"

Description

Specifies the background color for the button which appears next to the names of text files when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*textButton.foreground

Class textButton.Foreground

Type Color

Default "NavyBlue"

Description

Specifies the foreground color for the button which appears next to the names of text files when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*directoryButton.pixmap

Class directoryButton.Pixmap

Type pixmap

Description

Specifies the pixmap for the button which appears next to the names of directories when the Desktop displays files in name mode.

Xdesktop*directoryButton.background

Class directoryButton.Background

Type Color

Default "white"

Description

Specifies the background color for the button which appears next to the names of directories when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*directoryButton.foreground

Class directoryButton.Foreground

Type Color

Default "NavyBlue"

Description

Specifies the foreground color for the button which appears next to the names of directories when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*executableButton.pixmap

Class executableButton.Pixmap

Type pixmap

Description

Specifies the pixmap for the button which appears next to the names of executable files when the Desktop displays files in name mode.

Xdesktop*executableButton.background

Class executableButton.Background

Type Color

Default "white"

Description

Specifies the background color for the button which appears next to the names of executable files when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*executableButton.foreground

Class executableButton.Foreground

Type Color

Default "NavyBlue"

Description

Specifies the foreground color for the button which appears next to the names of executable files when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*otherButton.pixmap

Class otherButton.Pixmap

Type pixmap

Description

Specifies the pixmap for the button which appears next to the names of other files when the Desktop displays files in name mode.

Xdesktop*otherButton.background

Class otherButton.Background

Type Color

Default "white"

Description

Specifies the background color for the button which appears next to the names of other files when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*otherButton.foreground

Class otherButton.Foreground

Type Color

Default "NavyBlue"

Description

Specifies the foreground color for the button which appears next to the names of other files when the Desktop displays files in name mode. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

Xdesktop*message.timeout

Class Message.Timeout

Type integer

Default 0

Description

Specifies the length of time (in seconds) for which the greeting message is displayed. If it is set to the **0** value, the window must be explicitly closed.

AIXwindows Desktop Process Launch Resource Set

Xdesktop*process.launch.show

Class Process.Launch.Show

Type Boolean

Default TRUE

Description

Determines whether the cursor changes to the launch cursor when a process is run.

Xdesktop*processLaunch.data

Class Cursor.Bitmap

Type pixmap

Default xdt_c_small/prog_d.px

Description

Indicates the data file for the process launch cursor.

Xdesktop*processLaunch.mask

Class Cursor.Bitmap

Type pixmap

Default xdt_c_small/prog_m.px

Description

Indicates the mask file for the process launch cursor.

Xdesktop*process.launch.time

Class Process.Launch.Time

Type Integer

Description

indicates the length of time (in seconds) for which the launch cursor is shown.

Xdesktop*process.showBorder

Class Process.ShowBorder

Type Boolean

Default FALSE

Description

Indicates whether process window borders are visible within the process window frame.

Xdesktop*process.titleBorder

Class	**Process.titleBorder**
Type	**Integer**
Default	**3**

Description

Specifies the border width, in pixels, for process titles.

AIXwindows Window Management

AIXwindows Window Management Client Specific Resource Set

The following resources specify the appearance and behavior for the icons and window frames of a particular client window or class of client windows:

clientDecoration
matteBackground
clientFunctions
matteBottomShadowColor
focusAutoRaise
matteBottomShadowPixmap
iconImage
matteForeground
iconImageBackground
matteTopShadowColor
iconImageBottomShadowColor
matteTopShadowPixmap
iconImageBottomShadowPixmap
matteWidth
iconImageForeground
maximumClientSize
iconImageTopShadowColor
useClientIcon
iconImageTopShadowPixmap
windowMenu

clientDecoration

Class **ClientDecoration**

Type **string**

Default **all**

Description

Controls the amount of window frame decoration. The resource is specified as a list of decorations to specify their inclusion in the frame. Decorations preceded by a minus sign are excluded from the frame. The sign of the first item in the list determines the initial amount of decoration. If the sign of the first decoration is minus, window management assumes all decorations are present and starts subtracting from that set. If the sign of the first decoration is plus (or not specified), window management starts with no decoration and builds up a list from the resource.

Name	Description
all	Includes all decorations (default value).
border	Includes window border.
maximize	Includes maximize button (includes title bar).
minimize	Includes minimize button (includes title bar).
none	Includes no decorations.
resizeh	Includes border resize handles (includes border).
menu	Includes window menu button (includes title bar).
title	Includes title bar (includes border).

Examples:

```
Mwm*XClock.clientDecoration: -resizeh -maximize
```

This removes the resize handles and maximize button from **XClock** windows.

```
Mwm*XClock.clientDecoration: menu minimize border
```

This does the same thing as above. Note that either the **menu** or **minimize** value implies the **title** value.

clientFunctions

Class **ClientFunctions**

Type **string**

Default **all**

Description

Indicates which window management functions are applicable (or not applicable) to the client window. The value for the resource is a list of functions. If the first function in the list is preceded by a minus sign, window management starts with all functions and subtracts from that set. If the first function in the list is preceded by a plus sign, the window management starts with no functions and builds up a list. Each function in the list must be preceded by the appropriate plus or minus sign and be separated from the next function by a space.

The table below lists the functions available for this resource:

Name	Description
all	Includes all functions (default value)
none	No functions
resize	f.resize
move	f.move
minimize	f.minimize
maximize	f.maximize
close	f.kill

focusAutoRaise

Class **FocusAutoRaise**

Type **Boolean**

Default **True**

Description

When the value of this resource is set to the **True** value, clients automatically move to the top of the window stack when they get the keyboard input focus. If the value is set to the **False** value, the stacking of windows on the display is not changed when a window gets the keyboard input focus.

iconImage

Class **IconImage**

Type **string**

Default **System defined image**

Description

Specifies an icon image for a client (for example, the **Mwm*myclock*iconImage** image). The resource value is a path name for a bitmap file. The value of the client–specific **useClientIcon** resource determines whether user-supplied icon images are used instead of client supplied icon images. The default value is to display a built–in window management icon image.

iconImageBackground

Class **Background**

Type **color**

Default Specified by the **Mwm*background** or **Mwm*icon*background** resources.

Description

Specifies the background color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon background color (specified by the **Mwm*background** or **Mwm*icon*background** resources). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

iconImageBottomShadowColor

Class **Foreground**

Type **color**

Default Specified by the **Mwm*icon*bottomShadowColor** resource.

Description

Specifies the bottom shadow color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon bottom shadow color (specified by the **Mwm*icon*bottomShadowColor** resource). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

iconImageBottomShadowPixmap

Class **BottomShadowPixmap**

Type **color**

Default Specified by the **Mwm*icon*bottomShadowPixmap** resource.

Description

Specifies the bottom shadow pixmap of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon bottom shadow pixmap (specified by the **Mwm*icon*bottomShadowPixmap** resource).

iconImageForeground

Class **Foreground**

Type **color**

Default Specified by the **Mwm*foreground** or **Mwm*icon*foreground** resources.

Description

Specifies the foreground color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon foreground color (i.e.,specified by the **Mwm*foreground** or **Mwm*icon*foreground** resources). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

iconImageTopShadowColor

Class **Background**

Type **color**

Default specified by the **Mwm*icon*topShadowColor** resource

Description

Specifies the top shadow color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon top shadow color (i.e., specified by the **Mwm*icon*topShadowColor** resource). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

iconImageTopShadowPixmap

Class **TopShadowPixmap**

Type **string**

Default specified by the **Mwm*icon*topShadowPixmap** resource

Description

Specifies the top shadow pixmap of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon top shadow pixmap (specified by the **Mwm*icon*topShadowPixmap** resource).

matteBackground

Class **Background**

Type **color**

Default specified by the **Mwm*background** or **Mwm*client*background** resources

Description

Specifies the background color of the matte, when the **matteWidth** resource is positive. The default value of this resource is the client background color (specified by the **Mwm*background** or **Mwm*client*background** resources). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

matteBottomShadowColor

Class **Foreground**

Type **color**

Default specified by the **Mwm*bottomShadowColor** or **Mwm*client*bottomShadowColor** resources

Description

Specifies the bottom shadow color of the matte, when the **matteWidth** resource is positive. The default value of this resource is the client bottom shadow color (specified by the **Mwm*bottomShadowColor** or **Mwm*client*bottomShadowColor** resources). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

matteBottomShadowPixmap

Class **BottomShadowPixmap**

Type **string**

Default specified by the **Mwm*bottomShadowPixmap** or **Mwm*client*bottomShadowPixmap** resource

Description

Specifies the bottom shadow pixmap of the matte, when the **matteWidth** resource is positive. The default value of this resource is the client bottom shadow pixmap (i.e., specified by the **Mwm*bottomShadowPixmap** or **Mwm*client*bottomShadowPixmap** resource).

matteForeground

Class **Foreground**

Type **color**

Default specified by the **Mwm*foreground** or **Mwm*client*foreground** resources

Description

Specifies the foreground color of the matte, when the **matteWidth** resource is positive. The default value of this resource is the client foreground color (specified by the **Mwm*foreground** or **Mwm*client*foreground** resources). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

matteTopShadowColor

Class **Background**

Type **color**

Default specified by the **Mwm*topShadowColor** or **Mwm*client*topShadowColor** resources

Description

Specifies the top shadow color of the matte, when the **matteWidth** resource is positive. The default value of this resource is the client top shadow color (specified by the **Mwm*topShadowColor** or **Mwm*client*topShadowColor** resources). The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

matteTopShadowPixmap

Class **TopShadowPixmap**

Type **string**

Default specified by the **Mwm*topShadowPixmap** or **Mwm*client*topShadowPixmap** resources

Description

Specifies the top shadow pixmap of the matte, when the **matteWidth** resource is positive. The default value of this resource is the client top shadow pixmap (specified by the **Mwm*topShadowPixmap** or **Mwm*client*topShadowPixmap** resources).

matteWidth

Class **MatteWidth**

Type **pixels**

Default **0**

Description

Specifies the width of the optional matte. The default value is 0, which effectively disables the matte.

maximumClientSize

Class **MaximumClientSize**

Type **integer x integer**

Default **fill the screen**

Description

Indicates, in pixel units, the client size to be used when an application is maximized. The resource value is specified as **widthxheight**. The width and height are interpreted in the units that the client uses (for example, terminal emulators generally use character units). If this resource is not specified, the maximum size from the **WM_NORMAL_HINTS** property is used if set. Otherwise the default value is the size where the client window with window management borders fills the screen. When the maximum client size is not determined by the **maximumClientSize** resource, the **maximumMaximumSize** resource value is used as a constraint on the maximum size.

useClientIcon

Class **UseClientIcon**

Type **Boolean**

Default **False**

Description

If this resource is set to **True**, a client supplied icon image takes precedence over a user supplied icon image. The default value is **False**, making the user supplied icon image have higher precedence than the client supplied icon image.

windowMenu

Class **WindowMenu**

Type **string**

Default **string**

Description

Indicates the name of the menu pane that is posted when the window menu is popped up (usually by pressing the left mouse button on the window menu button on the client window frame). Menu panes are specified in **$HOME/.mwmrc**, the window management resource description file. Window menus can be customized on a client class basis by specifying resources of the form **Mwm** ******client_name****window** *or* **Mwm** ******class****windowMenu**. The default value of this resource is the name of the built–in window menu specification.

AIXwindows Window Management Component Appearance Resource Set

The following resources control the appearance of components of AIXWindows Window Management:

background
backgroundPixmap
bottomShadowColor
bottomShadowPixmap
fontList
foreground
saveUnder
topShadowColor

background

Class **Background**

Type **color**

Default **Varies, based on visual type of screen.**

Description

Specifies the background color. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen. The default value is chosen based on the visual type of the screen.

backgroundPixmap

Class **BackgroundPixmap**

Type **string**

Default **Varies, based on visual type of screen.**

Description

Specifies the background pixmap of window management decoration when the window is inactive (does not have the keyboard focus). The default value is chosen based on the visual type of the screen.

bottomShadowColor

Class **Foreground**

Type **color**

Default **Varies, based on visual type of screen.**

Description

Specifies the bottom shadow color. This color is used for the lower and right sides of window management decoration. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen. The default value is chosen based on the visual type of the screen.

bottomShadowPixmap

Class **BottomShadowPixmap**

Type **string**

Default **Varies, based on visual type of screen.**

Description

Specifies the name of the image containing the bottom shadow pixmap. This pixmap is used for the lower and right sides of the window management decoration. The default is chosen based on the visual type of the screen.

fontList

Class **FontList**

Type **string**

Default **fixed**

Description

Specifies the font used in the window management decoration. The character encoding of the font should match the character encoding of the associated strings. The **/usr/lpp/X11/defaults/fonts** file contains a list of valid font names, along with the information that AIXwindows uses in displaying them.

foreground

Class **Foreground**

Type **color**

Default **Varies, based on visual type of screen.**

Description

Specifies the foreground color. The default is chosen based on the visual type of the screen. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen

saveUnder

Class **SaveUnder**

Type **Boolean**

Default **False**

Description

Indicates whether "save unders" are used for window management components. For this resource to have any effect, "save unders" must be implemented by the AIXwindows server. If "save unders" are implemented, the AIXwindows server will save the contents of windows obscured by windows that have the save under resource set. If the **saveUnder** resource is set to the **True** value, window management sets the **saveUnder** resource on the window management frame of any client that has it set. If the **saveUnder** resource is set to the **False** value, save unders will not be used on any window management frames.

topShadowColor

Class **Background**

Type **color**

Default **Varies, based on visual type of screen.**

Description

Specifies the top shadow color. This color is used for the upper and left sides of the window management decoration. The default is chosen based on the visual type of the screen.The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

topShadowPixmap

Class **TopShadowPixmap**

Type **string**

Default **Varies, based on visual type of screen.**

Description

Specifies the name of the file containing the top shadow pixmap. This pixmap is used for the upper and left sides of the window management decoration. The default is chosen based on the visual type of the screen.

activeBackground

Class **Background**

Type **Color**

Default **Varies, based on visual type of screen.**

Description

Specifies the background color of the window management decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

activeBackgroundPixmap

Class **ActiveBackgroundPixmap**

Type **string**

Default **Varies, based on visual type of screen.**

Description

Specifies the name of the file containing the background pixmap of the window management decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeBottomShadowColor

Class **Foreground**

Type **color**

Default **Varies, based on visual type of screen.**

Description

Specifies the bottom shadow color of the window management decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

activeBottomShadowPixmap

Class **BottomShadowPixmap**

Type **string**

Default **Varies, based on visual type of screen.**

Description

Specifies the bottom shadow pixmap of the window management decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeForeground

Class **Foreground**

Type **color**

Default **Varies, based on visual type of screen.**

Description

Specifies the name of the foreground color of the window management decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

activeTopShadowColor

Class **Background**

Type **color**

Default **Varies, based on visual type of screen.**

Description

Specifies the top shadow color of the window management decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen. The **/usr/lpp/X11/rgb/rgb.txt** file contains a listing of valid color names, and the information AIXwindows uses in displaying them on the screen.

activeTopShadowPixmap

Class **TopShadowPixmap**

Type **string**

Default **Varies, based on visual type of screen**

Description

Specifies the name of the file containing the top shadow pixmap of the window management decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

AIXwindows Window Management Specific Appearance And Behavior Resource Set

The following resources control interface appearance and behavior for AIXwindows Window Management:

autoKeyFocus
iconPlacementMargin
autoRaiseDelay
interactivePlacement
bitmapDirectory
keyBindings
buttonBindings
keyboardFocusPolicy
clean Text
limitResize
clientAutoPlace
lowerOnIconify
colormapFocusPolicy
maximumMaximumSize
configFile
moveThreshold
deiconifyKeyFocus
passButtons
doubleClickTime
passSelectButton
enforceKeyFocus
positionIsFrame
fadeNormalIcon
positionOnScreen
frameBorderWidth
quitTimeout
iconAutoPlace
resizeBorderWidth
iconBoxGeometry
resizeCursors
iconBoxName
showFeedback
iconBoxTitle
startupKeyFocus
iconClick
transientDecoration
iconDecoration
transientFunctions
iconImageMaximum
useIconBox
iconImageMinimum
wMenuButtonClick
iconPlacement
wMenuButtonClick2

autoKeyFocus

Class **AutoKeyFocus**

Type **Boolean**

Default **True**

Description

Only available when the keyboard input focus policy is explicit. If the **autoKeyFocus** resource is set to the **True** value when a window with the keyboard input focus is withdrawn from window management or is changed into an icon, the focus is set to the previous window that had the focus. If the value is set to the **False** value, there is no automatic setting of the keyboard input focus.

autoRaiseDelay

Class **AutoRaiseDelay**

Type **milliseconds**

Default **500**

Description

Only available when the **focusAutoRaise** resource is set to the **True** value and the keyboard focus policy is the **pointer** value. The **autoRaiseDelay** resource specifies the amount of time (in milliseconds) that window management waits before raising a window after it gets the keyboard focus. The default value of this resource is 500 (milliseconds).

bitmapDirectory

Class **BitmapDirectory**

Type **directory**

Default **/usr/include/X11/bitmaps**

Description

Identifies a directory to be searched for bitmaps referenced by window management resources. This directory is searched if a bitmap is specified without an absolute pathname. The default value for this resource is the **/usr/include/X11/bitmaps** value.

buttonBindings

Class **ButtonBindings**

Type **string**

Default **NULL**

Description

Identifies the set of button bindings for window management functions. The named set of button bindings is specified in the **$HOME/.mwmrc** file, the window resource description file. These button bindings are merged with the built-in default bindings. The default value for this resource is the **NULL** value (no button bindings are added to the built-in button bindings).

cleanText

Class **CleanText**

Type **Boolean**

Default **True**

Description

Controls the display of window management text in the client title and feedback windows. If the default **True** value is used, the text is drawn with a clear background (with no stippling). This makes text easier to read on monochrome systems when a background pixmap is specified. Only the stippling in the area immediately around the text is cleared. If the **cleanText** resource is set to the **False** value, the text is drawn directly on top of the existing background.

clientAutoPlace

Class **ClientAutoPlace**

Type **Boolean**

Default **True**

Description

Determines the position of a window when the window has not been given a user–specified position. If the **clientAutoPlace** resource is set to the **True** value, windows are positioned with the top left corners of the frames offset horizontally and vertically. If the **clientAutoPlace** resource is set to the **False** value, window management places the window at its currently configured position. In either case, window management attempts to place the windows completely on-screen.

colormapFocusPolicy

Class **ColormapFocusPolicy**

Type **string**

Default **keyboard**

Description

Indicates the color map focus policy to be used. If the resource value is set to the **explicit** value, color map selection action sets the colormap focus to that client window. If the value is set to the **pointer** value, the client window containing the pointer has the colormap focus. If the value is set to the **keyboard** value, the client window that has the keyboard input focus has the colormap focus. The default value for this resource is the **keyboard** value.

configFile

Class **ConfigFile**

Type **file**

Default **$HOME/.mwmrc**

Description

The resource value is the path name for a window management resource description file. The default is set to the user's **$HOME/.mwmrc** file. If this file does not exist, the default becomes the **/usr/lib/X11/$LANG/system.mwmrc** file (where **$LANG** represents the value of the user's **LANG** environment variable).

deiconifyKeyFocus

Class **DeiconifyKeyFocus**

Type **Boolean**

Default **True**

Description

Applies only when the keyboard input focus policy is explicit. If a value of **True** is used, a window receives the keyboard input focus when the user restores it from its icon. If a value of **False** is used, the keyboard input focus remains as it was.

doubleClickTime

Class **DoubleClickTime**

Type **millisecond**

Default **500**

Description

Sets the maximum time (in milliseconds) that can elapse between the two clicks (mouse button presses and releases) that make up a double-click. The default value of this resource is 500 (milliseconds).

enforceKeyFocus

Class **EnforceKeyFocus**

Type **Boolean**

Default **True**

Description

If the **enforceKeyFocus** resource is given a value of **True**, the keyboard input focus is always explicitly set to selected windows even if they are "globally–active" input windows (such as scroll bars that can be operated without setting the focus to their client.). If the **enforceKeyFocus** resource is set to the **False** value, the keyboard input focus is not explicitly set to globally–active windows.

fadeNormalIcon

Class **FadeNormalIcon**

Type **Boolean**

Default **False**

Description

If this resource is given a value of **True**, an icon is displayed as a grey silhouette whenever its window has been opened.

frameBorderWidth

Class **FrameBorderWidth**

Type **integer in pixel units**

Default **5**

Description

Specifies the width (in pixels) of a client window frame border without resize handles. The border width includes the 3–D shadows.

iconAutoPlace

Class **IconAutoPlace**

Type **Boolean**

Default **True**

Description

If the **iconAutoPlace** resource has a value of **True**, the window management places icons on the screen automatically. If the **iconAutoPlace** resource has a value of **False**, the user places the icons. Users can specify an initial icon position and can move icons after initial placement; the window management will adjust the user–specified position to fit into an invisible grid. When icons are automatically placed, window management places them into the grid according to a pattern set with the **iconPlacement** resource.

iconBoxGeometry

Class **IconBoxGeometry**

Type **string**

Default **6x1+0–0**

Description

Indicates the initial position and size of the icon box, in units of icon width and height. The value of the resource is a standard window geometry string with the following syntax:

```
[=][widthxheight][{+–}xoffset{+–}yoffset]
```

If the offsets are not provided, the icons are placed according to the value of the **iconPlacement** resource. The units for width and height are columns and rows.

The actual screen size of the icon box window depends on the **iconImageMaximum** and **iconDecoration** resources. The default value for the **iconImageMaximum** resource is (6*iconWidth + padding) wide by (1 * iconHeight + padding) high.

iconBoxName

Class **IconBoxName**

Type **string**

Default **iconbox**

Description

Specifies the name that is used to look up icon box resources. The default name is the **iconbox** value.

iconBoxTitle

Class **IconBoxTitle**

Type **string**

Default **Icons**

Description

Specifies the name that is used in the title area of the icon box frame. The default value is the **Icons** value.

iconClick

Class **IconClick**

Type **Boolean**

Default **True**

Description

When the **iconClick** resource is given the **True** value, the system menu is posted and left posted when the user clicks on an icon. If the **iconClick** resource is given the **False** value, the menu is not left posted.

iconDecoration

Class **IconDecoration**

Type **string**

Default **Varies**

Description

Specifies the general icon decoration. The **iconDecoration** resource can be set to the following values:

label Only the label part is displayed.

image Only the image part is displayed.

label image Both the label and image parts are displayed. When the icon is selected, the label is truncated.

activelabel Both the label and image parts are displayed.

The default icon decoration for **icon box** icons varies: each icon has a label part and an image part (label image).

The default icon decoration for **stand–alone** icons also varies: each icon has an active label part, a label part, and an image part (activelabel label image).

iconImageMaximum

Class **IconImageMaximum**

Type **width x height**

Default **50x50**

Description

Specifies the maximum size, in pixels, of the icon image. The resource value is **width**x**height** (for example, 64x64). The maximum supported size is 128x128. The default value of this resource is 50x50.

iconImageMinimum

Class **IconImageMinimum**

Type **width**x**height**

Default **32**x**32**

Description

Specifies the minimum size, in pixels, of the icon image. The resource value is **width**x**height** (e.g., 32x50). The minimum supported size is 16x16. The default value of this resource is 32x32.

iconPlacement

Class **IconPlacement**

Type **string**

Default **left bottom**

Description

Specifies the icon placement pattern to be used. The resource value has the following syntax:

```
primary_layout secondary_layout
```

where

primary_layout Indicates whether, when an icon placement is done, the icon is placed in a row or a column and the direction of placement

secondary_layout Indicates where to place new rows or columns.

Each value can be set to one of the following:

top Lays the icons out top to bottom.

bottom Lays the icons out bottom to top.

left Lays the icons out left to right.

right Lays the icons out right to left.

If the **primary_layout** value is set to **top** or **bottom**, the **secondary_layout** should be set to **left** or **right**.

For example, a value of **top right** indicates that icons should be placed top to bottom on the screen and that columns should be added from right to left on the screen. The default placement is the **left bottom** value (icons are placed left to right on the screen, with the first row on the bottom of the screen, and new rows added from the bottom of the screen to the top of the screen).

iconPlacementMargin

Class **IconPlacementMargin**

Type **pixels**

Default **varies**

Description

Sets the distance between the edge of the screen and the icons that are placed closest to the edge. The value should be greater than or equal to 0. If the value specified is invalid, window management uses a default value equal to the space between icons as they are placed on the screen (this space is based on maximizing the number of icons in each row and column).

interactivePlacement

Class **InteractivePlacement**

Type **Boolean**

Default **False**

Description

Controls the initial placement of new windows on the screen. If the value is set to the **True** value, the pointer shape changes before a new window is placed on the screen, indicating to the user that a position should be selected for the upper–left hand corner of the window. If the value is set to the **False** value, window management places windows according to the initial window configuration resources.

keyBindings

Class **KeyBindings**

Type **string**

Default **system dependent**

Description

This resource identifies the set of key bindings for window management functions. If specified, these key bindings replace the built–in default bindings. The named set of key bindings is specified in the window management resource description file. The default value for this resource is the set of system–compatible key bindings.

keyboardFocusPolicy

Class **KeyboardFocusPolicy**

Type **string**

Default **explicit**

Description

If this resource is set to the **pointer** value, the keyboard focus is set to the client window that contains the pointer or in the client window decoration that window management adds. If this resource is set to the **explicit** value, the keyboard focus is set to a client window when the user presses the left mouse button with the pointer on the client window or any part of the associated window management decoration.

limitResize

Class **LimitResize**

Type **Boolean**

Default **True**

Description

If this resource is set to the **False** value, the user is allowed to resize a window to greater than the maximum size. If the resource is set to the **True** value, the window's size is restricted to the maximum size.

lowerOnIconify

Class **LowerOnIconify**

Type **Boolean**

Default **True**

Description

If this resource is given the default **True** value, a window's icon appears on the bottom of the window stack when the window is minimized into an icon. A **False** value of places the icon in the stacking order at the same place as its associated window.

maximumMaximumSize

Class **MaximumMaximumSize**

Type **width x height (pixels)**

Default **(screen width * 2) x (screen height * 2)**

Description

Limits the maximum size of a client window as set by the user or client. The resource value is **widthxheight** (for example, 1024x1024) where the width and height are in pixels.

moveThreshold

Class **MoveThreshold**

Type **pixels**

Default **4**

Description

Controls the sensitivity of dragging operations that move windows and icons. The value of this resource indicates the number of pixels that the locator is moved with a button down before the move operation is started. This is used to prevent window or icon movement when the user clicks or double–clicks the mouse button and unintentionally moves the mouse pointer with the button down.

passButtons

Class **PassButtons**

Type **Boolean**

Default **False**

Description

If the resource is set to the **False** value, window management does not pass button press events to clients after processing them. If the resource is set to the **True** value, window management passes button presses to the client window. The window management function is done in either case.

passSelectButton

Class **PassSelectButton**

Type **Boolean**

Default **True**

Description

If the **passSelectButton** resource is set to the **True** value, window management passes mouse button presses used to select a window for keyboard input focus selection to the client in the window or uses the button press for window management operations, if appropriate (if the **keyboardFocusPolicy** resource is set to the **explicit** value). If the resource is set to the **False** value, the button press is not used for any operation other than selecting the window to be the keyboard input focus. The mouse button press selects the keyboard input focus in either case.

positionIsFrame

Class **PositionIsFrame**

Type **Boolean**

Default **True**

Description

Indicates how window management interprets client window position information (from the **WM_NORMAL_HINTS** property and from configuration requests). If the resource is set to the **True** value, window management interprets the information as the position of the client window frame. If the resource is set to the **False** value, window management interprets the information as the position of the client area of the window.

positionOnScreen

Class **PositionOnScreen**

Type **Boolean**

Default **True**

Description

If the **positionOnScreen** resource is set to the **True** value, indicates that, if possible, window management places windows initially so that they are not clipped by the edge of the screen. If a window is larger then the size of the screen, at least the upper left corner of the window will be on–screen.

If the resource is set to the **False** value, the window management places windows in the requested position even if the position is totally off–screen.

quitTimeout

Class **QuitTimeout**

Type **milliseconds**

Default **1000**

Description

Specifies the amount of time (in milliseconds) that window management waits for a client to update the **WM_COMMAND** property after window management has sent the **WM_SAVE_YOURSELF** message. The window management only uses this resource for those clients that have a WM_SAVE_YOURSELF atom and no WM_DELETE_WINDOW atom in the **WM_PROTOCOLS** client window property.

resizeBorderWidth

Class **ResizeBorderWidth**

Type **pixels**

Default **10**

Description

Specifies the width (in pixels) of a client window frame border with resize handles. The specified border width includes the 3–D shadows. The default is 10 (pixels).

resizeCursors

Class **ResizeCursors**

Type **Boolean**

Default **True**

Description

Indicates whether window management always displays the resize cursors when the pointer is in the window size border. If this resource is set to the **True** value, the cursors are shown; otherwise, the window management cursor is shown.

showFeedback

Class **ShowFeedback**

Type **string**

Default **all**

Description

Controls whether window management displays window position and size feedback during move or resize operations, and initial client placement and window management message and dialog boxes. The value for this resource is a list of names of the feedback options to be enabled; the names must be separated by a space. The names of the feedback options are as follows:

Name	Description
all	Shows all feedback. (Default value.)
behavior	Confirms behavior switch.
move	Shows position during move.
none	Shows no feedback.
placement	Shows position and size during initial placement.
resize	Shows size during resize
restart	Confirms window management restart.

Example

The following command line illustrates the syntax for the **showFeedback** resource:

```
Mwm*showFeedback: placement resize behavior restart
```

This resource specification provides feedback for initial client placement and resize, and enables the dialog boxes to confirm the restart and set behavior functions. It disables feedback for the move function.

startupKeyFocus

Class **StartupKeyFocus**

Type **Boolean**

Default **True**

Description

Available only when the keyboard input focus policy is set to the **explicit** value. If the **startupKeyFocus** resource is set to the **True** value, a window gets the keyboard input focus when the window is initially managed. If the **startupKeyFocus** value is set to the **False** value, the window does not get the keyboard input focus.

transientDecoration

Class **TransientDecoration**

Type **string**

Default **menu title**

Description

This controls the amount of decoration that window management puts on transient windows. Transient windows are identified by the **WM_TRANSIENT_FOR** property which is added by the client to indicate a relatively temporary window. The default value for this resource is the **menu title** value (transient windows have resize borders and a titlebar with a window menu button). The names of the feedback options are:

Name	Description
all	Includes all decorations (default value).
border	Includes window border.
maximize	Includes maximize button (includes title bar).
minimize	Includes minimize button (includes title bar).
none	Includes no decorations.
resizeh	Includes border resize handles (includes border).
menu	Includes window menu button (includes title bar).
title	Includes title bar (includes border).

transientFunctions

Class **TransientFunctions**

Type **string**

Default **–minimize –maximize**

Description

Indicates which window management functions apply (or do not apply) to transient windows. The **transientFunctions** resource can have the following values:

Name	Description
all	Includes all decorations (default value).
border	Includes window border.
maximize	Includes maximize button (includes title bar).
minimize	Includes minimize button (includes title bar).
none	Includes no decorations.
resizeh	Includes border resize handles (includes border).
menu	Includes window menu button (includes title bar).
title	Includes title bar (includes border).

useIconBox

Class **UseIconBox**

Type **Boolean**

Default **False**

Description

If this resource is given of the **True** value, icons are placed in an icon box. When an icon box is not used, the icons are placed on the root window.

wMenuButtonClick

Class **WMenuButtonClick**

Type **Boolean**

Default **True**

Description

Indicates whether a click of the mouse when the pointer is over the window menu button posts the system menu and leaves it posted. If this resource is set to the **True** value, then the menu remains posted.

wMenuButtonClick2

Class **WMenuButtonClick2**

Type **Boolean**

Default **True**

Description

When this resource is given the default of **True** value, a double–click action on the window menu button performs an **f.kill** function.

AIXwindows Window Management Resource Description File Functions

- f.beep
- f.circle_down
- f.circle_up
- f.exec or !
- f.focus_color
- f.focus_key
- f.kill
- f.lower
- f.maximize
- f.menu
- f.minimize
- f.move
- f.next_cmap
- f.next_key
- f.nop
- f.normalize
- f.pack_icons
- f.pass_keys
- f.post_wmenu
- f.prev_cmap
- f.prev_key
- f.quit_mwm
- f.raise
- f.raise_lower
- f.refresh
- f.refresh_win
- f.resize
- f.restart
- f.send_msg
- f.separator
- f.set_behavior
- f.title

f.beep

Syntax

f.beep

Description

Causes a terminal to beep.

f.circle_down

Syntax

f.circle_down icon

f.circle_down window

Description

Causes the window or icon on the top of the window stack to be put on the bottom of the window stack (so that it no longer obscures any other window or icon). This function affects only windows and icons that obscure other windows and icons, or that are obscured by other windows and icons.

Secondary windows are restacked with their associated primary window. Secondary windows always stay on top of their associated primary window. No other primary windows can exist between the secondary windows and their primary window.

If an icon function argument is specified, the function applies only to icons; if a window function argument is specified, the function applies only to windows.

f.circle_up

Syntax

f.circle_up icon

f.circle_up window

Description

Raises the window or icon on the bottom of the window stack to the top (so that it is not obscured by any other windows). This function affects only windows and icons that obscure other windows and icons, or that are obscured by other windows and icons.

Secondary (transient) windows are restacked with their associated primary window. If an icon function argument is specified, the function applies only to icons; if a window function argument is specified, the function applies only to windows.

f.exec or !

Syntax

f.exec *Command*

! *Command*

Description

Runs a shell command (using the value of the **$SHELL** environment variable if it is set, otherwise /**bin**/**sh**).

f.focus_color

Syntax

f.focus_color

Description

Sets the colormap focus to a client window. If this function is done in a root context, then the default colormap (set up by the AIX X–Windows System for the screen where window management is running) is installed and there is no specific client window colormap focus. This function is treated as the **f.nop** function if the **colormapFocusPolicy** resource is not set to the **explicit** value.

f.focus_key

Syntax

f.focus_key

Description

Sets the keyboard input focus to a client window or icon. This function is treated as the **f.nop** function (and does nothing) if the **keyboardFocusPolicy** resource is not set to the **explicit** value or the function is executed in a root context.

f.kill

Syntax

f.kill

Description

If the **WM_DELETE_WINDOW** protocol is set up, the client is sent a client message event indicating that the client window should be deleted. If the **WM_SAVE_YOURSELF** protocol is set up and the **WM_DELETE_WINDOW** protocol is not set up, the client is sent a client message event indicating that the client needs to prepare to end. If the client does not have the **WM_DELETE_WINDOW** or **WM_SAVE_YOURSELF** protocol set up, this function causes a client's AIXwindows connection to end (usually resulting in the closing of the client).

f.lower

Syntax

f.lower [*–Client*]

Description

Lowers a client window to the bottom of the window stack, behind the other windows). Secondary windows are restacked with their associated primary window.

The *Client* parameter indicates the name or class of a client to lower. If the *Client* parameter is not specified, the context in which the function was invoked indicates the window or icon to lower.

f.maximize

Syntax

f.maximize

Description

Causes a client window to be displayed with its maximum size.

f.menu

Syntax

f.menu *MenuName*

Description

Associates a pull–down menu with a menu pane entry or a menu with a button or key binding. The *MenuName* parameter identifies the menu to be used.

f.minimize

Syntax

f.minimize

Description

Causes a client window to be minimized (changed into an icon). When a window is minimized and no icon box is used, its icon is placed on the bottom of the window stack, behind the other windows. If an icon box is used, the client's icon changes to its iconified form inside the icon box. A secondary window is minimized with its associated primary window. There is only one icon for each primary window and all of its secondary windows.

f.move

Syntax

f.move

Description

Allows a client window to be moved interactively.

f.next_cmap

Syntax

f.next_cmap

Description

Installs the next colormap in the list of colormaps for the window with the colormap focus.

f.next_key

Syntax

f.next_key *Type*

The *Type* parameter has a value of **icon**, **window**, or **transient**.

Description

Sets the keyboard input focus to the next window or icon in the set of windows or icons managed, as determined by the stacking of windows on the screen. This function is treated as the **f.nop** function (and does nothing) if the **keyboardFocusPolicy** resource is not set to the **explicit** value.

If the **transient** argument is specified, all secondary windows are traversed; otherwise, traversal is done only to the last focused window in a transient group. If an **icon** function argument is specified, the function applies only to icons; if a **window** function argument is specified, the function applies only to windows.

f.nop

Syntax

f.nop

Description

Does nothing.

f.normalize

Syntax

f.normalize

Description

Causes a client window to be displayed with its normal size. Secondary windows are placed in their normal state along with their associated primary window.

f.pack_icons

Syntax

f.pack_icons

Description

Rearranges icons on the root window or in the icon box (depending on the layout policy being used). In general, this causes icons to be packed into the icon grid.

f.pass_keys

Syntax

f.pass_keys

Description

Enables or disables processing of key bindings for window management functions. When it disables key binding processing, all keys are passed on to the window with the keyboard input focus and no window management functions are invoked. If the **f.pass_keys** function is invoked with a key binding to disable key binding processing, the same key binding can be used to enable key binding processing.

f.post_wmenu

Syntax

f.post_wmenu

Description

Posts the window menu. If a key is used to post the window menu and a window menu button is present, the window menu is automatically placed with its top–left corner at the bottom–left corner of the window menu button for the client window. If no window menu button is present, the window menu is placed at the top–left corner of the client window.

f.prev_cmap

Syntax

f.prev_cmap

Description

Installs the previous colormap in the list of colormaps for the window with the colormap focus.

f.prev_key

Syntax

f.prev_key *Type*

The *Type* parameter has a value of **icon**, **window**, or **transient**.

Description

Sets the keyboard input focus to the previous window or icon in the set of windows or icons (as determined by the stacking of windows on the screen). This function is treated as the **f.nop** value (and does nothing) if the **keyboardFocusPolicy** resource is not set to the **explicit** value.

If the **transient** argument is specified, secondary windows are traversed; otherwise, traversal is done only to the last focused window in a transient group. If an **icon** function argument is specified, the function applies only to icons; if a **window** function argument is specified, the function applies only to windows.

f.quit_mwm

Syntax

f.quit_mwm

Description

Ends window management, but not the Enhanced Windows system.

f.raise

Syntax

f.raise [*Client*]

Description

Raises a client window to the top of the window stack, so that the other windows are behind it. Secondary windows are restacked with their associated primary window.

The *Client* parameter indicates the name or class of a client to raise. If the *Client* parameter is not specified, the context in which the function was invoked indicates the window or icon to raise.

f.raise_lower

Syntax

f.raise_lower

Description

If the client window is at the top of the window stack, this function lowers it to the bottom. Otherwise, it raises it to the top. Secondary windows are restacked with their associated primary window.

f.refresh

Syntax

f.refresh

Description

Redraws all windows.

f.refresh_win

Syntax

f.refresh_win

Description

Redraws a client window.

f.resize

Syntax

f.resize

Description

Allows a client window to be resized interactively.

f.restart

Syntax

f.restart

Description

Causes window management to be restarted (effectively ending it and restarting it).

f.send_msg

Syntax

f.send_msg *MessageNumber*

Description

Sends a client message of the **_MOTIF_WM_MESSAGES** type (the message type is indicated by the function's *MessageNumber* parameter). The client message will only be sent if the message number is included in the client's **_MOTIF_WM_MESSAGES** property. A menu item label is displayed in gray if the menu item is used to do a **f.send_msg** function for a message that is not included in the client's **_MOTIF_WM_MESSAGES** property.

f.separator

Syntax

f.separator

Description

Places a menu separator in the menu pane at the specified location (the label is ignored).

f.set_behavior

Syntax

f.set_behavior

Description

Causes window management to restart with the default OSF behavior (if a custom behavior is configured) or a custom behavior (if an OSF default behavior is configured).

f.title

Syntax

f.title

Description

Inserts a title in the menu pane at the specified location.

AIXwindows Window Management Event Specifications

Events are indicated as part of the specifications for button binding sets, key binding sets, and menu panes.

Button events have the following syntax:

Button = [*ModifierList*]<*ButtonEventName*>
ModifierList = *ModifierName {ModifierName}*

All specified modifiers within modifier lists are exclusive; only the specified modifiers can be present when the button event occurs. The following values can be used for *ModifierName*:

Modifier	Description
Ctrl	Control Key
Shift	Shift Key
Alt	Alt or Meta Key
Meta	Meta or Alt Key
Lock	Lock Key
Mod1	Modifier1
Mod2	Modifier2
Mod3	Modifier3
Mod4	Modifier4
Mod5	Modifier5

The following values can be used for *ButtonEventName*:

Button	Description
Btn1Down	Button 1 Press
Btn1Up	Button 1 Release
Btn1Click	Button 1 Press and Release
Btn1Click2	Button 1 Double Click
Btn2Down	Button 2 Press
Btn2Up	Button 2 Release
Btn2Click	Button 2 Press and Release
Btn2Click2	Button 2 Double Click
Btn3Down	Button 3 Press
Btn3Up	Button 3 Release
Btn3Click	Button 3 Press and Release
Btn3Click2	Button 3 Double Click
Btn4Down	Button 4 Press
Btn4Up	Button 4 Release
Btn4Click	Button 4 Press and Release
Btn4Click2	Button 4 Double Click
Btn5Down	Button 5 Press
Btn5Up	Button 5 Release

Btn5Click Button 5 Press and Release

Btn5Click2 Button 5 Double Click

The key events that window management uses for menu mnemonics and for binding to window management functions are single key presses; key releases are ignored. Key events have the following syntax:

Key = [ModiferList]<Key>KeyName
ModifierList = ModifierName {ModifierName}

All specified modifiers are exclusive; only the specified modifiers can be present when the button event occurs. Modifiers for keys are the same as those that apply to buttons. The key name is an AIXwindows keysym name, as listed in the **keysymdef.h** file, with the **XK_** prefix removed.

AIXwindows Window Management Button Bindings

The **buttonBindings** resource value is the name of a set of button bindings that configure window management behavior. Users can invoke window management functions by moving the mouse pointer and pressing a button over a framed client window, an icon, or the root window. The context for indicating where the button press applies is also the context for calling the window management function when the button press is done.

The button binding syntax is the following:

buttons_bindings_set_name {
Button Context Function
Button Context Function
...
Button Context Function **}**

The syntax for the context specification is the following:

Context = Object[| *context*]

The *Object* parameter has a value of **root**, **icon**, **window**, **title**, **frame**, **border**, or **app**.

The *Context* parameter indicates where the pointer must be for the button binding to be effective. The *Context* parameter can be set to the following values:

window	A client window or window management frame for the button binding to be effective.
frame	The window management frame around a client window (including the border and titlebar).
border	The border part of the window management frame (not including the titlebar).
title	The title area of the window management frame.
app	The application window (not including the window management frame).

If an **f.nop** function is specified for a button binding, the button binding is not done.

AIXwindows Window Management Key Bindings

The **keyBindings** resource value is the name of a set of key bindings that are used to configure window management behavior. A window management function can be invoked by pressing a particular key. The context in which the key binding applies is indicated in the key binding specification. The valid contexts are the same as those that apply to button bindings.

The key binding syntax is the following:

key_bindings_set_name {

Key Context Function
Key Context Function

. . .

KeyContext Function **}**

If an **f.nop** function is specified for a key binding, the key binding is not done. If an **f.post_wmenu** or **f.menu** function is bound to a key, the same key can automatically remove the menu from the screen after the menu pops up.

The *Context* parameter syntax is the same as for button bindings. For key bindings, the **frame**, **title**, **border**, and **app** contexts are equivalent to the window context. The context for a key event is the window or icon that has the keyboard input focus (or the root window if no window or icon has the keyboard input focus).

AIXwindows Window Management Menu Panes

Menus can be popped up using the **f.post_wmenu** and **f.menu** window management functions. The context for window management functions that are invoked from a menu is the **root**, **icon** or **window** context depending on the manner in which the menu was popped up. In the case of the window menu or menus popped up with a key binding, the location of the keyboard input focus indicates the context. For menus popped up using a button binding, the context of the button binding is the context of the menu.

The menu pane specification syntax is the following:

Menu menu_name {
Label [*Mnemonic*] [*Accelerator*] *Function*
Label [*Mnemonic*] [*Accelerator*] *Function*
...
Label [*Mnemonic*] [*Accelerator*] *Function*}

Each line in the menu specification identifies the label for a menu item and the function to be invoked if the menu item is selected. Optionally a menu button mnemonic and a menu button keyboard accelerator can be specified. Mnemonics are functional only when the menu is posted and users can select menu items by typing the mnemonic letter.

The label specification has the following syntax:

Label = String

If the string is preceded by the @ character, it is the name of a bitmap file. Otherwise, it indicates the string that is displayed as the title. The string can be enclosed in quotes.

The string encoding for labels must be compatible with the menu font being used. Labels are greyed out for menu items that do the **f.nop** function or that do functions that are not valid or functions that do not apply in the current context.

A mnemonic specification has the following syntax:

mnemonic = _character

The first matching character in the label is underlined. If there is no matching character in the label, no window management mnemonic is registered for that label. Although the character must exactly match a character in the label, the mnemonic does not execute if any modifier (such as the **Shift** modifier) is pressed with the character key.

The accelerator specification is a key event specification with the same syntax as is used for key bindings to window management functions.

Enhanced X–Windows Toolkit Subroutines

MenuPopdown Translation Action

Purpose

Pops down a spring-loaded menu.

Library

Intrinsics Library (**libXt.a**)

Syntax

void MenuPopdown(*ShellName***)**
String *ShellName*;

Description

The **MenuPopdown** translation action pops down a spring-loaded menu when a pointer button is released or when the pointer is moved into a window.

If a shell name is not specified, the **MenuPopdown** translation action calls the **XtPopdown** subroutine with the widget for which the translation is specified.

If a shell name is specified in the translation table, the **MenuPopdown** translation action tries to find the shell by searching the widget tree. It starts at the parent of the widget where it was called. If it finds a shell with the specified name in the pop-up children of that parent, it pops down the shell. Otherwise, it moves up the parent chain as needed.

If the **MenuPopdown** translation action gets to the application top-level widget and cannot find a matching shell, it generates an error.

Parameter

ShellName Specifies the name of the widget shell to be popped down.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCallbackPopdown** callback procedure.

The **MenuPopup** translation action.

MenuPopup Translation Action

Purpose

Pops up a menu.

Library

Intrinsics Library (**libXt.a**)

Syntax

void MenuPopup(*ShellName***)**
String *ShellName*;

Description

The **MenuPopup** translation action pops up a menu when a pointer button is pressed or when the pointer is moved into a window.

The **MenuPopup** translation action is known to the translation manager, which must perform special actions for spring-loaded pop-ups. Calls to the **MenuPopup** translation action in a translation specification are mapped into calls to a non-exported action procedure. The translation manager fills in fields based on the event specified on the left-hand side of a translation.

If the **MenuPopup** translation action is invoked upon a **ButtonPress** event, possibly with modifiers, the translation manager pops up the shell with the *grab_kind* field set to the **XtGrabExclusive** value and the *spring_loaded* field set to the **True** value.

If the **MenuPopup** translation action is invoked upon the **EnterWindow** event, possibly with modifiers, the translation manager pops up the shell with the *grab_kind* field set to the **XtGrabNonexclusive** value and the *spring_loaded* field set to the **False** value. Otherwise, the translation manager generates an error.

When the widget is popped up, the **MenuPopup** translation action does the following:

- Calls the **XtCheckSubclass** subroutine to ensure that the pop-up shell is a subclass of the **Shell** widget
- Generates an error if the *popped_up* field of the shell already is the value of **True**
- Calls the callback procedures on the *popup_callback* list of the shell
- Sets the *popped_up* field of the shell to the value of **True**
- Sets the *grab_kind* and *spring_loaded* fields of the shell appropriately
- Calls the *create_popup_child* field of the shell with the *popup_shell* field if the *create_popup_child* field of the shell is not the value of **NULL**
- Calls the **XtAddGrab** subroutine with the *popup_shell* and *spring_loaded* fields set as specified and the *grab_kind* field set to the **XtGrabExclusive** value
- Calls the **XtRealizeWidget** subroutine with the *popup_shell* field set as specified
- Calls the **XMapWindow** subroutine with the *popup_shell* field set as specified

The **MenuPopup** translation action tries to find the shell by searching up the widget tree starting at the parent of the widget in which this routine is invoked. If it finds a shell with the

specified name in the pop-up children of that parent, it pops up the shell with the appropriate fields. Otherwise, it moves up the parent chain as needed.

The **MenuPopup** translation action generates an error if it gets to the application widget and cannot find a matching shell.

Parameter

ShellName Specifies the name of the widget shell to be popped up.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **MenuPopdown** translation action.

XtAddActions Subroutine

Purpose

Declares an action table and registers it with the translation manager.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAddActions(*Actions*, *NumberActions*)
XtActionList *Actions*;
Cardinal *NumberActions*;

Description

Note: The **XtAddActions** subroutine is left over from a previous release of Enhanced X-Windows, and is provided as a convenience to users converting from these earlier versions.

The **XtAddActions** subroutine declares an action table and registers it with the translation manager. If more than one action with the same name is registered, the most recently registered action is used. If duplicate actions exist in an action table, the first entry is used.

The Intrinsics library registers an action table for the **MenuPopup** translation action and the **MenuPopdown** translation action as part of the tool kit initialization.

Parameters

Actions	Specifies the action table to be registered.
NumberActions	Specifies the number of entries in this action table.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddActions** subroutine.

XtAddCallback Subroutine

Purpose

Adds a callback procedure to the callback list of a specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAddCallback(*WidgetID* **,** *CallbackName*,
Callback, *ClientData*)
Widget *WidgetID*;
String *CallbackName*;
XtCallbackProc *Callback*;
caddr_t *ClientData*;

Description

The **XtAddCallback** subroutine adds a callback procedure to the callback list of the specified widget. A callback is invoked as many times as it occurs in the callback list.

Parameters

WidgetID	Specifies the widget.
CallbackName	Specifies the callback list to which the procedure is to be appended.
Callback	Specifies the callback procedure.
ClientData	Specifies the argument for the callback if the callback is invoked by the **XtCallCallbacks** subroutine. Otherwise, this parameter is set to **NULL**..

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddCallbacks** subroutine, **XtRemoveCallback** subroutine, **XtRemoveCallbacks** subroutine, **XtRemoveAllCallbacks** subroutine, **XtCallCallbacks** subroutine, **XtHasCallbacks** subroutine.

The **XtCallbackRec** type, **XtCallbackProc** type.

XtAddCallbacks Subroutine

Purpose

Adds a list of callback procedures to the callback list of the specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAddCallbacks(*WidgetID*, *CallbackName*, *Callbacks***)**
Widget *WidgetID*;
String *CallbackName*;
XtCallbackList *Callbacks*;

Description

The **XtAddCallbacks** subroutine adds a list of callback procedures to the callback list of specified widget. To handle the callback lists correctly, declare the **XtCallbackList** with a resource type of **XtRCallback**.

Parameters

WidgetID	Specifies the widget.
CallbackName	Specifies the callback list where the procedure will be appended.
Callbacks	Specifies the null-terminated list of callback procedures and corresponding client data.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddCallback** subroutine, **XtRemoveCallback** subroutine, **XtRemoveCallbacks** subroutine, **XtRemoveAllCallbacks** subroutine, **XtCallCallbacks** subroutine, **XtHasCallbacks** subroutine.

The **XtCallbackRec** type, **XtCallbackProc** type.

XtAddConverter Subroutine

Purpose

Registers a new converter.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAddConverter(*FromType*, *ToType*, *Converter*, *ConvertArguments*, *NumberArguments*)
String *FromType*;
String *ToType*;
XtConverter *Converter*;
XtConvertArgList *ConvertArguments*;
Cardinal *NumberArguments*;

Description

The **XtAddConverter** subroutine registers a new converter. If the type converter does not require additional arguments, the *ConvertArguments* parameter is **NULL**, and the *NumberArguments* parameter is **0**.

For converters that require additional arguments, use the **XtAddressMode** enumerated type and the **XtConvertArgRec** data structure to specify how each argument is derived. These are defined in the <**X11/Convert.h**> header file.

Note: This subroutine exists only as a convenience to users converting from earlier versions of the toolkit; it has been replaced by the **XtAppAddConverter** subroutine.

Parameters

FromType	Specifies the source type.
ToType	Specifies the destination type.
Converter	Specifies the type converter procedure.
ConvertArguments	Specifies how to compute the additional arguments to the converter. If the type converter does not require additional arguments, this parameter is the value of **NULL**..
NumberArguments	Specifies the number of additional arguments to the converter. If the type converter does not require additional arguments, this parameter is the value of **0**.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The <**X11/Convert.h**> header file.

The **XtAddressMode** enumerated type, **XtConvertArgRec** data structure.

XtAddEventHandler Subroutine

Purpose

Registers an event handler procedure with the dispatch mechanism.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAddEventHandler(*WidgetID*, *EvtMask*, *Nonmaskable*, *Procedure*, *ClientData*)
Widget *Widget*;
EventMask *EvtMask*;
Boolean *Nonmaskable*;
XtEventHandler *Procedure*;
caddr_t *ClientData*;

Description

The **XtAddEventHandler** subroutine registers an event handler procedure with the dispatch mechanism that is to be called when an event matching the mask occurs on the specified widget. If the widget is realized, the **XtAddEventHandler** subroutine calls the **XSelectInput** subroutine (if necessary).

If the event handler *Procedure* is registered with the same *ClientData* already, the specified mask is ORed into the existing mask.

Parameters

WidgetID	Specifies the widget for which the event handler is being registered.
EvtMask	Specifies the event mask for this procedure.
Nonmaskable	Specifies a Boolean value that indicates if this procedure should be called on the non-maskable events. The non-maskable events are the following: **ClientMessage** **SelectionClear** **GraphicsExpose** **SelectionNotify** **MappingNotify** **SelectionRequest** **NoExpose**
Procedure	Specifies the event handler procedure to be called.
ClientData	Specifies additional data for the client's event handler.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtEventHandler** data type.

The **XtRemoveEventHandler** subroutine, **XtAddRawEventHandler** subroutine, **XtRemoveRawEventHandler** subroutine, **XtBuildEventMask** subroutine, **XtRealizeWidget** subroutine.

The **XSetWindowAttributes** data structure.

The **XtAllEvents** event mask.

XtAddExposureToRegion Subroutine

Purpose

Merges **Expose** and **GraphicsExpose** events into a region.

Library

Intrinsics Library (**libXt.a**)

C Syntax

void XtAddExposureToRegion(*Event, RegionPtr***)**
XEvent **Event*;
Region *RegionPtr*;

Description

The **XtAddExposureToRegion** subroutine merges **Expose** events and **GraphicsExpose** events into a region clients can process at once rather than processing individual rectangles. This subroutine computes the union of the rectangle defined by the specified exposure event and region; it then stores the results back in the specified region.

If the *Event* parameter is not an **Expose** event or **GraphicsExpose** event, the **XtAddExposureToRegion** subroutine returns without an error and does not modify the region.

Parameters

Event	Specifies a pointer to the **Expose** event or **GraphicsExpose** event.
RegionPtr	Specifies the region object as defined in the **<X11/Xutil.h>** header file.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The Exposure Compression Mechanism.

XtAddGrab Subroutine

Purpose

Redirects user input to a modal widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAddGrab(*WidgetID*, *Exclusive*, *SpringLoaded***)**
Widget *WidgetID*;
Boolean *Exclusive*;
Boolean *SpringLoaded*;

Description

The **XtAddGrab** subroutine redirects user input to a modal widget. Modal widgets are widgets that disable user-event processing by an application, except for events that occur in the dialog box. This subroutine appends the widget and its associated parameters to the modal cascade and checks that the *Exclusive* parameter is the value of **True** if the *SpringLoaded* parameter is the value of **True**. Otherwise, it generates an error.

The modal cascade is used by the **XtDispatchEvent** subroutine when it tries to dispatch a user event. When at least one modal widget is in the widget cascade, the **XtDispatchEvent** subroutine first determines if the event should be delivered. It starts at the most recent cascade entry added with the *Exclusive* parameter of the value of **True**.

This subset of the modal cascade along with all descendants of these widgets comprise the active subset. User events that occur outside the widgets in this subset are ignored or remapped. Modal menus with submenus generally add a submenu widget to the cascade with the *Exclusive* parameter of the value of **False**. Modal dialog boxes that need to restrict user input to the most deeply nested dialog box add a subdialog widget to the cascade with the *Exclusive* parameter of the value of **True**. User events that occur within the active subset are delivered to the appropriate widget, which is usually a child or further descendant of the modal widget.

Regardless of where they occur on the screen, remap events are always delivered to the most recent widget in the active subset of the cascade that has the *SpringLoaded* parameter set to the value of **True**, if any such widget exists.

Parameters

WidgetID	Specifies the widget to add to the modal cascade.
Exclusive	Specifies whether user events should be dispatched exclusively to this widget or dispatched also to previous widgets in the cascade.
SpringLoaded	Specifies if this widget was popped up by pressing a pointer button.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRemoveGrab** subroutine, **XtPopup** subroutine.

The **KeyPress** remap event, **KeyRelease** remap event, **ButtonPress** remap event, **ButtonRelease** remap event.

XtAddInput Subroutine

Purpose

Registers a new source of events with the default X Toolkit application.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtInputId XtAddInput(*Source*, *Condition*, *Procedure*,
ClientData)
int *Source*;
caddr_t *Condition*;
XtInputCallbackProc *Procedure*;
caddr_t *ClientData*;

Description

The **XtAddInput** subroutine registers a new source of events, usually as file input or file output, with the X Toolkit default application. (In this case, "file" represents any source of data.) This subroutine also specifies the conditions under which the source can generate events. When input on this source in the default application context is pending, the callback procedure is called.

The **XtInitialize** subroutine must be called before using the **XtAddInput** subroutine.

Note: This subroutine exists only as a convenience for users converting earlier versions of the toolkit. The **XtAddInput** subroutine has been replaced by the **XtAppAddInput** subroutine.

Parameters

Source	Specifies the source file descriptor or another system-dependent device specification.
Condition	Specifies the mask that indicates a read, write, or exception condition or another operating system dependent condition.
Procedure	Specifies the callback procedure when input is available.
ClientData	Specifies the parameter to be passed to the callback procedure when input is available.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddInput** subroutine.

XtAddRawEventHandler Subroutine

Purpose

Registers an event handler procedure with the dispatch mechanism, without causing the server to select for that event.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAddRawEventHandler(*WidgetID*, *EvtMask*, *Nonmaskable*,
Procedure, *ClientData*)
Widget *WidgetID*;
EventMask *EvtMask*;
Boolean *Nonmaskable*;
XtEventHandler *Procedure*;
caddr_t *ClientData*;

Description

The **XtAddRawEventHandler** subroutine registers an event handler procedure with the dispatch mechanism without causing the server to select for that event. This subroutine is similar to the **XtAddEventHandler** subroutine except that it does not affect the mask of the widget or call the **XSelectInput** subroutine for its events. Note that the widget might already have those mask bits set because of other non-raw event handlers registered on it.

Parameters

WidgetID	Specifies the widget for this event handler.
EvtMask	Specifies the event mask for this procedure.
Nonmaskable	Specifies a Boolean value that indicates if this procedure should be removed on the nonmaskable events. The nonmaskable events are the following:

ClientMessage	**SelectionClear**
GraphicsExpose	**SelectionRequest**
MappingNotify	**SelectionNotify**
NoExpose	

Procedure	Specifies the client event handler procedure to be registered.
ClientData	Specifies additional data for the client event handler.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddEventHandler** subroutine, **XtRemoveEventHandler** subroutine, **XtRemoveRawEventHandler** subroutine, **XtBuildEventMask** subroutine, **XtRealizeWidget** subroutine.

The **XSetWindowAttributes** data structure.

The **XtEventHandler** type.

The **XtAllEvents** event mask.

XtAddTimeOut Subroutine

Purpose

Creates a time-out value in the default application context.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtIntervalId XtAddTimeOut(*Interval*, *Procedure*, *ClientData***)**
unsigned long *Interval*;
XtTimerCallbackProc *Procedure*;
caddr_t *ClientData*;

Description

The **XtAddTimeOut** subroutine creates a time-out value in the default application context and returns an identifier for it. The time-out value is set in the *Interval* parameter. The callback procedure is called when the time interval elapses, after which the time-out is removed.

The **XtInitialize** subroutine must be called before using this subroutine.

Note: This subroutine exists only as a convenience for users converting earlier versions of the X Toolkit. The **XtAddTimeOut** subroutine has been replaced by the **XtAppAddTimeOut** subroutine.

Parameters

Interval	Specifies the time interval in milliseconds.
Procedure	Specifies the callback procedure.
ClientData	Specifies the parameters to be passed on to the callback procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddTimeOut** subroutine, **XtInitialize** subroutine

The **XtInputCallbackProc** type.

XtAddWorkProc Procedure

Purpose

Registers a work procedure in the default application context.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtWorkProcId XtAddWorkProc (*Procedure*, *Closure*)
XtWorkProc *Procedure*;
Opaque *Closure*;

Description

The **XtAddWorkProc** procedure registers a work procedure in the default application context. The **XtInitialize** subroutine must be called before using this routine.

Note: This procedure exists only as a convenience for users converting earlier versions of the X Toolkit. The **XtAddWorkProc** procedure has been replaced by the **XtAppAddWorkProc** procedure.

Parameters

Procedure Specifies the work procedure.

Closure Specifies the client data to be passed to the procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddWorkProc** subroutine

The **XtWorkProc** data type.

XtAppAddActions Subroutine

Purpose

Declares an action table and registers it with the translation manager.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppAddActions(*ApplicationContext*, *Actions*, *NumberActions*)
XtAppContext *ApplicationContext*;
XtActionList *Actions*;
Cardinal *NumberActions*;

Description

The **XtAppAddActions** subroutine declares an action table and registers it with the translation manager.

If more than one action is registered with the same name, the most recently registered action is used. If duplicate actions exist in an action table, the first action is used.

The Intrinsics library registers an action table for the **MenuPopup** and **MenuPopdown** translation actions as part of X Toolkit initialization.

Parameters

ApplicationContext	Specifies the application context.
Actions	Specifies the action table to register.
NumberActions	Specifies the number of entries in this action table.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **MenuPopup** translation action, **MenuPopdown** translation action.

XtAppAddConverter Subroutine

Purpose

Registers a new converter.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppAddConverter(*ApplicationContext*, *FromType*, *ToType*, *Converter*, *ConvertArguments*, *NumberArguments*)
XtAppContext *ApplicationContext*;
String *FromType*;
String *ToType*;
XtConverter *Converter*;
XtConvertArgList *ConvertArguments*;
Cardinal *NumberArguments*;

Description

The **XtAppAddConverter** subroutine registers a new converter. If the same value for the *FromType* and *ToType* parameters is specified in two calls to the **XtAppAddConverter** subroutine, the second call overrides the first call.

For converters that require additional arguments, use the **XtAddressMode** structure and the **XtConvertArgRec** structure to specify how each argument is derived. These are defined in the <**X11/Convert.h**> header file.

Parameters

ApplicationContext	Specifies the application context.
FromType	Specifies the source type.
ToType	Specifies the destination type.
Converter	Specifies the type converter procedure.
ConvertArguments	Specifies how to compute the additional arguments to the converter. If the *Converter* parameter does not contain additional arguments, this parameter is the value of **Null**.
NumberArguments	Specifies the number of additional arguments to the converter. If the *Converter* parameter does not contain additional arguments, this parameter is the value of **0**. (This parameter is the value of **0**, if the *ConvertArguments* parameter is the value of **NULL**.)

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **<X11/Convert.h>** header file.

The **XtConverter** data type, **XtAddressMode** enumerated type, **XtConvertArgRec** data structure.

The **XtStringConversionWarning** subroutine, **XtConvert** subroutine, **XtDirectConvert** subroutine.

XtAppAddInput Subroutine

Purpose

Registers a new file as an input source for a specified application.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtInputID XtAppAddInput(*ApplicationContext*, *Source*, *Condition*,
Procedure, *ClientData*)
XtAppContext *ApplicationContext*;
int *Source*;
caddr_t *Condition*;
XtInputCallbackProc *Procedure*;
caddr_t *ClientData*

Description

The **XtAppAddInput** subroutine registers with the Intrinsics library read routine a new source of events, which can be either file input or output. (In this instance, "file" represents any sink or source data.) The **XtAppAddInput** subroutine also specifies the conditions under which the source can generate events. When input is pending on this source, the **XtInputCallbackProc** procedure is called.

Parameters

ApplicationContext	Specifies the application context that identifies the application.
Source	Specifies the source file descriptor or another system-dependent device specification.
Condition	Specifies the mask that indicates a read, write, or exception condition or another operating system dependent condition. This parameter is some union of the **XtInputRead**, **XtInputWrite**, and **XtInputExcept** masks.
Procedure	Specifies the **XtInputCallbackProc** callback procedure.
ClientData	Specifies the argument for the specified procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRemoveInput** subroutine

The **XtInputCallbackProc** data type.

XtAppAddTimeOut Subroutine

Purpose

Creates a time-out value.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtIntervalId XtAppAddTimeOut(*ApplicationContext,*
Interval, Procedure,
ClientData)
XtAppContext *ApplicationContext*;
unsigned long *Interval*;
XtTimerCallbackProc *Procedure*;
caddr_t *ClientData*;

Description

The **XtAppAddTimeOut** subroutine creates a time-out and returns an identifier for it. The time-out is set to the *Interval* parameter. The callback procedure is called when the time interval elapses; the time-out is then removed.

Parameters

ApplicationContext	Specifies the application context for which the timer is to be set.
Interval	Specifies the time interval in milliseconds.
Procedure	Specifies the callback procedure.
ClientData	Specifies the argument to be passed to the callback procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRemoveTimeOut** subroutine.

The **XtTimerCallbackProc** type.

XtAppAddWorkProc Subroutine

Purpose

Registers a work procedure for a specified application.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtWorkProcId XtAppAddWorkProc(*ApplicationContext*, *Procedure*, *ClientData*)
XtAppContext *ApplicationContext*;
XtWorkProc *Procedure*;
caddr_t *ClientData*

Description

The **XtAppAddWorkProc** subroutine registers a work procedure for a specified application. This procedure is useful when there is limited support for background processing and applications must wait for input.

While multiple work procedures can be registered, the most recently added procedure is the procedure that is called. However, if a work procedure adds another work procedure, the current work procedure has priority over the newly added procedure.

The **XtWorkProcId** structure is an opaque identifier for this work procedure. Multiple work procedures can be registered, and the most recently added one is always the one that is called. If a work procedure adds another work procedure, however, the newly added one has lower priority than the current one.

Parameters

ApplicationContext	Specifies the application context that identifies the application.
Procedure	Specifies the work procedure to be called when the application is idle.
ClientData	Specifies the argument to be passed to the specified procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRemoveWorkProc** procedure.

XtAppCreateShell Subroutine

Purpose

Creates a top-level widget that is the root of a widget tree.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtAppCreateShell(*ApplicationName*, *ApplicationClass*, *WidgetClass*,
DisplayPtr, *Arguments*, *NumberArguments*)
String *ApplicationName*;
String *ApplicationClass*;
WidgetClass *WidgetClass*;
Display **DisplayPtr*;
ArgList *Arguments*;
Cardinal *NumberArguments*;

Description

The **XtAppCreateShell** subroutine creates a top-level widget that is the root of a widget tree. It saves the specified application name and class for qualifying all widget resource specifiers. The application name and class are used as the left-most components in all widget resource names for this application.

The **XtAppCreateShell** subroutine creates:

- A new logical application within a program by allowing the specification of a new root in the resource hierarchy, or
- A shell on another display by using the resource database associated with the other display.

This subroutine returns a widget with the WM_COMMAND property set for session managers.

Parameters

ApplicationName	Specifies the name of the application instance. If this parameter is the value of **NULL**, the application name passed to the **XtDisplayInitialize** subroutine is the application name that is used.
ApplicationClass	Specifies the application class name.
WidgetClass	Specifies the widget class for the application top-level widget. The value for this parameter is usually the **applicationShellWidgetClass** value.
DisplayPtr	Specifies the display from which to get the resources.
Arguments	Specifies the argument list for the WM_COMMAND property.
NumberArguments	Specifies the number of arguments in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCreatePopupShell** subroutine.

XtAppError Subroutine

Purpose

Calls the installed fatal error procedure.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppError(*ApplicationContext, Message***)**
XtAppContext *ApplicationContext*;
String *Message*;

Description

The **XtAppError** subroutine calls the installed fatal error procedure. To customize and internalize error messages for most application programs, use the **XtAppErrorMsg** subroutine.

Parameters

ApplicationContext	Specifies the application context.
Message	Specifies the message to be reported.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppGetErrorDatabase** subroutine, **XtAppGetErrorDatabaseText** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtAppSetWarningMsgHandler** subroutine, **XtAppWarningMsg** subroutine, **XtAppSetErrorHandler** subroutine, **XtAppSetWarningHandler** subroutine, **XtAppWarning** subroutine, **XtAppErrorMsg** subroutine.

XtAppErrorMsg Subroutine

Purpose

The high-level error handler.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppErrorMsg(*ApplicationContext, Name,*
Type, Class, Default,
*Parameters, NumberParameters***)**
XtAppContext *ApplicationContext*;
String *Name*;
String *Type*;
String *Class*;
String *Default*;
String **Parameters*;
Cardinal **NumberParameters*;

Description

The **XtAppErrorMsg** subroutine is the high-level error handler. Use the **XtAppErrorMsg** subroutine to customize and internalize error messages for most application programs.

Parameters

ApplicationContext	Specifies the application context.
Name	Specifies the general kind of error.
Type	Specifies the detailed name of the error.
Class	Specifies the resource class. For all Intrinsics internal errors, this parameter has the value of **XtToolkitError**.
Default	Specifies a default message if an error database entry is not found.
Parameters	Specifies a pointer to a list of values to be stored in the message.
NumberParameters	Specifies the number of values in the parameter list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppGetErrorDatabase** subroutine, **XtAppGetErrorDatabaseText** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtAppError** subroutine, **XtAppSetWarningMsgHandler** subroutine, **XtAppWarningMsg** subroutine, **XtAppSetErrorHandler** subroutine, **XtAppSetWarningHandler** subroutine, **XtAppWarning** subroutine.

XtAppGetErrorDatabase Subroutine

Purpose

Obtains the error database.

Library

Intrinsics Library (**libXt.a**)

Syntax

XrmDatabase *XtAppGetErrorDatabase(*ApplicationContext***)**
XtAppContext *ApplicationContext*;

Description

The **XtAppGetErrorDatabase** subroutine obtains the error database and merges it with an application or widget-specific database with the first call to the **XtAppGetErrorDatabaseText** subroutine. The **XtAppGetErrorDatabase** subroutine then returns the address of the error database.

Parameter

ApplicationContext Specifies the application context.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppSetErrorMsgHandler** subroutine, **XtAppError** subroutine, **XtAppErrorMsg** subroutine, **XtAppSetWarningMsgHandler** subroutine, **XtAppWarningMsg** subroutine, **XtAppSetErrorHandler** subroutine, **XtAppSetWarningHandler** subroutine, **XtAppWarning** subroutine.

XtAppGetErrorDatabaseText Subroutine

Purpose

Obtains the error database text for an error or warning.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppGetErrorDatabaseText(*ApplicationContext*,
Name, *Type*, *Class*,
Default,*BufferReturn*,
NumberBytes, *Database*)

XtAppContext *ApplicationContext*;
char **Name*,**Type*,**Class*;
char **Default*;
char **BufferReturn*;
int *NumberBytes*;
XrmDatabase *Database*;

Description

The **XtAppGetErrorDatabaseText** subroutine obtains the error database text for an error or warning. It returns the appropriate or default message from the error database.

Parameters

ApplicationContext	Specifies the application context.
BufferReturn	Specifies the buffer into which the error message is to be returned.
Class	Specifies the resource class of the error message.
Database	Specifies the name of the alternative database to be used. If the application database is to be used, this parameter is the value of **NULL**.
Default	Specifies the default message to use if an error database entry is not found.
Name	Specifies the name concatenated to form the resource name of the error message.
NumberBytes	Specifies the size of the buffer in bytes.
Type	Specifies the type concatenated to form the resource name of the error message.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppGetErrorDatabase** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtAppError** subroutine, **XtAppErrorMsg** subroutine, **XtAppSetWarningMsgHandler** subroutine, **XtAppWarningMsg** subroutine, **XtAppSetErrorHandler** subroutine, **XtAppSetWarningHandler** subroutine, **XtAppWarning** subroutine.

XtAppGetSelectionTimeout Subroutine

Purpose

Gets the current selection time–out value.

Library

Intrinsics Library (**libXt.a**)

C Syntax

unsigned long XtAppGetSelectionTimeout(*ApplicationContext)*
XtAppContext *ApplicationContext*;

Description

The **XtAppGetSelectionTimeout** subroutine gets the current selection timeout value in milliseconds. The **selectionTimeout** is the time within which two communicating applications must respond to one another.

The intial time–out value is set by the selection time–out application resource. If not specified, the time–out value defaults to 5 seconds.

Parameter

ApplicationContext Specifies the application context.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppSetSelectionTimeout** subroutine.

XtAppMainLoop Subroutine

Purpose

Processes input from a specified application.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppMainLoop(*ApplicationContext***)**
XtAppContext *ApplicationContext*;

Description

The **XtAppMainLoop** subroutine processes input from a specified application by doing the following:

- Calling the **XtAppNextEvent** subroutine.
- Calling the **XtDispatchEvent** subroutine, which dispatches the event to the appropriate registered procedure.

This process constitutes the main loop of X Toolkit applications, and as such, it does not return. Applications should exit in response to user action.

Parameter

ApplicationContext Specifies the application context that identifies the application.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppNextEvent** subroutine, **XtDispatchEvent** subroutine.

XtAppNextEvent Subroutine

Purpose

Returns the value from the top of a specified application input queue.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppNextEvent(*ApplicationContext*, *EventReturn*)
XtAppContext *ApplicationContext*;
XEvent **EventReturn*;

Description

The **XtAppNextEvent** subroutine returns the value from the top of a specified application input queue. If there is no input in the X input queue, the **XtAppNextEvent** subroutine flushes the X output buffer and waits for an event while looking at other input sources and time-out values and calling any callback procedures triggered by them.

During the time that your application is waiting for input, you can register an idle-time work procedure with the **XtWorkProc** procedure for background processing.

Parameters

ApplicationContext	Specifies the application context that identifies the application.
EventReturn	Returns the event information to the specified event structure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppPending** subroutine, **XtAppPeekEvent** subroutine, **XPending** subroutine, **XPeekEvent** subroutine, **XNextEvent** subroutine.

The **XtWorkProc** data type, **XtAddWorkProc** procedure.

XtAppPeekEvent Subroutine

Purpose

Returns the value from the top of an application input queue without removing input from the queue.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtAppPeekEvent(*ApplicationContext*, *EventReturn*)
XtAppContext *ApplicationContext*;
XEvent **EventReturn*;

Description

The **XtAppPeekEvent** subroutine returns the value from the top of a specified application input queue without removing the input from the queue.

If there is an event in the queue, the **XtAddPeekEvent** subroutine fills in the event and returns the value of **True**.

If no X input is on the queue, the **XtAddPeekEvent** subroutine flushes the output buffer and blocks until input is available (possibly calling time-out callbacks in the process). If the input is an event, the **XtAddPeekEvent** subroutine fills in the event and returns the value of **True**. Otherwise, the input is for an alternate input source, and the **XtAddPeekEvent** subroutine returns the value of **False**.

Parameters

ApplicationContext	Specifies the application context that identifies the application.
EventReturn	Returns the event information to the specified event structure.

Return Values

True	Indicates that there is an event in the queue.
False	Indicates that the input on the queue is for an alternate input source.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppPending** subroutine, **XtAppNextEvent** subroutine, **XPending** subroutine, **XPeekEvent** subroutine, **XNextEvent** subroutine.

XtAppPending Subroutine

Purpose

Determines whether there are events pending in the input queue for a specified application.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtInputMask XtAppPending(*ApplicationContext***)**
XtAppContext *ApplicationContext*;

Description

The **XtAppPending** subroutine determines if events for a specified application are pending in the input queue.

Parameter

ApplicationContext	Specifies the application context that identifies the application to check.

Return Values

Nonzero	Indicates that there are events pending from the X Server, timer pending, or other input sources pending. The value returned is a bit mask OR of the **XtIMXEvent**, **XtIMTimer**, and **XtIMAlternate** inputs.
0	Indicates that there are no events pending. The **XtAppPending** subroutine flushes the buffer if this is the case.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppNextEvent** subroutine, **XtAppPeekEvent** subroutine, **XtAppProcessEvent** subroutine, **XPending** subroutine, **XPeekEvent** subroutine, **XNextEvent** subroutine.

XtAppProcessEvent Subroutine

Purpose

Controls the processing for different types of input.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppProcessEvent(*ApplicationContext*, *Mask***)**
XtAppContext *ApplicationContext;*
XtInput *Mask*;

Description

The **XtAppProcessEvent** subroutine controls the processing for different types of input. This routine processes one timer, alternate input, or X event.

If there is nothing to process, the **XtAppProcessEvent** subroutine blocks until there is. It processes things in a random order.

The **XtAppProcessEvent** subroutine processes:

- Timer events by calling the appropriate timer callbacks
- Alternate input by calling the appropriate input callbacks
- X events by calling the **XtDispatchEvent** subroutine

This routine usually is not used by client applications.

Parameters

ApplicationContext — Specifies the application context that identifies the application to process.

Mask — Specifies the types of events to process. The *Mask* parameter is the bitwise-inclusive OR of any combination of the following:

XtIMXEvent

XtIMTimer

XtIMAlternateInput

XtIMAll — Can be used as the bitwise-inclusive OR of all event types.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppMainLoop** subroutine, **XtDispatchEvent** subroutine.

The **XtTimerCallbackProc** procedure, **XtInputCallbackProc** data type.

XtAppSetErrorHandler Subroutine

Purpose

Registers a procedure to call on fatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppSetErrorHandler(*ApplicationContext*, *Handler*)
XtAppContext *ApplicationContext*;
XtErrorHandler *Handler*;

Description

The **XtAppSetErrorHandler** subroutine registers a procedure to call on fatal error conditions. The default error handler provided by the Intrinsics library is the **_XtError** subroutine. On UNIX based systems, it prints the message to standard error and ends the application.

Fatal error message handlers should not return. If a fatal error message handler returns, subsequent X Toolkit behavior is not defined.

Parameters

ApplicationContext	Specifies the application context.
Handler	Specifies the new fatal error procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtError** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtErrorMsg** subroutine, **XtAppErrorMsg** subroutine, **XtAppError** subroutine, **XtAppWarningMsg** subroutine.

The **XtErrorMsgHandler** data type, **XtErrorHandler** data type.

XtAppSetErrorMsgHandler Subroutine

Purpose

Registers a procedure to call on fatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppSetErrorMsgHandler(*ApplicationContext*, *MessageHandler*)
XtAppContext *ApplicationContext*;
XtErrorMsgHandler *MessageHandler*;

Description

The **XtAppSetErrorMsgHandler** subroutine registers a procedure to call on fatal error conditions. The default error handler provided by the Intrinsics library constructs a string from the error resource database and calls the **XtError** subroutine.

Fatal error message handlers should not return. If a fatal error message handler returns, subsequent X Toolkit behavior is undefined.

Parameters

ApplicationContext	Specifies the application context.
MessageHandler	Specifies the new fatal error procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtError** subroutine, **XtAppSetErrorHandler** subroutine, **XtErrorMsg** subroutine, **XtAppErrorMsg** subroutine, **XtAppError** subroutine, **XtAppWarningMsg** subroutine.

The **XtErrorMsgHandler** data type, **XtErrorHandler** data type.

XtAppSetSelectionTimeout Subroutine

Purpose

Sets the Intrinsics library selection time-out value.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppSetSelectionTimeout(*ApplicationContext*, *Timeout*)
XtAppContext *ApplicationContext*;
unsigned long *Timeout*;

Description

The **XtAppSetSelectionTimeout** subroutine sets the selection time-out value.

Parameters

ApplicationContext	Specifies the application context.
Timeout	Specifies the selection time-out in milliseconds.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppGetSelectionTimeout** subroutine.

XtAppSetWarningHandler Subroutine

Purpose

Registers a procedure to call on nonfatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppSetWarningHandler(*ApplicationContext*, *Handler*)
XtAppContext *ApplicationContext*;
XtErrorHandler *Handler*;

Description

The **XtAppSetWarningHandler** subroutine registers a procedure to call on nonfatal error conditions. The default warning handler provided by the Intrinsics library is the **_XtWarning** subroutine. On UNIX based systems, it prints the message to standard error and returns to the caller.

Parameters

ApplicationContext	Specifies the application context.
Handler	Specifies the new nonfatal error procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppWarning** subroutine, **XtAppGetErrorDatabase** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtAppError** subroutine, **XtAppErrorMsg** subroutine, **XtAppSetWarningMsgHandler** subroutine, **XtAppWarningMsg** subroutine, **XtAppSetErrorHandler** subroutine.

The **XtErrorHandler** data type.

XtAppSetWarningMsgHandler Subroutine

Purpose

Registers a procedure to call on nonfatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppSetWarningMsgHandler(*ApplicationContext*,
MessageHandler)
XtAppContext *ApplicationContext*;
XtErrorMsgHandler *MessageHandler;*

Description

The **XtAppSetWarningMsgHandler** subroutine registers a procedure to call on nonfatal error conditions. The default warning handler provided by the Intrinsics constructs a string from the error resource database; it then calls the **XtWarning** subroutine and returns to the caller.

Parameters

ApplicationContext	Specifies the application context.
MessageHandler	Specifies the new nonfatal error procedure, which usually returns.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppWarning** subroutine, **XtAppGetErrorDatabase** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtAppError** subroutine, **XtAppErrorMsg** subroutine, **XtAppSetWarningHandler** subroutine, **XtAppWarningMsg** subroutine, **XtAppSetErrorHandler** subroutine.

The **XtErrorMsgHandler** data type.

XtAppWarning Subroutine

Purpose

This is the nonfatal error procedure.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppWarning(*ApplicationContext*, *Message*)
XtAppContext *ApplicationContext*;
String *Message*;

Description

The **XtAppWarning** subroutine is the installed nonfatal error procedure. Most programs should use the **XtAppWarningMsg** subroutine, not the **XAppWarning** subroutine, to customize and internalize warning messages.

Parameters

ApplicationContext	Specifies the application context.
Message	Specifies the nonfatal error message to be reported.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppGetErrorDatabase** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtAppError** subroutine, **XtAppErrorMsg** subroutine, **XtAppSetWarningMsgHandler** subroutine, **XtAppWarningMsg** subroutine, **XtAppSetErrorHandler** subroutine, **XtAppSetWarningHandler** subroutine.

XtAppWarningMsg Subroutine

Purpose

This is the high-level warning handler.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAppWarningMsg(*ApplicationContext, Name,*
Type, Class, Default,
*Parameters, NumberParameters***)**
XtAppContext *ApplicationContext*;
String *Name*;
String *Type*;
String *Class*;
String *Default*;
String **Parameters*;
Cardinal **NumberParameters*;

Description

The **XtAppWarningMsg** subroutine is the installed high-level warning handler. Use this routine to customize and internalize warning messages.

Parameters

ApplicationContext	Specifies the application context.
Name	Specifies the general kind of error.
Type	Specifies the detailed name of the error.
Class	Specifies the resource class. For all Intrinsics library internal warnings, this parameter is set to the value of **XtToolkitError**.
Default	Specifies a default message to be used if there is no entry in the error database.
Parameters	Specifies a pointer to a list of values stored in the message.
NumberParameters	Specifies the number of values in the parameter list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtError** subroutine, **XtAppSetErrorMsgHandler** subroutine, **XtErrorMsg** subroutine, **XtAppErrorMsg** subroutine, **XtAppError** subroutine.

The **XtErrorMsgHandler** procedure, **XtErrorHandler** procedure.

XtAugmentTranslations Subroutine

Purpose

Merges new translations into an existing widget translation table.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtAugmentTranslations(*WidgetID*, *Translations***)**
Widget *WidgetID;*
XtTranslations *Translations;*

Description

The **XtAugmentTranslations** subroutine merges new translations into an existing widget translation table. The new translation is ignored if it contains an event or event sequence that is already in the widget translation table.

Parameters

WidgetID	Specifies the widget into which the new translations are to be merged.
Translations	Specifies the compiled translation table in which to merge the new translations. This parameter cannot be the value of **NULL**.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtParseTranslationTable** subroutine, **XtOverrideTranslations** subroutine, **XtUninstallTranslations** subroutine.

The **XtTranslations** structure.

XtBuildEventMask Subroutine

Purpose

Retrieves the event mask for a specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

EventMask XtBuildEventMask(*WidgetID***)**
Widget *WidgetID;*

Description

The **XtBuildEventMask** subroutine retrieves the event mask for a specified widget. This subroutine returns the event mask representing the logical OR of all event masks for event handlers registered on the widget with the **XtAddEventHandler** subroutine. All event translations, including accelerators, installed on the widget are returned also.

This event mask is the same as the one stored into the **XSetWindowAttributes** structure by the **XtRealizeWidget** subroutine and sent to the server when event handlers and translations are installed or removed on the realized widget.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtEventHandler** data type.

The **XtAddEventHandler** subroutine, **XtRemoveEventHandler** subroutine, **XtAddRawEventHandler** subroutine, **XtRemoveRawEventHandler** subroutine, **XtRealizeWidget** subroutine.

The **XSetWindowAttributes** structure.

The **XtAllEvents** event mask.

XtCallAcceptFocus Subroutine

Purpose

Calls the **accept_focus** procedure of a specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtCallAcceptFocus(*WidgetID, TimeStamp***)**
Widget *WidgetID;*
Time **TimeStamp;*

Description

The **XtCallAcceptFocus** subroutine calls the procedure specified in the *accept_focus* field of a widget. This subroutine passes the specified widget and time to the procedure and returns (to the user) the information the procedure returns.

Parameters

WidgetID	Specifies the widget.
TimeStamp	Specifies the X time of the event causing the **accept_focus** procedure.

Return Value

False	If the *accept_focus* field for the specified widget is the value of **NULL**. Otherwise, this subroutine returns what the **accept_focus** procedure returns.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtSetKeyboardFocus** subroutine.

The **XtAcceptFocusProc** type.

XtCallCallbacks Subroutine

Purpose

Executes the callback procedures in a specified widget callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCallCallbacks(*WidgetID*, *CallbackName*, *CallData***)**
Widget *WidgetID*;
String *CallbackName*;
caddr_t *CallData*;

Description

The **XtCallCallbacks** subroutine runs the callback procedure in a widget callback list.

Parameters

WidgetID	Specifies the widget.
CallbackName	Specifies the callback list to be ran.
CallData	Specifies the **callback_list** specific data value for the specified callback procedures. This parameter has the value of the actual data if only one 32-bit long word is needed or the value of the address of the data if more than one word is needed. If no data is required, this parameter is the value of **NULL**.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCallbackList** data type, **XtCallbackProc** type, **XtCallbackRec** type.

The **XtAddCallback** subroutine, **XtAddCallbacks** subroutine, **XtHasCallbacks** subroutine, **XtRemoveCallback** subroutine, **XtRemoveCallbacks** subroutine, **XtRemoveAllCallbacks** subroutine.

XtCallbackExclusive Subroutine

Purpose

Maps a pop-up widget from a specified widget callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCallbackExclusive(*WidgetID*, *ClientData*, *CallData***)**
Widget *WidgetID;*
caddr_t *ClientData;*
caddr_t *CallData;*

Description

The **XtCallbackExclusive** subroutine maps a pop-up widget from a specified widget callback list in the following manner:

1. Calls the **XtPopup** subroutine with the shell specified by the *ClientData* parameter and the *grab_kind* field of the shell set to the **XtGrabExclusive** value.
2. Sets the widget to be insensitive by calling the **XtSetSensitive** subroutine.

The **XtCallbackExclusive** subroutine is not required in callbacks. In particular, an application must provide customized code for callbacks that create pop-up shells dynamically or callbacks that must do more than desensitize the button.

Parameters

WidgetID	Specifies the widget running the callback.
ClientData	Specifies the shell to be popped up.
CallData	Specifies the callback data. This parameter is not used by this subroutine.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCallbackNone** subroutine, **XtCallbackNonexclusive** subroutine, **XtCallbackPopdown** subroutine, **XtPopup** subroutine.

XtCallbackNone Subroutine

Purpose

Maps a pop-up widget menu from a specified widget callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCallbackNone(*WidgetID*, *ClientData*, *CallData***)**
Widget *WidgetID;*
caddr_t *ClientData;*
caddr_t *CallData;*

Description

The **XtCallbackNone** subroutine maps a pop-up widget menu from a specified widget callback list in the following manner:

1. Calls the **XtPopup** subroutine with the shell specified by the *ClientData* parameter and the *grab_kind* field of the shell set to the **XtGrabNone** value.

2. Sets the widget to be insensitive by calling the **XtSetSensitive** subroutine.

The **XtCallbackNone** subroutine is not required in callbacks. In particular, an application must provide customized code for callbacks that create pop-up shells dynamically or callbacks that must do more than desensitize the button.

Parameters

WidgetID	Specifies the widget running the callback.
ClientData	Specifies the shell to be popped up.
CallData	Specifies the callback data. This parameter is not used by this subroutine.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCallbackExclusive** subroutine, **XtCallbackNonexclusive** subroutine, **XtCallbackPopdown** subroutine, **XtPopup** subroutine.

XtCallbackNonexclusive Subroutine

Purpose

Maps a pop-up widget menu from a specified widget callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCallbackNonexclusive(*WidgetID*, *ClientData*, *CallData***)**
Widget *WidgetID*;
caddr_t *ClientData*;
caddr_t *CallData*;

Description

The **XtCallbackNonexclusive** subroutine maps a pop-up widget from a specified widget callback list in the following manner:

1. Calls the **XtPopup** subroutine with the shell specified by the *ClientData* parameter and the *grab_kind* field of the shell set to the **XtGrabNonexclusive** value.

2. Sets the widget to be insensitive by calling the **XtSetSensitive** subroutine.

The **XtCallbackNonexclusive** subroutine is not required in callbacks. In particular, an application must provide customized code for callbacks that create pop-up shells dynamically or callbacks that must do more than desensitize the button.

Parameters

WidgetID	Specifies the widget running the callback.
ClientData	Specifies the shell to be popped up.
CallData	Specifies the callback data. This parameter is not used by this subroutine.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCallbackNone** subroutine, **XtCallbackExclusive** subroutine, **XtCallbackPopdown** subroutine, **XtPopup** subroutine.

XtCallbackPopdown Subroutine

Purpose

Pops down a shell that has been popped up with the **XtCallbackNone,** the **XtCallbackNonexclusive,** or the **XtCallbackExclusive** subroutine.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCallbackPopdown(*WidgetID*, *ClientData*, *CallData***)**
Widget *WidgetID*;
caddr_t *ClientData*;
caddr_t *CallData*;

Description

The **XtCallbackPopdown** subroutine pops down a shell mapped by the **XtCallbackNone**, **XtCallbackNonexclusive**, or **XtCallbackExclusive** subroutine. The **XtCallbackPopdown** subroutine casts the *ClientData* parameter to an **XtPopdownID** pointer. It calls the **XtPopdown** subroutine with the specified shell widget in the **PopdownID** data structure; then it calls the **XtSetSensitive** subroutine to resensitize the enable widget of the **XtPopdownID** data structure.

Parameters

WidgetID	Specifies the widget.
ClientData	Specifies a pointer to the **XtPopdownID** data structure.
CallData	Specifies the callback data. This parameter is not used by this subroutine.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtPopdown** subroutine, **XtSetSensitive** subroutine, **XtCallbackNone** subroutine, **XtCallbackNonexclusive** subroutine, **XtCallbackExclusive** subroutine.

The **XtPopdownID** data structure.

XtCalloc Subroutine

Purpose

Allocates and initializes an array.

Library

Intrinsics Library (**libXt.a**)

Syntax

char *XtCalloc(*Number*, *Size***)**
Cardinal *Number*;
Cardinal *Size*;

Description

The **XtCalloc** subroutine allocates and initializes an array. It allocates space for the specified number of array elements of the specified size and initializes the space to a value of 0. If there is insufficient memory to allocate the new block, the **XtCalloc** subroutine calls the **XtErrorMsg** subroutine.

Parameters

Number Specifies the number of array elements to allocate.

Size Specifies the size of an array element in bytes.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtNumber** macro.

The **XtMalloc** subroutine, **XtRealloc** subroutine, **XtFree** subroutine.

XtCheckSubclass Macro

Purpose

Checks the subclass of a widget and generates a debugging error message.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCheckSubclass(*WidgetID*, *WidgetClass*, *Message*)
Widget *WidgetID*;
WidgetClass *WidgetClass*;
String *Message*

Description

The **XtCheckSubclass** macro checks the subclass of a widget and generates a debugging error message. This subroutine determines if the class of the specified widget is equal to or a subclass of the value in the *WidgetClass* parameter. The widget can be any number of subclasses down the chain. Use the **XtCheckSubclass** subroutine at the entry point of the exported subroutines to ensure that a valid widget class for the exported operations is passed.

The **XtCheckSubclass** macro is executed only when the widget has been compiled with the **DEBUG** compiler symbol defined. Otherwise, the **XtCheckSubclass** macro is defined as an empty string and does not generate code.

If the widget is not a subclass of the value in the *WidgetClass* parameter, the **XtCheckSubclass** macro constructs an error message from the supplied message, the actual class of the widget, and the expected class of the widget; then it calls the **XtErrorMsg** subroutine.

Parameters

WidgetID	Specifies the widget to test.
WidgetClass	Specifies the widget class to test against.
Message	Specifies the error message to be used.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtClass** subroutine, **XtSuperclass** subroutine, **XtIsSubclass** subroutine.

XtClass Macro

Purpose

Obtains the class of a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

WidgetClass XtClass(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtClass** macro obtains the class of a widget and returns a pointer to the widget class structure.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtSuperclass** macro, **XtCheckSubclass** macro.

XtIsSubclass subroutine.

The **WidgetClass** structure.

XtCloseDisplay Subroutine

Purpose

Closes a specified display and removes it from an application context.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCloseDisplay(*DisplayPtr***)**
Display **DisplayPtr*;

Description

The **XtCloseDisplay** subroutine closes a specified display and removes it from an application context as soon as it is possible to do so. Use the **XtCloseDisplay** subroutine only when an application will be executing after the display is closed. Otherwise, use the **XtDestroyApplicationContext** subroutine or exit appropriately.

Parameter

DisplayPtr Specifies the display.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtOpenDisplay** subroutine, **XtDisplayInitialize** subroutine, **XtCreateApplicationContext** subroutine.

XtConfigureWidget Subroutine

Purpose

Moves and resizes the sibling widget of the child making the geometry request.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtConfigureWidget(*WidgetID*, *X* , *Y*,
Height, *Width*, *BorderWidth*)
Widget *WidgetID*;
Position *X*;
Position *Y*;
Dimension *Width*;
Dimension *Height*;
Dimension *BorderWidth*;

Description

The **XtConfigureWidget** subroutine moves and resizes the sibling widget of the child making the geometry request. It returns immediately if the specified geometry fields are the same value as the old geometry fields. Otherwise, the **XtConfigureWidget** subroutine writes the new *X*, *Y*, *Width*, *Height*, and *BorderWidth* parameters into the widget.

If the widget is realized, the **XtConfigureWidget** subroutine calls the **XConfigureWindow** subroutine on the window of the widget.

If the new value for either the *Width* or the *Height* parameter is different from the previous values, the **XtConfigureWidget** subroutine calls the resize procedure of the widget to notify the widget of the size change.

Parameters

Height	Specifies the new height for the widget.
WidgetID	Specifies the widget.
X	Specifies the new x coordinate for the widget.
Width	Specifies the new width for the widget.
BorderWidth	Specifies the new widget size.
Y	Specifies the new y coordinate for the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtMoveWidget** subroutine, **XtResizeWidget** subroutine, **XtQueryGeometry** subroutine, **XConfigureWindow** subroutine.

XtConvert Subroutine

Purpose

Calls resource converters.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtConvert(*WidgetID*, *FromType*, *From*,
ToType, *ToReturn*)
Widget *WidgetID*;
String *FromType*;
XrmValuePtr *From*;
String *ToType*;
XrmValuePtr *ToReturn*;

Description

The **XtConvert** subroutine invokes resource converters. Conversion subroutines are used if a resource or a widget default value in an application are different than what the user requires. The **XtConvert** subroutine looks up the type converter registered to convert FromType to ToType and computes any additional arguments needed; then it calls the **XtDirectConvert** subroutine.

Parameters

WidgetID	Specifies the widget to use for additional arguments.
FromType	Specifies the source type.
From	Specifies the value to be converted.
ToType	Specifies the destination type.
ToReturn	Returns the converted value.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetSubresources** subroutine, **XtGetApplicationResources** subroutine, **XtDirectConvert** subroutine.

XtConvertCase Subroutine

Purpose

Determines the uppercase and lowercase equivalents for a key symbol.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtConvertCase(*DisplayPtr*, *KeySym*, *LowerReturn*,
UpperReturn)
Display **DisplayPtr*;
KeySym *KeySym*;
KeySym **LowerReturn*;
KeySym **UpperReturn*;

Description

The **XtConvertCase** subroutine determines the uppercase and lowercase equivalents for a key symbol. It calls the appropriate converter and returns the results. This subroutine can be used with a user-supplied **XtKeyProc** procedure.

Parameters

DisplayPtr	Specifies the display that provided the key symbol.
KeySymbol	Specifies the key symbol to convert.
LowerReturn	Returns the lowercase equivalent of the key symbol.
UpperReturn	Returns the uppercase equivalent of the key symbol.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtKeyProc** procedure, **XtCaseProc** procedure.

The **XtRegisterCaseConverter** subroutine, **XtTranslateKeycode** subroutine, **XtSetKeyTranslator** subroutine.

The **KeySym** structure.

The **XtCaseProc** data type.

XtCreateApplicationContext Subroutine

Purpose

Creates an application context.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtAppContext XtCreateApplicationContext();

Description

The **XtCreateApplicationContext** subroutine creates an application context which is an opaque type. Each application must have at least one application context.

Return Value

ApplicationContext

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtDestroyApplicationContext** subroutine, **XtWidgetToApplicationContext** subroutine, **XtToolkitInitialize** subroutine, **XtDisplayInitialize** subroutine, **XtOpenDisplay** subroutine.

XtCreateApplicationShell Subroutine

Purpose

Creates an application shell widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtCreateApplicationShell(*Name*, *Class*, *Arguments*,
NumberArguments)
String *Name;*
WidgetClass *Class;*
ArgList *Arguments;*
Cardinal *NumberArguments;*

Description

The **XtCreateApplicationShell** subroutine creates an application shell widget by calling the **XtAppCreateShell** subroutine with the following:

- An *ApplicationName* parameter that is the value of **NULL**.
- The *ApplicationClass* parameter passed to the **XtInitialize** subroutine
- The default application context created by the **XtInitialize** subroutine.

Note: The **XtCreateApplicationShell** subroutine exists only as a convenience to users converting from earlier versions of the toolkit. It has been replaced by the **XtAppCreateShell** subroutine.

Parameters

Name	This parameter is ignored. The value of **NULL** can be specified.
Class	Specifies the widget class pointer for the application shell widget. Usually the value for this parameter is the **topLevelShellWidgetClass** class or a subclass thereof.
Arguments	Specifies the argument list to override the resource defaults.
NumberArguments	Specifies the number of arguments.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtInitialize** subroutine, **XtAppCreateShell** subroutine.

XtCreateManagedWidget Subroutine

Purpose

Creates and manages a child widget in a single procedure.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtCreateManagedWidget(*Name*, *Class*,
Parent, *Arguments*,
NumberArguments)

String *Name;*
WidgetClass *Class;*
Widget *Parent;*
ArgList *Arguments;*
Cardinal *NumberArguments;*

Description

The **XtCreateManagedWidget** subroutine creates and manages a child widget in a single procedure. It calls the **XtCreateWidget** and **XtManageChild** subroutines.

Parameters

Name	Specifies the text name for the created widget.
Class	Specifies the widget class pointer for the created widget.
Parent	Specifies the parent widget.
Arguments	Specifies the argument list to override the resource defaults.
NumberArguments	Specifies the number of arguments in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtManageChild** subroutine, **XtUnmanageChildren** subroutine, **XtUnmanageChild** subroutine.

The **XtIsManaged** macro.

The **XtManageChildren** procedure.

XtCreatePopupShell Subroutine

Purpose

Creates a pop-up shell.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtCreatePopupShell(*Name*, *Class*, *Parent*,
Arguments, *NumberArguments*)
String *Name;*
WidgetClass *Class;*
Widget *Parent;*
ArgList *Arguments;*
Cardinal *NumberArguments;*

Description

The **XtCreatePopupShell** subroutine creates a pop-up shell. It ensures that the specified class is a subclass of the **Shell** widget. The **XtCreatePopupShell** subroutine attaches the shell directly to the pop-up list of the parent shells instead of attaching the widget to the children list of the parent.

A spring-loaded pop-up shell called from a translation table must exist at the time that the translation is invoked in order for the translation manager can find the shell by name.

Parameters

Name	Specifies the text name for the created shell widget.
Class	Specifies the widget class pointer for the created shell widget.
Parent	Specifies the parent widget.
Arguments	Specifies the argument list to override the resource defaults.
NumberArguments	Specifies the number of arguments in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppCreateShell** subroutine, **XtPopup** subroutine.

XtCreateWidget Subroutine

Purpose

Creates an instance of a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtCreateWidget(*Name*, *Class*, *Parent*,
Arguments, *NumberArguments*)
String *Name;*
WidgetClass *Class;*
Widget *Parent;*
ArgList *Arguments;*
Cardinal *NumberArguments;*

Description

The **XtCreateWidget** subroutine creates an instance of a widget.

The **XtCreateWidget** subroutine performs many of the boilerplate operations of creating a widget, such as the following:

- Checks if the *class_initialize* procedure has been called for this class and for all superclasses. If not, it calls those necessary in a superclass-to-subclass order.
- Allocates memory for the widget instance.
- If the parent widget is a subclass of the **constraintWidgetClass**, allocates memory for the constraints of the parent widget and stores the address of this memory into the constraints field.
- Initializes the core non-resource data fields, such as the parent field and the visible field.
- Initializes the resource fields, such as the *background_pixel* field, by using the resource lists specified for this class and all superclasses.
- If the parent widget is a subclass of the **constraintWidgetClass**, initializes the resource fields of the constraints record by using the constraint resource list specified for the class of the parent and all superclasses up to the **constraintWidgetClass**.
- Calls the initialize procedures for the widget, starting at the core initialize procedure and moving down to the widget initialize procedure.
- If the parent widget is a subclass of the **compositeWidgetClass**, it puts the widget into the children list of its parent by calling the **InsertChild** procedure.
- If the parent widget is a subclass of the **constraintWidgetClass**, it calls the constraint initialize procedures for each widget class, starting with the **constraintWidgetClass** and moving down to the parent widget's constraint_initialize procedure.

The number of arguments in an argument list can be computed automatically with the **XtNumber** macro.

Parameters

Name	Specifies the resource name for the created widget. This name is used for retrieving resources. This name should not be the same as the name for another widget that is a child of the same parent widget.
Class	Specifies the widget class pointer for the created widget.
Parent	Specifies the parent widget.
Arguments	Specifies the argument list to override the resource defaults.
NumberArguments	Specifies the number of arguments in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCreateManagedWidget** subroutine.

XtCreateWindow Subroutine

Purpose

Creates a window with values from the widget structure and the passed parameters.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtCreateWindow(*WidgetID*, *WindowClass*, *VisualPtr*,
ValueMask, *Attributes*)
Widget *WidgetID;*
unsigned int *WindowClass;*
Visual **VisualPtr;*
XtValueMask *ValueMask;*
XSetWindowAttributes **Attributes;*

Description

The **XtCreateWindow** subroutine calls the **XCreateWindow** subroutine with the values from the widget structure and parameters passed to **XtCreateWindow** subroutine. It then assigns the created window into the window field of the widget.

This subroutine evaluates the following fields of the core widget structure: depth, screen, parent -> core.window, x, y, width, height, border_width.

Parameters

WidgetID	Specifies the widget used to create the window.
WindowClass	Specifies the **Xlib** library window class. The *WindowClass* parameter can be the following: **CopyFromParent** **InputOnly** **InputOutput**
VisualPtr	Specifies the visual type, which is usually the **CopyFromParent** value.
ValueMask	Specifies which attribute fields to use.
Attributes	Specifies the window attributes for the **XCreateWindow** call.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XCreateWindow** subroutine.

XtDatabase Subroutine

Purpose

Obtains the resource database for a particular display.

Library

Intrinsics Library (**libXt.a**)

Syntax

XrmDatabase XtDatabase(*DisplayPtr***)**
Display ***DisplayPtr;***

Description

The **XtDatabase** subroutine obtains the resource database for a particular display. This subroutine returns the fully merged resource database that was built by the **XtDisplayInitialize** subroutine associated with the display that was passed in. If this display has not been initialized by the **XtDisplayInitialize** subroutine, the results are not defined.

Parameter

DisplayPtr Specifies the display.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtDisplayInitialize** subroutine.

XtDestroyApplicationContext Subroutine

Purpose

Destroys an application context.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtDestroyApplicationContext(*ApplicationContext***)**
XtAppContext *ApplicationContext*;

Description

The **XtDestroyApplicationContext** subroutine destroys an application context as soon as it is safe to do so. If this subroutine is called from within an event dispatch, such as a callback procedure, it does not destroy the application context until the dispatch is completed.

Parameter

ApplicationContext Specifies the application context.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCreateApplicationContext** subroutine, **XtWidgetToApplicationContext** subroutine, **XtDisplayInitialize** subroutine, **XtOpenDisplay** subroutine.

XtDestroyGC Subroutine

Purpose

Deallocates a graphics context.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtDestroyGC(*WidgetID*, *GraphicsContext***)**
Widget *WidgetID;*
GC *GraphicsContext;*

Description

The **XtDestroyGC** subroutine deallocates a shared graphics context when it is no longer needed. It counts references to shareable graphics contexts and generates a free request to the server when the last user of a specified graphics context destroys it.

Note: Earlier versions of the X Toolkit did not require a widget argument for the **XtDestroyGC** subroutine. Therefore, this subroutine is not very portable; use the **XtReleaseGC** subroutine instead.

Parameters

WidgetID	Specifies the widget.
GraphicsContext	Specifies the graphics context to be deallocated.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtReleaseGC** subroutine.

XtDestroyWidget Subroutine

Purpose

Destroys a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtDestroyWidget(*WidgetID***)**
Widget *WidgetID;*

Description

The **XtDestroyWidget** subroutine destroys a widget instance. This is the only method for destroying widgets, including widgets that need to destroy themselves. It is used at any time, even from an application callback procedure of the widget being destroyed. The destroy process consists of two phases to avoid dangling references to destroyed widgets.

In Phase 1, the **XtDestroyWidget** subroutine performs the following tasks:

- Returns immediately, if the *being_destroyed* field of the widget is set to the value of **True**.
- Recursively descends the widget tree and sets the *being_destroyed* field to the value of **True** for the widget and all children.
- Adds the widget to a list of widgets (the destroy list) that should be destroyed when it is safe to do so.

Entries on the destroy list satisfy the invariant that if `widget2` occurs after `widget1` on the destroy list then `widget2` is not a descendant of `widget1`. (The term `descendant` in this case refers to both normal and pop-up children.)

Phase 2 occurs when all procedures that should run as a result of the current event have been called, including all procedures registered with the event and translation managers. Phase 2 starts immediately when not in the **XtDispatchEvent** subroutine or when the current invocation of the **XtDispatchEvent** subroutine is about to return.

In phase 2, the **XtDestroyWidget** subroutine performs the following actions on each entry in the destroy list:

- Calls the destroy callbacks registered on the widget and all descendants in post-order, calling children callbacks before parent callbacks.
- If the parent widget is a subclass of the **compositeWidgetClass** and is not being destroyed, it calls the **XtUnmanageChild** subroutine on the widget, then calls the *delete_child* procedure of the widget parent.
- If the parent of the widget is a subclass of **constraintWidgetClass,** it calls the constraint destroy procedure for the parent widget , then for the superclass of the parent upward, until the constraint destroy procedure for the **constraintWidgetClass** has been called.
- Calls the destroy class procedures for the widget and all descendants in post-order. For each such widget, it calls the destroy procedure declared in the widget class, then the destroy procedure declared in its superclass upward, until the destroy procedure declared in the Core class record has been called.

- Calls the **XDestroyWindow** subroutine if the widget is realized (it has a window). The server recursively destroys all descendant windows.
- Recursively descends the tree and deallocates all pop-up widgets, constraint records, callback lists, and, if the widget is a subclass of the **compositeWidgetClass**, all children widgets.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XDestroyWindow** subroutine.

XtDirectConvert Subroutine

Purpose

Invokes resource converters.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtDirectConvert(*Converter*, *Arguments*, *NumberArguments*, *From*, *ToReturn*)
XtConverter *Converter*;
XrmValuePtr *Arguments*;
Cardinal *NumberArguments*;
XrmValuePtr *From*;
XrmValuePtr *ToReturn*;

Description

The **XtDirectConvert** subroutine invokes resource converters. It looks in the converter cache to see if this conversion procedure has been called with the specified arguments. If this conversion procedure has been called, it returns a descriptor for information stored in the cache.

Before calling the specified converter, the **XtDirectConvert** subroutine sets the return value to 0 and the return value address to the value of **NULL**. Then it calls the converter and enters the results in the cache.

If the address in the *ToReturn* parameter contains a value other than the value of **NULL**, it indicates that the conversion was successful.

The **XtDirectConvert** subroutine is usually called after the **XtConvert** subroutine.

Parameters

Converter	Specifies the conversion procedure to be called.
Arguments	Specifies the argument list for the conversion. This parameter can be the value of **NULL**.
NumberArguments	Specifies the number of additional arguments. This parameter can be the value of **0**.
From	Specifies the value to be converted.
ToReturn	Returns the converted value.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddConverter** subroutine.

The **XtConverter** data type.

XtDisownSelection Subroutine

Purpose

Informs the selection mechanism that the specified widget is to lose ownership of the selection.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtDisownSelection(*WidgetID*, *Selection*, *TimeStamp***)**
Widget *WidgetID;*
Atom *Selection;*
Time *TimeStamp;*

Description

The **XtDisownSelection** subroutine informs the Intrinsics selection mechanism that the specified widget is to lose ownership of the selection.

After the **XtDisownSelection** subroutine is called, the widget convert procedure (the **XtConvertSelectionProc** type) cannot be called. However, the widget done procedure (the **XtSelectionDoneProc** type) can be called if a conversion procedure which started before the call to the **XtDisownSelection** subroutine finishes after the call to this subroutine.

The **XtDisownSelection** subroutine takes no action if the specified widget does not currently own the selection.

Parameters

WidgetID	Specifies the widget that will lose ownership.
Selection	Specifies the atom that identifies the selection to be lost.
TimeStamp	Specifies the time stamp that indicates when the selection ownership is relinquished.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtOwnSelection** subroutine, **XtDisownSelection** subroutine.

The **XtConvertSelectionProc** data type, **XtSelectionDoneProc** data type.

XtDispatchEvent Subroutine

Purpose

Dispatches events through event handlers.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtDispatchEvent(*Event***)**
XEvent ***Event**;

Description

The **XtDispatchEvent** subroutine receives X events. Then it calls the appropriate event handlers and passes the widget, the event, and client-specific data registered with each procedure. It sends those events to the event handler subroutines that have been registered previously with the dispatch routine.

The **XtDispatchEvent** subroutine dispatches both events acquired with the **XtAppNextEvent** subroutine and those constructed by the user.

If there are no event handlers for the registered events, the event is ignored and the dispatcher returns the value of **False**.

The **XtDispatchEvent** subroutine also implements the grab semantics for the **XtAddGrab** subroutine.

Parameter

Event	Specifies a pointer to the event structure to be dispatched to the appropriate event handler.

Return Values

True	Indicates that the event was dispatched to an event handler.
False	Indicates that no event handler was found for the event.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddEventHandler** subroutine, **XtAppProcessEvent** subroutine, **XtNextEvent** subroutine.

The **XEvent** data structure.

XtDisplay Macro

Purpose

Returns the display pointer for the specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Display *XtDisplay(*WidgetID***)**
Widget *WidgetID;*

Description

The **XtDisplay** macro returns the display pointer for the specified widget.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtDisplayInitialize Subroutine

Purpose

Initializes a display and adds it to an application context.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtDisplayInitialize(*ApplicationContext*, *DisplayPtr*, *ApplicationName*,
ApplicationClass, **Options**, *NumberOptions*, *argc*, *argv*)
XtAppContext *ApplicationContext*;
Display **DisplayPtr;*
String *ApplicationName;*
String *ApplicationClass;*
XrmOptionDescRec **Options;*
Cardinal *NumberOptions;*
Cardinal **argc;*
String **argv;*

Description

The **XtDisplayInitialize** subroutine initializes a display and adds it to an application context. It builds the resource database, calls the **XrmParseCommand** subroutine to parse the command line, and performs other individual display initialization.

After the **XrmParseCommand** subroutine has been called, the *argc* parameter and the *argv* parameter contain only those parameters that were not in the standard option table or in the table specified by the *Options* parameter.

If the modified *argc* parameter does not have the **0** value, most applications print out the modified *argv* parameter and a message listing the options allowed. The application name is typically the final component of the *argv*[0] parameter.

The **XtDisplayInitialize** subroutine has a table of standard command line options that is passed to the **XrmParseCommand** subroutine for adding resources to the resource database. The **XtDisplayInitialize** subroutine also takes additional application-specific resource abbreviations in the *Options* parameter. The format of this table is as follows:

```
typedef enum {

   XrmoptionNoArg,           /* Value specified in
                                OptionDescRec.value */
   XrmoptionIsArg,           /* Value is the option string */
   XrmoptionStickyArg,       /* Value is characters following
                                option */
   XrmoptionSepArg,          /* Value is next argument in the argv
                                parameter */
   XrmoptionSkipArg,         /* Ignore this option and the next
                                argument in the argv parameter */
   XrmoptionSkipLine         /* Ignore this option and the rest of
                                the argv parameter */

} XrmOptionKind;
```

```
typedef struct {
   char *Option;              /* Option name in the argv parameter */
   char *Specifier;           /* Resource name (without application
                                 name) */
   XrmOptionKind argKind;     /* Style of option */
   caddr_t Value;             /* Value to provide if the argKind
                                 parameter is the value of
                                 XrmoptionNoArg value */

} XrmOptionDescRec, *XrmOptionDescList;
```

The standard **XtDisplayInitialize** command line options are as following:

Option String	Resource	Argument	Resource Value
-background	background	SepArg	next argument
-bd	borderColor	SepArg	next argument
-bg	background	SepArg	next argument
-borderwidth	borderWidth	SepArg	next argument
-bordercolor	borderColor	SepArg	next argument
-bw	borderWidth	SepArg	next argument
-display	display	SepArg	next argument
-fg	foreground	SepArg	next argument
-fn	font	SepArg	next argument
-font	font	SepArg	next argument
-foreground	foreground	SepArg	next argument
-geometry	geometry	SepArg	next argument
-iconic	iconic	NoArg	true
-name	name	SepArg	next argument
-reverse	reverseVideo	NoArg	on
-rv	reverseVideo	NoArg	on
+rv	reverseVideo	NoArg	off
-selectionTimeout	selectionTimeout	SepArg	next argument
-synchronous	synchronize	NoArg	on
+synchronous	synchronize	NoArg	off
-title	title	SepArg	next argument
-xrm	next argument	ResArg	next argument

Notes:

1. Any unique abbreviation for an option name in the standard table or in the application table is accepted.
2. If the **XtOpenDisplay** subroutine is called before the **XtDisplayInitialize** subroutine, any *-display* or *-name* parameter specified with the **XtOpenDisplay** subroutine overrides the same parameters specified for the **XtDisplayInitialize** subroutine.
3. If the reverseVideo resource is the value of **True**, the Intrinsics library exchanges the values of **XtDefaultForeground** and **XtDefaultBackground** for widgets created on this display.
4. If the synchronize resource for the specified application is the value of **True**, the **XtDisplayInitialize** subroutine calls the **XSynchronize** subroutine to put the **Xlib** lilbrary into synchronous mode for this display connection.
5. The *-xrm* option provides a method of setting any resource in an application. The next argument should be a quoted string identical in format to a line in the user resources file.

For example, to give a red background to all command buttons in an application named **xmh**, enter the following:

```
xmh -xrm 'xmh*Command.background: red'
```

6. When the **XtDisplayInitialize** subroutine fully parses the command line, it merges the application option table with the standard option table before calling the **XrmParseCommand** subroutine.
7. If an entry in the application table is the same as an entry in the standard table, it overrides the standard table entry. If an option name is a prefix of another option name, both names are kept in the merged table.

Parameters

ApplicationContext	Specifies the application context.
DisplayID	Specifies the display. Each display can be in one application context only.
ApplicationName	Specifies the name of the application instance.
ApplicationClass	Specifies the class name of this application, which is usually the generic name for all instances of this application.
Options	Specifies how to parse the command line for any application-specific resources. The *Options* parameter is passed as a parameter to the **XrmParseCommand** subroutine.
NumberOptions	Specifies the number of entries in the options list.
argc	Specifies a pointer to the number of command line parameters.
argv	Specifies the command line parameters.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtToolkitInitialize** subroutine, **XtCreateApplicationContext** subroutine, **XtOpenDisplay** subroutine, **XtAppCreateShell** subroutine, **XrmParseCommand** subroutine.

XtError Subroutine

Purpose

Calls the installed fatal error procedure.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtError(*Message*)
String *Message*;

Description

The **XtError** subroutine calls the installed fatal error procedure. To customize and internalize error messages, use the **XtErrorMsg** subroutine.

Note: The **XtError** subroutine only exists as a convenience to users converting from earlier versions of the toolkit. Most programs should use the **XtErrorMsg** subroutine instead.

Parameter

Message Specifies the error message to be reported.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetErrorDatabase** subroutine, **XtGetErrorDatabaseText** subroutine, **XtSetErrorMsgHandler** subroutine, **XtErrorMsg** subroutine.

XtErrorMsg Subroutine

Purpose

Displays an error message.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtErrorMsg(*Name*, *Type*, *Class*, *Default*,
Parameters, *NumberParameters*)
String *Name;*
String *Type;*
String *Class;*
String *Default;*
String **Parameters;*
Cardinal **NumberParameters;*

Description

The **XtErrorMsg** subroutine displays an error message. This subroutine is the high-level error handler. The *Class* parameter for all the Intrinsics internal errors have the **XtToolkitError** value.

Parameters

Name	Specifies the general kind of error.
Type	Specifies the detailed name of the error.
Class	Specifies the resource class.
Default	Specifies a default message if an error database entry is not found.
Parameters	Specifies a pointer to a list of values to be stored in the message.
NumberParameters	Specifies the number of values in the parameter list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtError** subroutine.

XtFree Subroutine

Purpose

Frees an allocated block of storage.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtFree(*Pointer*)
char **Pointer*;

Description

The **XtFree** subroutine frees an allocated block of storage. This subroutine returns the storage and allows it to be reused. The **XtFree** subroutine returns immediately if the *Pointer* parameter is the value of **Null**.

Parameter

Pointer Specifies a pointer to the block of storage to be freed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCalloc** subroutine, **XtMalloc** subroutine, **XtRealloc** subroutine.

XtGetApplicationResources Subroutine

Purpose

Retrieves the resources specific to the application.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetApplicationResources(*WidgetID*, *Base*, *Resources*, *NumberResources*, *Arguments*, *NumberArguments*)
Widget *WidgetID*;
caddr_t *Base*;
XtResourceList *Resources*;
Cardinal *NumberResources*;
ArgList *Arguments*;
Cardinal *NumberArguments*;

Description

The **XtGetApplicationResources** subroutine retrieves resources specific to an application. The **XtGetApplicationResources** subroutine does the following:

1. Constructs a resource name and class list with the specified widget, which is usually an application shell.
2. Retrieves the resources from the argument list, the resource database, or the resource list default values.
3. Adds the value of the *Base* parameter to each address and copies the resources into the *Resources* resource list.

Parameters

WidgetID	Specifies the widget that identifies the resource database to search. (This is the database associated with the display for this widget.)
Base	Specifies the base address of the subpart data structure where the resources should be written.
Resources	Specifies the resource list for the subpart.
NumberResources	Specifies the number of resources in the resource list.
Arguments	Specifies the argument list to override resources obtained from the resource database.
NumberArguments	Specifies the number of arguments in the argument list.

Note:
If the *Arguments* parameter is the value of **NULL**, the *NumberArguments* parameter must have the value of **0**. However, it the *NumberArguments* parameter has the value of **0**, the *Arguments* parameter is not referenced.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtGetErrorDatabase Subroutine

Purpose

Obtains the error database.

Library

Intrinsics Library (**libXt.a**)

Syntax

XrmDatabase *XtGetErrorDatabase()

Description

The **XtGetErrorDatabase** subroutine obtains the error database and returns the address of the error database.

The Intrinsics library does a lazy binding of the error database and does not merge in the database file until the first call to the **XtGetErrorDatabaseText** subroutine.

Note: The **XtGetErrorDatabase** exists as a convenience to users converting from earlier versions of the toolkit.

Return Value

A pointer to the database of the **XrmDatabse *** type.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppGetErrorDatabase** subroutine, **XtAppGetErrorDatabaseText** subroutine, **XtGetErrorDatabaseText** subroutine.

XtGetErrorDatabaseText Subroutine

Purpose

Obtains the error database text for an error or warning.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetErrorDatabaseText(*Name,Type,*
Class,Default,
BufferReturn,
NumberBytes)

char* *Name*, **Type*, **Class*;
char* *Default*;
char* *BufferReturn*;
int* *NumberBytes*;

Description

The **XtGetErrorDatabaseText** subroutine obtains the error database text for an error or warning. It returns the appropriate message or a default message (if no appropriate message was found) from the error database.

Note: The **XtGetErrorDatabaseText** subroutine exists as a convenience to users converting from earlier versions of the toolkit.

Parameters

Name	Specifies the name that is concatenated to form the resource name of the error message.
Type	Specifies the type that is concatenated to form the resource name of the error message.
Class	Specifies the resource class of the error message.
Default	Specifies the default message if an entry in the error database is not found.
BufferReturn	Specifies the buffer into which the error message is to be returned.
NumberBytes	Specifies the size of the buffer in bytes.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppGetErrorDatabase** subroutine.

XtGetGC Subroutine

Purpose

Returns a read-only shareable graphics context.

Library

Intrinsics Library (**libXt.a**)

Syntax

GC XtGetGC(*WidgetID*, *ValueMask*, *Values***)**
Widget *WidgetID*;
XtGCMask *ValueMask*;
XGCValues * *Values*;

Description

The **XtGetGC** subroutine returns a read-only shareable graphics context (**GC**). The parameters for this subroutine are the same as those for the **XCreateGC** subroutine except that a widget is passed instead of a display. It shares only the graphics contexts, where all values in the **GC** returned by the **XCreateGC** subroutine are the same. The **XtGetGC** subroutine uses the *ValueMask* parameter only to inform the server which fields should be filled in with widget data and which fields should be filled in with default values.

The Intrinsics library provides a mechanism which allows cooperating clients to share a **GC**, thereby reducing the number of **GC**'s created and the total number of server calls in any biven application. The mechanism is a simple caching scheme, and all **GC**'s obtained by means of this mechanism must be treated as read-only.

Parameters

WidgetID	Specifies the widget.
ValueMask	Specifies the value fields.
Values	Specifies the actual values for this graphics context.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtReleaseGC** subroutine, **XCreateGC** subroutine.

The **XGCValues** structure.

XtGetResourceList Subroutine

Purpose

Obtains the resource list structure for a particular class.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetResourceList(*Class*, *ResourcesReturn*, *NumberResourcesReturn*)
WidgetClass *Class*;
XtResourceList **ResourcesReturn*;
Cardinal **NumberResourcesReturn*;

Description

The **XtGetResourceList** subroutine obtains the resource list structure for a particular class.

The resource list specified in the widget class record is returned if the **XtGetResourceList** subroutine is called before the widget class is initialized. Otherwise, a merged resource list containing the resources for all superclasses is returned.

Parameters

Class	Specifies the widget class pointer for the created shell widget.
ResourcesReturn	Specifies a pointer to the storage place for the resource list returned. The **XtFree** subroutine must be called to free this storage.
NumberResourcesReturn	Specifies a pointer to the storage place for the number of entries in the resource list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtSetValues** subroutine, **XtGetValues** subroutine.

The **XtResource** structure.

XtGetSelectionTimeout Subroutine

Purpose

Obtains the current selection time-out value.

Library

Intrinsics Library (**libXt.a**)

Syntax

unsigned long XtGetSelectionTimeout()

Description

The **XtGetSelectionTimeout** subroutine obtains the current selection time-out value. The selection time-out is the time during which two communicating applications must respond to each other. If one application does not respond within this time, the Intrinsics ends execution of the selection request.

Use the **XtSetSelectionTimeout** subroutine to specify the selection time-out value. If a selection time-out value is not set, it defaults to 5 seconds.

Note: The **XtGetSelectionTimeout** subroutine exists as a convenience to users converting from earlier version of the toolkit.

Return Value

The time-out value.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtSetSelectionTimeout** subroutine.

XtGetSelectionValue Subroutine

Purpose

Obtains the selection value in a single logical unit.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetSelectionValue(*WidgetID*, *Selection*,
Target, *Callback*,
ClientData, *TimeStamp*)
Widget *WidgetID*;
Atom *Selection*;
Atom *Target*;
XtSelectionCallbackProc *Callback*;
caddr_t *ClientData*;
Time *TimeStamp*;

Description

The **XtGetSelectionValue** subroutine obtains the selection value in a single logical unit. This subroutine requests the value of the selection that has been converted to the target type.

The callback procedure communicates the selection values to the client. It can be called before or after the **XtGetSelectionValue** subroutine returns.

Parameters

WidgetID	Specifies the widget making the request.
Selection	Specifies the selection desired, for example, primary or secondary.
Target	Specifies the type of information needed about the selection.
Callback	Specifies the callback procedure to be called when the selection has been obtained.
ClientData	Specifies the argument for the specified callback procedure.
TimeStamp	Specifies the time stamp that indicates when the selection will be started. It should be the timestamp of the event which triggered this request. The **CurrentTime** value cannot be used for this parameter.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetSelectionValues** subroutine.

The **XtSelectionCallbackProc** data type.

XtGetSelectionValues Subroutine

Purpose

Obtains the current value of the selection converted to a list of targets.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetSelectionValues(*WidgetID*, *Selection*, *Targets*, *Count*, *Callback*, *ClientData*, *TimeStamp*)

Widget *WidgetID*;
Atom *Selection*;
Atom **Targets*;
int *Count*;
XtSelectionCallbackProc *Callback*;
caddr_t *ClientData*;
Time *TimeStamp*;

Description

The **XtGetSelectionValues** subroutine takes a list of target types and client data and obtains the current value of the selection converted to each of the targets.

The callback procedure, which communicates the selection values to the client, is called once with the corresponding client data for each target.

The **XtGetSelectionValues** subroutine guarantees that all the conversions will use the same selection value because ownership of the selection cannot change within the list.

Parameters

WidgetID	Specifies the widget making the request.
Selection	Specifies the particular selection (primary or secondary).
Targets	Specifies the types of information about the selection that is needed.
Count	Specifies the length of the targets and client data lists.
Callback	Specifies the callback procedure.
ClientData	Specifies the client data (one for each target type) passed to the callback procedure.
TimeStamp	Specifies the timestamp that indicates when the selection will be started. This should be the timestamp of the event which triggered this request. The **CurrentTime** value cannot be used for this parameter.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetSelectionValue** subroutine.

XtGetSubresources Subroutine

Purpose

Obtains resources from subparts of widgets that are not widgets themselves.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetSubresources(*WidgetID*, *Base*, *Name*, *Class*, *Resources*, *NumberResources*, *Arguments*, *NumberArguments*)
Widget *WidgetID*;
caddr_t *Base*;
String *Name*;
String *Class*;
XtResourceList *Resources*;
Cardinal *NumberResources*;
ArgList *Arguments*;
Cardinal *NumberArguments*;

Description

The **XtGetSubresources** subroutine obtains resources from widget subparts that are not widgets themselves. The **XtGetSubresources** subroutine constructs a name and class list from the application name and application class the names and classes of all its ancestors, and the widget itself. Then it appends to this list the name and class pair passed in. The resources are taken from the argument list, the resource database, or the default values in the resource list. Then they are copied into the subpart record.

Parameters

WidgetID	Specifies the widget that wants resources for a subpart.
Base	Specifies the base address of the subpart data structure where the resources should be written.
Name	Specifies the name of the subpart.
Class	Specifies the class of the subpart.
Resources	Specifies the resource list for the subpart.
NumberResources	Specifies the number of resources in the resource list.
Arguments	Specifies the argument list to override resources obtained from the resource database.
NumberArguments	Specifies the number of arguments in the argument list. **Note:** If the *Arguments* parameter is the value of **NULL**, the *NumberArguments* parameter be 0. However, if the *NumberArguments* parameter is 0, the argument list is not referenced.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCreateWidget** subroutine, **XtGetApplicationResources** subroutine.

The **XtResource** data structure.

XtGetSubvalues Subroutine

Purpose

Retrieves the current value of non-widget resource data associated with a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetSubvalues(*Base*, *Resources*, *NumberResources*, *Arguments*, *NumberArguments*)
caddr_t *Base*;
XtResourceList *Resources*;
Cardinal *NumberResources*;
ArgList *Arguments*;
Cardinal *NumberArguments*;

Description

The **XtGetSubvalues** subroutine retrieves the current value of a non-widget resource data associated with a widget instance. It obtains the resource values from the structure identified by the base.

Parameters

Base	Specifies the base address of the subpart data structure from which the resources should be retrieved.
Resources	Specifies the non-widget resources list.
NumberResources	Specifies the number of resources in the resource list.
Arguments	Specifies the argument list of name and address pairs with the resource name and the address into which the resource value is to be stored. The arguments and values (passed) are dependent on the subpart of the widget. The storage used for the argument list must be deallocated by the application when it is no longer needed.
NumberArguments	Specifies the number of arguments in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetValues** subroutine, **XtSetValues** subroutine, **XtSetSubvalues** subroutine.

The **XtResource** structure.

The **XtArgsProc** data type, **XtSetValuesFunc** data type, **XtArgsFunc** data type.

XtGetValues Subroutine

Purpose

Retrieves the current value of a resource associated with a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtGetValues(*WidgetID*, *Arguments*, *NumberArguments***)**
Widget *WidgetID*;
ArgList *Arguments*;
Cardinal *NumberArguments*;

Description

The **XtGetValues** subroutine retrieves the current value of a resource associated with a widget instance. It starts with the resources specified for the core widget fields and proceeds down the subclass chain to the widget. The value field of the *Arguments* parameter list should contain the resource name and the address in which the resource value will be stored. The calling subroutine should allocate and deallocate this storage space according to the size of the resource representation type used within the widget.

If the parent of the widget is a subclass of the **constraintWidgetClass**, then the **XtGetValues** subroutine obtains the values for any constraint resources requested. The **XtGetValues** subroutine starts with the constraint resources specified for the **constraintWidgetClass** and proceeds down the subclass chain to the constraint resources of the parent.

If the argument list contains a resource name that is not found in the resource lists, the value at the corresponding address is not modified.

If the *get_values_hook* fields are a value other than **NULL**, they are called in superclass-to-subclass order after the resource values have been obtained by the **XtGetValues** subroutine. This permits a subclass to provide non-widget resource data to the **XtGetValues** subroutine.

Parameters

WidgetID	Specifies the widget.
Arguments	Specifies the argument list of name and address pairs that contain the resource name and address into which the resource value is to be stored. Resource names are widget dependent.
NumberArguments	Specifies the number of arguments in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetSubvalues** subroutine, **XtSetValues** subroutine, **XtSetSubvalues** subroutine.

The **XtResource** structure.

The **XtArgsProc** data type, **XtSetValuesFunc** data type, **XtArgsFunc** data type.

XtGrabKey Subroutine

Purpose

Establishes a passive grab on the specified keys.

Library

Intrinsics Library (**libXt.a**)

Syntax

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

void XtGrabKey(WidgetID, Keycode, Modifiers,
               OwnerEvents, PointerMode,
               KeyboardMode)
Widget WidgetID;
int Keycode;
unsigned int Modifiers;
Boolean OwnerEvents;
int PointerMode;
int KeyboardMode;
```

Description

The **XtGrabKey** subroutine establishes a passive grab on the specified keys; when the specified key-modifier combination is pressed, the keyboard is grabbed. It also allows the client to redirect the specified key event to the root widget of a hierarchy.

Parameters

Widget	Specifies the root widget to the **XtGrabKeyboard** call. All key events that would have been dispatched to other subwindows are dispatched to the root widget subject to the *OwnerEvents* parameter.
KeyCode	Specifies the key code. This maps to the specific key to be grabbed.
Modifiers	Specifies the set of key masks. This mask is the bitwise-inclusive OR of these key mask bits: the **ShiftMask**, **LockMask**, **ControlMask**, **Mod1Mask**, **Mod2Mask**, **Mod3Mask**, **Mod4Mask**, and **Mod5Mask** bits. You can also pass the **AnyModifier** bit, which is equivalent to issuing the grab key request for all possible modifier combinations, including the combination of no modifiers.
OwnerEvents	Specifies if the pointer events are to be reported normally (pass the value of **True**) or with respect to the grab window if selected by the event mask (pass the value of **False**).
PointerMode	Specifies further processing of pointer events. You can pass the **GrabModeSync** or **GrabModeAsync** value.
KeyboardMode	Specifies further processing of keyboard events. You can pass the **GrabModeSync** or **GrabModeAsync** value.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XGrabKey** subroutine, **XtUngrabKey** subroutine.

XtGrabKeyboard Subroutine

Purpose

Actively grabs control of the main keyboard.

Library

Intrinsics Library (**libXt.a**)

Syntax

#include <X11/PassivGrab.h>

int XtGrabKeyboard(*WidgetID, OwnerEvents, PointerMode, KeyboardMode, TimeStamp***)**
Widget *WidgetID*;
Boolean *OwnerEvents*;
int *PointerMode*;
int *KeyboardMode*;
Time *TimeStamp*;

Description

The **XtGrabKeyboard** subroutine actively grabs control of the main keyboard. If the grab is successful, it returns the constant **GrabSuccess**. Further key events are reported to the grab widget.

Parameters

WidgetID	Specifies the root widget to the **XtGrabKeyboard** call. All key events that would have been dispatched to other subwindows will get dispatched to the root widget subject to the *OwnerEvents* parameter.
OwnerEvents	Specifies if the pointer events are to be reported normally (pass the value of **True**) or with respect to the grab window if selected by the event mask (pass the value of **False**).
PointerMode	Specifies further processing of pointer events. You can pass the **GrabModeSync** or **GrabModeAsync** value.
KeyboardMode	Specifies further processing of keyboard events. You can pass the **GrabModeSync** or **GrabModeAsync** value.
TimeStamp	Specifies the time. You can pass either a time stamp, expressed in milliseconds, or the **CurrentTime** value.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtUngrabKeyboard** subroutine.

XtHasCallbacks Subroutine

Purpose

Gives the status of a specified widget callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

```
typedef enum {XtCallbackNoList, XtCallbackHasNone,
                    XtCallbackHasSome}XtCallbackStatus;

XtCallbackStatus XtHasCallbacks(WidgetID, CallbackName)
Widget WidgetID;
String CallbackName;
```

Description

The **XtHasCallbacks** subroutine finds the status of a specified widget callback list. First, it searches for a callback list identified by the *CallbackName* parameter. Then it returns the appropriate return value.

Parameters

CallbackName	Specifies the callback list to be checked.
WidgetID	Specifies the widget.

Return Values

XtCallbackNoList	Indicates that the callback list does not exist.
XtCallbackHasNone	Indicates that the callback list exists but is empty.
XtCallbackHasSome	Indicates that the callback list exists and contains at least one registered callback procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddCallback** subroutine, **XtAddCallbacks** subroutine, **XtCallCallbacks** subroutine, **XtRemoveCallback** subroutine, **XtRemoveCallbacks** subroutine.

The **XtCallbackList** data structure.

XtInitialize Subroutine

Purpose

Initializes the toolkit internals.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtInitialize(*ShellName*, *ApplicationClass*, *Options*, *NumberOptions*, *argc*, *argv*)
String *ShellName*;
String *ApplicationClass*;
XrmOptionDescRec *Options*[ ];
Cardinal *NumberOptions*;
Cardinal **argc*;
String *argv*[ ];

Description

The **XtInitialize** subroutine does the following:

1. Calls the **XtToolkitInitialize** subroutine to initialize the toolkit internals.
2. Creates a default application context for use by other utility routines.
3. Calls the **XtOpenDisplay** subroutine with the value of **NULL** *DisplayString* and *ApplicationName* parameters.
4. Calls the **XtAppCreateShell** subroutine with the value of **NULL** *ApplicationName* parameter.
5. Returns the created shell.

If the **XtInitialize** subroutine is called more than once, the behavior of the toolkit is undefined.

Note: This subroutine exists as a convenience for users converting from earlier versions of the toolkit.

Parameters

ShellName	This parameter is ignored. It may be the value of **NULL**.
ApplicationClass	Specifies the class name of this application.
Options	Specifies how to parse the command line for any application-specific resources. This argument is passed as a parameter to the **XrmParseCommand** subroutine.
NumberOptions	Specifies the number of entries in the options list.
argc	Specifies a pointer to the number of command line parameters.
argv	Specifies the command line parameters.

Return Value

The widget representing the created shell.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtToolkitInitialize** subroutine, **XtDisplayInitialize** subroutine, **XtOpenDisplay** subroutine.

XtInstallAccelerators Subroutine

Purpose

Installs accelerators from one widget to another.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtInstallAccelerators(*Destination, Source***)**
Widget *Destination*;
Widget *Source*;

Description

The **XtInstallAccelerators** subroutine installs accelerators from a source widget to a destination widget. It installs the accelerators by augmenting the destination translations with the source accelerators.

If the source *display_accelerator* procedure is a value other than **NULL**, the **XtInstallAccelerators** subroutine calls it with the source widget and a string representation of the accelerator table. This indicates that the accelerators have been installed and should be displayed appropriately.

The string representation of the accelerator table is its canonical translation table representation.

Parameters

Destination	Specifies the widget that receives the accelerators.
Source	Specifies the widget that supplies the accelerators.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtParseAcceleratorTable** subroutine, **XtInstallAllAccelerators** subroutine.

The **XtStringProc** data type.

XtInstallAllAccelerators Subroutine

Purpose

Installs all accelerators from a widget and all its descendants onto one destination.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtInstallAllAccelerators(*Destination*, *Source*)
Widget *Destination*;
Widget *Source*;

Description

The **XtInstallAllAccelerators** subroutine installs all accelerators from a widget and all its descendants onto one destination. It recursively descends the widget tree rooted at the widget in the *Source* parameter. Then the subroutine installs the accelerators that it encounters onto the widget in the *Destination* parameter.

Parameters

Destination	Specifies the widget receiving the accelerators.
Source	Specifies the root widget supplying the accelerators.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtInstallAccelerators** subroutine, **XtParseAccelerators** subroutine.

The **XtStringProc** data type.

XtIsComposite Macro

Purpose

Determines whether a specified widget is a subclass of the **Composite** class.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtIsComposite(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtIsComposite** macro determines whether a specified widget is a subclass of the **Composite** class. This subroutine is equivalent to the **XtIsSubclass** subroutine with the **compositeWidgetClass** value specified for the *WidgetClass* parameter.

Parameter

WidgetID	Specifies the widget.

Return Values

True	Indicates that the specified widget is a subclass of the **Composite** class.
False	Indicates that the specified widget is not a subclass of the **Composite** class.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtIsSubclass** subroutine.

XtIsManaged Macro

Purpose

Determines the managed state of a specified child widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtIsManaged(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtIsManaged** macro determines the managed state of a specified child widget.

Parameter

WidgetID	Specifies the widget.

Return Values

False	Indicates that the child widget is not managed.
True	Indicates that the child widget is managed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtManageChild** subroutine, **XtCreateManagedWidget** subroutine, **XtUnmanageChild** subroutine.

XtIsRealized Macro

Purpose

Determines if a widget has been realized.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtIsRealized(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtIsRealized** macro determines if a widget has been realized.

Parameter

WidgetID	Specifies the widget.

Return Values

False	Indicates that the widget is not realized.
True	Indicates that the widget is realized; in other words, the widget has a nonzero Enhanced X-Windows.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRealizeWidget** subroutine.

The **XtRealizeProc** data type.

XtIsSensitive Macro

Purpose

Determines the current sensitivity state of a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtIsSensitive(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtIsSensitive** macro determines the current sensitivity state of a widget. This subroutine is usually called by the parent widget. It indicates whether user input events are being dispatched.

When a widget is dormant, it is considered insensitive. Therefore, the event manager does not dispatch the following events:

ButtonPress **KeyPress**

ButtonRelease **KeyRelease**

EnterNotify **LeaveNotify**

FocusIn **MotionNotify**

FocusOut

Many widgets appear differently when dormant.

A widget can be sensitive because its sensitive field is the value of **False** or because one of its parents is insensitive, and thus the widget's *ancestor_sensitive* field is the value of **False**. A widget does not need to distinguish between these two cases visually.

Parameters

WidgetID	Specifies the widget to be checked.

Return Values

False	Indicates that either the *sensitive* or *ancestor_sensitive* fields of the core are the value of **False**.
True	Indicates that both the *sensitive* and *ancestor_sensitive* fields of the core are the value of **True**.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtSetSensitive** subroutine.

XtIsSubclass Subroutine

Purpose

Determines the subclass of a specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtIsSubclass(*WidgetID*, *Class***)**
Widget *WidgetID*;
WidgetClass *Class*;

Description

The **XtIsSubclass** subroutine determines the subclass of a specified widget. The widget does not need to be an immediate subclass of the value specified in the *WidgetClass* parameter; it can be farther down the subclass chain.

Composite widgets that need to restrict the class of the items they contain can use the **XtIsSubclass** subroutine to find out if the widget belongs to the desired class of objects.

Parameters

WidgetID	Specifies the widget.
Class	Specifies the widget class to test against.

Return Values

False	Indicates that the widget is not a subclass of the value specified in the *WidgetClass* parameter.
True	Indicates that the widget is equal to or is a subclass of the value specified in the *WidgetClass* parameter.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtClass** subroutine, **XtSuperclass** subroutine, **XtCheckSubclass** subroutine.

XtMainLoop Subroutine

Purpose

Processes input.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtMainLoop()

Description

The **XtMainLoop** subroutine processes input. First, it reads the next incoming file, timer, or X event by calling the **XtNextEvent** subroutine. Then it dispatches this event to the appropriate registered procedure by calling the **XtDispatchEvent** subroutine.

The **XtMainLoop** subroutine is an infinite loop and does not return a value. Applications should exit directly in response to a user action.

Note: This subroutine exists only as a convenience to users converting from earlier versions of the toolkit; it has been replaced by the **XtAppMainLoop** subroutine.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppMainLoop** subroutine, **XtNextEvent** subroutine, **XtDispatchEvent** subroutine.

XtMakeGeometryRequest Subroutine

Purpose

Makes a general geometry manager request from a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtGeometryResult XtMakeGeometryRequest(*WidgetID*, *Request*, *ReplyReturn*)
Widget *WidgetID*;
XtWidgetGeometry **Request*;
XtWidgetGeometry **ReplyReturn*;

Description

The **XtMakeGeometryRequest** subroutine is a request from the child widget to a parent widget for a geometry change. The geometry manager returns one of the following values: the **XtGeometryYes, XtGeometryNo,** or **XtGeometryAlmost** value. The **XtMakeGeometryRequest** subroutine does not return the **XtGeometryDone** value.

Depending on the conditions, the **XtMakeGeometryRequest** subroutine performs the following:

- Makes the changes and returns the **XtGeometryYes** value, if the widget is unmanaged or the widget parent is not realized.
- Issues an error if the parent is not a subclass of the **compositeWidgetClass** class or the geometry manager of the parent is the value of **NULL**.
- Returns the **XtGeometryNo** value if the *being_destroyed* field of the widget is the value of **True**.
- Returns the **XtGeometryYes** value if the *x, y, width, height,* and *border_width* fields of the widget are all equal to the requested values; otherwise, it calls the *geometry_manager* field of the parent widget with the specified parameters.
- Calls the **XConfigureWindow** subroutine to reconfigure the widget window, if the geometry manager of the parent widget returns the **XtGeometryYes** value, the *request_mode* field does not have the **XtCWQuery** value, and the widget is realized.
- Does no configuring and returns the **XtGeometryYes** value if the geometry manager returns the **XtGeometryDone** value.

If none of the above conditions apply, the **XtMakeGeometryRequest** subroutine returns the results from the parent geometry manager.

Since children of primitive widgets are always unmanaged, this subroutine always returns the **XtGeometryYes** value when called by a child widget of a primitive widget.

Parameters

WidgetID	Specifies the widget ID of the widget making the request.
Request	Specifies the desired widget geometry (size, position, border width, and stacking order).

ReplyReturn	Returns the allowed widget size. If a widget is not interested in an **XtGeometryAlmost** value, the *ReplyReturn* parameter can be the value of **NULL**.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Return Values

XtGeometryAlmost

XtGeometryNo

XtGeometryYes

Related Information

The **XtMakeResizeRequest** subroutine.

The **XtWidgetGeometry** data structure.

The **XtGeometryHandler** data type.

XtMakeResizeRequest Subroutine

Purpose

Makes a simple resize request from a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtGeometryResult XtMakeResizeRequest(*WidgetID*, *Width*, *Height*, *WidthReturn*, *HeightReturn*)

Widget *WidgetID*;
Dimension *Width*, *Height*;
Dimension **WidthReturn*, **HeightReturn*;

Description

The **XtMakeResizeRequest** subroutine initiates a resize request from a widget. A child widget can use this routine to request a geometry change to the parent widget. This subroutine is an interface to the **XtMakeGeometryRequest** subroutine. It creates an **XtWidgetGeometry** structure and specifies that width and height should change. At this time, the geometry manager can modify any of the other window attributes, such as position or stacking order, to satisfy the resize request.

If the **XtGeometryAlmost** value is returned, the *WidthReturn* and *HeightReturn* parameters contain a compromise width and height. If this compromise is acceptable, the widget should immediately initiate the **XtMakeResizeRequest** subroutine for the compromise width and height. If the widget is not interested in the **XtGeometryAlmost** value, it can pass the value of **NULL** for the *WidthReturn* and *HeightReturn* parameters.

Parameters

WidgetID	Specifies the widget.
Width	Specifies the desired widget width.
Height	Specifies the desired widget height.
WidthReturn	Returns the allowed widget width.
HeightReturn	Returns the allowed widget height.

Return Values

XtGeometryYes

XtGeometryNo

XtGeometryAlmost

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtMakeGeometryRequest** subroutine.

The **XtGeometryHandler** data type.

The **XtWidgetGeometry** data structure.

XtMalloc Subroutine

Purpose

Allocates storage.

Library

Intrinsics Library (**libXt.a**)

Syntax

char *XtMalloc(*Size***)**
Cardinal *Size*;

Description

The **XtMalloc** subroutine allocates storage. It returns a pointer to a block of storage as specified in the *Size* parameter. If there is insufficient memory to allocate the new block, the **XtMalloc** subroutine calls the **XtErrorMsg** subroutine.

Parameter

Size	Specifies the size of the storage requested in bytes.

Return Value

Pointer to the block of storage.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCalloc** subroutine, **XtRealloc** subroutine, **XtFree** subroutine, **XtNew** subroutine, **XtNewString** subroutine.

XtManageChild Subroutine

Purpose

Adds a single child widget to a parent widget list of managed children widgets.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtManageChild(*Child***)**
Widget *Child*;

Description

The **XtManageChild** subroutine adds a single child widget to a parent widget list of managed children widgets. This subroutine is called after the **XtCreateWidget** subroutine creates the child widget.

The **XtManageChild** subroutine constructs a widget list of one widget; then it calls the **XtManageChildren** subroutine.

Parameter

Child Specifies the child widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtIsManaged** macro.

The **XtCreateManagedWidget** subroutine, **XtManageChildren** subroutine, **XtUnmanageChildren** subroutine.

XtManageChildren Subroutine

Purpose

Adds a list of widgets to the geometry-managed, displayable subset of the composite parent widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

typedef Widget *WidgetList;

void XtManageChildren(*Children*, *NumberChildren***)**
WidgetList *Children*;
Cardinal *NumberChildren*;

Description

The **XtManageChildren** subroutine adds a list of widgets to the geometry-managed, displayable subset of the composite parent widget. This routine is called after the child widgets have been created with the **XtCreateWidget** subroutine. The **XtManageChildren** procedure performs the following tasks:

- Issues an error if all the children widgets do not have the same parent widget or if the parent widget is not a subclass of the **compositeWidgetClass**.
- Returns immediately if the common parent widget is being destroyed. Otherwise, for each unique child widget on the list, it ignores the child widget if the child widget is already managed or is being destroyed. If neither case applies, this procedure marks the child widget.

If the parent widget is realized after all children widgets have been marked, the **XtManageChildren** procedure makes some of the newly managed children widgets visible in the following manner:

- Calls the procedure specified in the *change_managed* field of the parent widget.
- Calls the **XtRealizeWidget** subroutine for each previously unmanaged, unrealized child widget.
- Maps each previously unmanaged child that has a *map_when_managed* field that is the value of **True**.

Parameters

Children	Specifies a list of child widgets.
NumberChildren	Specifies the number of children widgets.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCreateWidget** subroutine, **XtRealizeWidget** subroutine.

XtMapWidget Subroutine

Purpose

Maps a widget explicitly.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtMapWidget(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtMapWidget** subroutine maps a widget explicitly.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtUnmapWidget** subroutine, **XtSetMappedWhenManaged** subroutine.

XtMergeArgLists Subroutine

Purpose

Merges two **ArgLists** structures.

Library

Intrinsics Library (**libXt.a**)

Syntax

ArgList XtMergeArgLists(*Argument1*, *NumberArguments1*,
Argument2, *NumberArguments2*)
ArgList *Argument1*;
Cardinal *NumberArguments1*;
ArgList *Argument2*;
Cardinal *NumberArguments2*;

Description

The **XtMergeArgLists** subroutine merges two **ArgList** structures. It allocates sufficient storage to hold the combined **ArgList** structures and copies them into that space. The **XtMergeArgLists** subroutine does not check the **ArgList** structures for duplicate entries.

Use the **XtFree** subroutine to deallocate the storage space after this routine completes.

Parameters

Argument1	Specifies the first argument list.
NumberArguments1	Specifies the number of arguments in the first argument list.
Argument2	Specifies the second argument list.
NumberArguments2	Specifies the number of arguments in the second argument list.

Return Value

Pointer to the merged argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtSetArg** subroutine.

The **ArgList** data structure.

XtMoveWidget Subroutine

Purpose

Moves a sibling widget of the child widget making the geometry request.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtMoveWidget(*WidgetID*, *X*, *Y*)
Widget *WidgetID*;
Position *X*;
Position *Y*;

Description

The **XtMoveWidget** subroutine moves a sibling widget of the child making the geometry request. It writes the new values for the *X* and *Y* parameters into the widget.

If the widget is realized, the **XtMoveWidget** subroutine issues calls to the **XMoveWindow** subroutine on the window of the widget.

The **XtMoveWidget** subroutine returns immediately if the specified geometry fields are the same as the old values.

Parameters

WidgetID	Specifies the widget.
X	Specifies the new widget x coordinate.
Y	Specifies the new widget y coordinate.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtResizeWidget** subroutine, **XtConfigureWidget** subroutine, **XtResizeWindow** subroutine, **XMoveWindow** subroutine.

XtNameToWidget Subroutine

Purpose

Translates a widget name to a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtNameToWidget(*Reference*, *Names***)**
Widget *Reference*;
String *Names*;

Description

The **XtNameToWidget** subroutine translates a widget name to a widget instance. It searches for a widget whose name is the first component in the *Names* parameter and that is a pop-up child widget of the specified reference widget (or a normal child widget if the reference widget is a subclass of the **compositeWidgetClass**). Then, it uses the widget as the new reference widget and repeats the search after deleting the first component from the *Names* parameter.

The Intrinsics library does not require unique names for the children of a widget. The **XtNameToWidget** subroutine does not back up and follow other matching branches of the widget tree.

Parameters

Reference	Specifies the widget from which to start the search.
Names	Specifies the fully qualified name of the desired widget.

Return Values

NULL	Indicates that the **XtNameToWidget** subroutine cannot find the specified widget. This value is also returned when the single branch the **XtNameToWidget** subroutine follows does not contain the named widget if the *Names* parameter contains more than one component, and more than one child matches the first component.
WidgetID	Specifies the widget corresponding to the *Names* parameter. This subroutine can return any of the child widgets if more than one child widget of the reference widget matches the name.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtNew Subroutine

Purpose

Allocates storage for a new instance of a data type.

Library

Intrinsics Library (**libXt.a**)

Syntax

Type ***XtNew**(*Type*)
Type;

Description

The **XtNew** subroutine allocates storage for a new instance of a data type. It returns a pointer to the allocated storage. If there is insufficient memory to allocate the new block, it calls the **XtErrorMsg** subroutine.

The **XtNew** subroutine is a convenience macro which calls the **XtMalloc** subroutine with the following arguments:

```
((type *) XtMalloc((unsigned) sizeof(type))
```

Parameter

Type Specifies a previously declared data type.

Return Value

Pointer to the allocated block of storage.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtNewString** subroutine, **XtFree** subroutine, **XtRealloc** subroutine, **XtCalloc** subroutine, **XtMalloc** subroutine.

XtNewString Macro

Purpose

Copies an instance of a string.

Library

Intrinsics Library (**libXt.a**)

Syntax

String XtNewString(*StringID***)**
String *StringID*;

Description

The **XtNewString** macro copies an instance of a string. It returns a pointer to the allocated storage. If there is insufficient memory to allocate the new block, this routine calls the **XtErrorMsg** subroutine.

The **XtNewString** subroutine macro is a convenience macro which calls the **XtMalloc** subroutine with the following parameters:

```
(strcpy(XtMalloc((unsigned)strlen(str) + 1), str))
```

Parameter

StringID Specifies a previously declared string.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtNew** subroutine, **XtMalloc** subroutine, **XtFree** subroutine, **XtRealloc** subroutine, **XtCalloc** subroutine.

XtNextEvent Subroutine

Purpose

Returns the value from the header of the input queue.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtNextEvent(*EventReturn***)**
XEvent* *EventReturn*;

Description

The **XtNextEvent** subroutine returns the value from the header of the input queue.

If there is no input on the Enhanced X-Windows input queue, the **XtNextEvent** subroutine flushes the output buffer. It waits for an event while looking at the other input sources and time-out values, calling any callback procedures triggered by them.

The **XtInitialize** subroutine must be called before using the **XtNextEvent** subroutine.

Note: This subroutine exists only as a convenience to users converting from earlier versions of the toolkit; it has been replaced by the **XtAppNextEvent** subroutine.

Parameter

EventReturn	Returns the event information to the specified event structure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtInitialize** subroutine, **XtAppNextEvent** subroutine, **XtMainLoop** subroutine.

XtNumber Subroutine

Purpose

Determines the number of elements in a fixed-size array.

Library

Intrinsics Library (**libXt.a**)

Syntax

Cardinal XtNumber(*Array***)**
ArrayVariable *Array*;

Description

The **XtNumber** subroutine determines the number of elements in a fixed-size array. It returns the number of elements in the specified argument lists, resources lists, and other counted arrays.

Parameter

Array Specifies a fixed-size array.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtOffset Macro

Purpose

Determines the byte offset of a resource field within a structure.

Library

Intrinsics Library (**libXt.a**)

Syntax

Cardinal XtOffset(*PointerType*, *FieldName*)
Type *PointerType*;
Field *FieldName*;

Description

The **XtOffset** macro determines the byte offset of a resource field within a structure. It calculates the offset from the beginning of a widget and can be used at compile time in static initializations.

Parameters

PointerType	Specifies a type declared as a pointer to the structure.
FieldName	Specifies the name of the field for which to calculate the byte offset.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtOpenDisplay Subroutine

Purpose

Opens, initializes, and adds a display to an application context.

Library

Intrinsics Library (**libXt.a**)

Syntax

Display *XtOpenDisplay(*ApplicationContext***,** *DisplayString***,**
*ApplicationName***,** *ApplicationClass***,**
*Options***,** *NumberOptions***,**
*argc***,** *argv***)**
XtAppContext *ApplicationContext*;
String *DisplayString*;
String *ApplicationName*;
String *ApplicationClass*;
XrmOptionDescRec **Options*;
Cardinal *NumberOptions*;
Cardinal **argc*;
String **argv*;

Description

The **XtOpenDisplay** subroutine opens, initializes, and adds a display to an application context. It calls the **XOpenDisplay** subroutine with the specified display name.

If the *DisplayString* parameter is the value of **NULL**, the **XtOpenDisplay** subroutine uses the current value of the *-display* option specified in the *argv* parameter.

If no display name is specified in the *argv* parameter, the user's default display is used (usually the value of the DISPLAY environment variable on UNIX based systems).

After a display name is found, the **XtOpenDisplay** subroutine calls the **XtDisplayInitialize** subroutine, passing it the opened display and the value of the *-name* option specified in the *argv* parameter as the application name. If no *-name* option is specified, the value specified in the *ApplicationName* parameter is used. If the *ApplicationName* parameter is the value of **NULL**, the last component of the *argv*[0] parameter is used.

Values for the *DisplayString* and *ApplicationName* parameters found in the *argv* parameter to the **XtOpenDisplay** subroutine override the values for the same parameter in the **XtDisplayInitialize** subroutine.

Parameters

ApplicationContext	Specifies the application context.
DisplayString	Specifies the display string. Each display can be in only one application context.
ApplicationName	Specifies the name of the application instance.
ApplicationClass	Specifies the class name of this application, usually the generic name for all instances of this application.

Options	Specifies how to parse the command line for any application-specific resources. The *Options* parameter is passed to the **XrmParseCommand** subroutine.
NumberOptions	Specifies the number of entries in the options list.
argc	Specifies a pointer to the number of command line parameters.
argv	Specifies the command line parameters.

Return Values

NULL	Indicates that the subroutine was unsuccessful in opening the specified display.
Pointer to display	Indicates that the subroutine was successful.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtDisplayInitialize** subroutine, **XtCloseDisplay** subroutine, **XtOpenDisplay** subroutine.

XtOverrideTranslations Subroutine

Purpose

Overwrites existing translations with new translations.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtOverrideTranslations(*WidgetID*, *Translations*)
Widget *WidgetID*;
XtTranslations *Translations*;

Description

The **XtOverrideTranslations** subroutine destructively merges new translations into existing widget translations. The old translations do not exist once they have been written over. If the new translations contain an event or event sequence that already exists for the widget, the new translation is merged in and overrides the existing widget.

To replace a widget's translations completely, use the **XtSetValues** subroutine on the **XtNtranslations** resource and specify a compiled translation as the value.

Parameters

WidgetID	Specifies the widget into which the new translations are to be merged.
Translations	Specifies the compiled translation table to merge. This parameter must not be the value of **NULL**.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtParseTranslationTable** subroutine, **XtAugmentTranslations** subroutine, **XtUninstallTranslations** subroutine, **XtSetValues** subroutine.

XtOwnSelection Subroutine

Purpose

Sets the selection owner.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtOwnSelection(*WidgetID*, *Selection*, *TimeStamp*,
ConvertProcedure,
LoseSelection,
DoneProcedure)
Widget *WidgetID*;
Atom *Selection*;
Time *TimeStamp*;
XtConvertSelectionProc *ConvertProcedure*;
XtLoseSelectionProc *LoseSelection*;
XtSelectionDoneProc *DoneProcedure*;

Description

The **XtOwnSelection** subroutine sets the selection owner when using atomic transfers. It informs the Intrinsics selection mechanism that a widget believes it owns a selection.

This widget may not become the selection owner for one of the following reasons:

- If another widget asserts ownership at a time later than the specified time stamp. (The specified *TimeStamp* parameter triggers this event.)
- If this widget surrenders ownership of the selection voluntarily.

The procedure specified in the *LoseSelection* parameter is not called if the widget does not obtain selection ownership.

Parameters

WidgetID	Specifies the widget that will become the owner.
Selection	Specifies an atom that describes the type of the selection (for example, the **XA_PRIMARY**, **XA_SECONDARY**, or **XA_CLIPBOARD** values).
TimeStamp	Specifies the timestamp. This should be the timestamp of the event that triggered ownership. (The **CurrentTime** value cannot be used.)
ConvertProcedure	Specifies the procedure to call when the current value of the selection is requested.
LoseSelection	Specifies the procedure to call when the widget loses selection ownership. This parameter is set to the value of **NULL** if the owner is not interested in being called back.

DoneProcedure	Specifies the procedure to call after the requester has received the selection. This parameter is set to the value of **NULL** if the owner is not interested in being called back.

Return Values

False	Indicates that the widget has not become the selection owner.
True	Indicates that the widget has become the selection owner.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtDisownSelection** subroutine.

The **XtConvertSelection** data type, **XtLoseSelectionProc** data type, **XtSelectionDone** data type.

XtParent Macro

Purpose

Returns the parent widget for the specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtParent(*WidgetID***)**
Widget ***WidgetID*****;**

Description

The **XtParent** macro returns the parent widget for the specified widget.

Parameter

WidgetID	Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Return Value

Parent of the specified widget.

XtParseAcceleratorTable Subroutine

Purpose

Parses an accelerator table.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtAccelerators XtParseAcceleratorTable(*Source***)**
String *Source*;

Description

The **XtParseAcceleratorTable** subroutine parses an accelerator table into the opaque internal representation. An accelerator is a translation table that is bound with its actions in the context of a particular widget.

Parameter

Source Specifies the accelerator table to compile.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtInstallAccelerators** subroutine, **XtInstallAllAccelerators** subroutine.

XtParseTranslationTable Subroutine

Purpose

Compiles a translation table.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtTranslations XtParseTranslationTable(*Table***)**
String *Table*;

Description

The **XtParseTranslationTable** subroutine compiles a translation table into the opaque internal representation of the **XtTranslations** type. If an empty translation table is required for any purpose, it can be obtained by calling the **XtParseTranslationTable** subroutine and passing an empty string.

Parameter

Table Specifies the translation table to compile.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAugmentTranslations** subroutine, **XtOverrideTranslations** subroutine, **XtUninstallTranslations** subroutine.

XtPeekEvent Subroutine

Purpose

Returns the value from the front of the input queue without removing it from the queue.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtPeekEvent(*EventReturn***)**
XEvent * *EventReturn***;**

Description

The **XtPeekEvent** subroutine returns the value from the front of the input queue without removing it from the queue.

If there is no X input in the queue, the **XtPeekEvent** subroutine flushes the output buffer and blocks until input is available, possibly calling some time-out callback procedures in the process. If the input is an event (or if there is already an event in the queue) this subroutine fills in the event and returns a nonzero value. Otherwise the input is for an alternate input source, and the **XtPeekEvent** subroutine returns a value of 0.

The **XtInitialize** subroutine must be called before using this routine.

Note: This subroutine exists only as a convenience to users converting from earlier versions of the toolkit; it has been replaced by the **XtAppPeekEvent** subroutine.

Parameter

EventReturn	Returns the event information to the specified event structure.

Return Values

Nonzero	Indicates that the input in the queue is an event.
0	Indicates that the input in the queue is for an alternate input source.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppPeekEvent** subroutine, **XPeekEvent** subroutine.

XtPending Subroutine

Purpose

Determines if the input queue has events pending.

Library

Intrinsics Library (**libXt.a**)

Syntax

Boolean XtPending();

Description

The **XtPending** subroutine determines if the input queue has events pending from the X Server or other input sources in the default application context. If there are no events pending, it flushes the output buffer and returns to the value of **0**. The **XtInitialize** subroutine must be called before using this routine.

Note: This subroutine exists only as a convenience to users converting from earlier version of the toolkit; it has been replaced by the **XtAppPending** subroutine.

Return Values

True	Indicates that there are events pending from the X server or other input sources in the default application context.
False	Indicates that the input was for an alternate input source.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppPending** subroutine, **XPending** subroutine.

XtPopdown Subroutine

Purpose

Unmaps a pop-up from within an application.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtPopdown(*PopupShell***)**
Widget *PopupShell*;

Description

The **XtPopdown** subroutine unmaps a pop-up from within an application. It does the following:

- Calls the **XtCheckSubclass** subroutine to ensure that the shell specified is a subclass of the **Shell** widget class.
- Generates an error if the shell specified in the *PopupShell* parameter is not currently popped up.
- Unmaps the window of the shell specified in the *PopupShell* paramete.r
- Calls the **XtRemoveGrab** subroutine if the *grab_kind* field of the shell specified in the *PopupShell* parameter has the **XtGrabNonexclusive** or **XtGrabExclusive** value.
- Sets the *popped_up* field of the shell specified in the *PopupShell* parameter to the value of **False**.
- Calls the callback procedures on the *popdown_callback* field list of the shell.

Parameter

PopupShell	Specifies the widget shell to be popped down.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtPopup** subroutine, **XtCallbackPopdown** subroutine, **XtCallbackNonexclusive** subroutine, **XtCallbackExclusive** subroutine, **MenuPopdown** subroutine.

The **MenuPopdown** definition.

The **XtCallbackList** structure.

XtPopup Subroutine

Purpose

Maps a pop-up from within an application.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtPopup(*PopupShell*, *GrabKind***)**
Widget *PopupShell*;
XtGrabKind *GrabKind*;

Description

The **XtPopup** subroutine maps a pop-up from within an application. It does the following:

- Calls the **XtCheckSubclass** subroutine to ensure that the shell specified in the *PopupShell* parameter is a subclass of the **Shell** widget class
- Generates an error if the *popped_up* field of the shell already is the value of **True**.
- Calls the callback procedures on the *popup_callback* field list of the shell
- Sets the shell (specified in the *PopupShell* parameter) *popped_up* field to the value of **True**. the shell *spring_loaded* field to the value of **False**, and the shell *grab_kind* field from the *GrabKind* field.
- Calls the *CreatePopupChild* parameter of the specified shell, with **PopupShell** as the value for the parameter, if the *CreatePopupChild* parameter of the shell is not the value of **NULL**.
- Calls the following:

```
XtAddGrab(PopupShell,(GrabKind==XtGrabExclusive),False)
```

  If the *GrabKind* parameter has the **XtGrabNonexclusive** or **XtGrabExclusive** value.

- Calls the **XtRealizeWidget** subroutine with the widget specified in the *PopupShell* parameter specified
- Calls the **XMapWindow** subroutine with the widget specified in the *PopupShell* parameter specified.

Parameters

PopupShell	Specifies the widget shell.
GrabKind	Specifies how user events should be constrained.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtPopdown** subroutine, **XtCallbackNone** subroutine, **XtCallbackNonexclusive** subroutine, **XtCallbackExclusive** subroutine.

The **MenuPopup** subroutine definition.

XtProcessEvent Subroutine

Purpose

Processes events according to type.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtProcessEvent(*Mask***)**
XtInputMask *Mask*;

Description

The **XtProcessEvent** subroutine processes one input event, time-out, or alternate input source (depending on the value of the mask). The **XtProcessEvent** subroutine waits for an event of the specified type if one is not available.

The **XtInitialize** subroutine must be called before using the **XtProcessEvent** subroutine.

Note: This subroutine exists only as a convenience to users converting from earlier versions of the toolkit; it has been replaced by the **XtAppProcessEvent** subroutine.

Parameter

Mask Specifies the type of input to process.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppProcessEvent** subroutine.

XtQueryGeometry Subroutine

Purpose

Queries the preferred geometry of a child widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtGeometryResult XtQueryGeometry(*WidgetID*, *Intended*, *PreferredReturn*)
Widget *WidgetID*;
XtWidgetGeometry **Intended*, **PreferredReturn*;

Description

The **XtQueryGeometry** subroutine queries the preferred geometry of a child widget. This subroutine is called by the parent widget to set any changes in the geometry of the child widget. The **XtQueryGeometry** subroutine is called after the parent widget sets the desired changes in the corresponding fields of the **Intended** structure. This subroutine also sets the corresponding bits in the Intended *request_mode* field.

The **XtQueryGeometry** subroutine clears all the bits in the PreferredReturn->*request_mode* field and checks the *query_geometry* field of the specified widget's class record.

If the query_geometry field is not the value of **NULL**, this subroutine calls the procedure specified by it, passing as parameters the specified widget and the *Intended* and *PreferredReturn* parameters.

If the *Intended* parameter has the value of **NULL**, this subroutine replaces it with a pointer to an **XtWidgetGeometry** structure with its *request_mode* field set to the value of **0** before calling the procedure specified in the *query_geometry* field (the query_geometry procedure). The query_geometry procedure pointer is of the **XtGeometryHandler** type.

After calling the query_geometry procedure or if the *query_geometry* field has the value of **NULL**, the **XtQueryGeometry** subroutine examines all the unset bits in the PreferredReturn->*request_mode* parameter and sets the corresponding fields in *PreferredReturn* parameter to the current values from the widget instance. If the **CWStackMode** value is not set, the *stack_mode* field is set to **XtSMDontChange**.

The **XtQueryGeometry** subroutine returns the value returned by the query_geometry procedure or **XtGeometryYes** is the *query_geometry* field is the value of **NULL**.

Regardless of whether the caller chooses to ignore the return value or the reply mask, the reply structure is guaranteed to contain the complete geometry information for the child widget.

Parent widgets should call the **XtQueryGeometry** is subroutine in their layout routine, and wherever other information is significant after the *ChangeManaged* procedure is used.

Parameters

WidgetID	Specifies the widget.
Intended	Specifies changes the parent widget intends for the geometry of the child geometry.

PreferredReturn	Returns the preferred geometry of the child widget.

Return Values

XtGeometryAlmost	Indicates that at least one field in the *PreferredReturn* parameter is different from the corresponding field in the *Intended* parameter or that a bit was set in the *PreferredReturn* parameter that was not set in the Intended *request_mode* field. In other words, this return value indicates that both the parent widget and the child widget required at least one common field and the child widget requirement does not match the parent requirement or the child widget required a field that the parent may also require.
XtGeometryNo	Indicates that the preferred geometry is identical to the current geometry. In other words, the parent widget and the child required the same field and the child widget suggested the current value of the field as the preferred value.
XtGeometryYes	Indicates that the proposed geometry is acceptable without modification.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGeometryHandler** data type.

XtRealizeWidget Subroutine

Purpose

Realizes a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRealizeWidget(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtRealizeWidget** subroutine realizes a widget instance. If the widget is already realized, the **XtRealizeWidget** subroutine simply returns. If the widget is not realized, the subroutine performs the following:

- Binds all action names in the widget translation table to procedures.
- Makes a post-order traversal of the widget tree that is rooted at the specified widget, then calls the procedure specified in the *change_managed* field of each composite widget that has one or more managed child widgets.
- Constructs an **XSetWindowAttributes** structure filled in with information derived from the Core widget fields and calls the realize procedure for the widget, which adds any widget-specific attributes and creates the Enhanced X-Windows. While filling in the mask and corresponding **XSetWindowAttributes** structure, the **XtRealizeWidget** subroutine sets the following fields based on information in the Core widget structure:
  - The *background_pixmap* field (or the *background_pixel* field if the *background_pixmap* field is the value of **Null**) is filled in from the corresponding parameter in the core structure.
  - The *border_pixmap* field (or the *border_pixel* field if the *border_pixmap* field is the value of **Null**) is filled in from the corresponding parameter in the core structure.
  - The *event_mask* field is filled in based on the event handlers registered, the event translations specified, whether the *expose* field is not set to the value of **Null**, and whether the *visible_interest* field is set to the value of **True**.
  - The *bit_gravity* field is set to the **NorthWestGravity** value if the *expose* field is the value of **Null**.
  - The *do_not_propagate* mask is set to propagate all pointer and keyboard events anywhere inside it, including on top of children widgets, as long as children widgets do not specify a translation for the event.
- All other fields in the **XSetWindowAttributes** structure and their corresponding bits in the *value_mask* field can be set by the realize procedure.

- If the widget is not a subclass of the **compositeWidgetClass** class, the **XtRealizeWidget** subroutine returns. Otherwise, it continues and does the following:
  - Descends recursively to each of the managed child widgets of the widget and calls the realize procedures. Primitive widgets that instantiate children widgets are responsible for realizing those children widgets themselves.
  - Maps all of the managed children windows that have the *mapped_when_managed* field set to the value of **True**. If a widget is managed but the *mapped_when_managed* field is the value of **False**, the widget is allocated visual space but is not displayed.
- If the widget is the top-level shell widget, which by definition does not have a parent widget, and the *mapped_when_managed* field is the value of **True**, the **XtRealizeWidget** subroutine maps the widget window.

The **XtRealizeWidget** subroutine maintains the following invariants:

- If a widget is realized, then all its managed children widgets are realized.
- If a widget is realized, then all its managed children widgets that are also *mapped_when_managed* fields are mapped.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtIsRealized** subroutine.

XtRealloc Subroutine

Purpose

Changes the size of an allocated block of storage.

Library

Intrinsics Library (**libXt.a**)

Syntax

char *XtRealloc(*Pointer*, *Number***)**
char ****Pointer*;**
Cardinal *Number*;

Description

The **XtRealloc** subroutine changes the size of an allocated block of storage, sometimes moving it. Then, it copies the old contents (or as much as will fit) into the new block and frees the old block.

If there is insufficient memory to allocate the new block, the **XtRealloc** subroutine calls the **XtErrorMsg** subroutine.

If the *Pointer* parameter is the value of **NULL**, the **XtRealloc** subroutine calls the **XtMalloc** subroutine, which allocates the new storage without copying the old contents.

Parameters

Pointer	Specifies the pointer to old storage.
Number	Specifies the number of bytes required in new storage.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtMalloc** subroutine, **XtCalloc** subroutine, **XtFree** subroutine, **XtNew** subroutine, **XtNewString** subroutine.

XtRegisterCaseConverter Subroutine

Purpose

Registers a specified case converter.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRegisterCaseConverter(*DisplayPtr*, *Procedure*,
Start, *Stop*)
Display **DisplayPtr*;
XtCaseProc *Procedure*;
KeySym *Start*;
KeySym *Stop*;

Description

The **XtRegisterCaseConverter** subroutine registers a specified case converter. The *Start* and *Stop* parameters provide the inclusive range of key symbols for which this converter is to be called. The new converter overrides previous converters for key symbols in the specified range.

There is no interface to remove converters; an identity converter must be registered. When a new converter is registered, the Intrinsics liebrary refreshes the keyboard state if necessary. The default converter understands case conversion for all key symbols defined in the core protocol.

Parameters

DisplayPtr	Specifies the display providing the key events.
Procedure	Specifies the **XtCaseProc** procedure for the conversions.
Start	Specifies the first for which this converter is valid.
Stop	Specifies the last key symbol for which this converter is valid.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtConvertCase** subroutine.

The **XtCaseProc** data type.

XtReleaseGC Subroutine

Purpose

Deallocates a shared graphics context.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtReleaseGC(*WidgetID*, *GraphicsContext***);**
Widget *WidgetID*;
GC *GraphicsContext*;

Description

The **XtReleaseGC** subroutine deallocates a shared graphics context when it is no longer needed. References to shareable graphics contexts are counted and a free request is generated to the server when the last user of a specified graphics context destroys the graphics context.

Parameters

WidgetID	Specifies the widget.
GraphicsContext	Specifies the graphics context to be deallocated.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetGC** subroutine.

XtRemoveAllCallbacks Subroutine

Purpose

Deletes all callback procedures from a specified widget's callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveAllCallbacks(*WidgetID*, *CallbackName***)**
Widget *WidgetID*;
String *CallbackName*;

Description

The **XtRemoveAllCallbacks** subroutine deletes all callback procedures from a specified widget callback list and frees all storage associated with the callback list.

Parameters

WidgetID	Specifies the widget.
CallbackName	Specifies the callback list to be removed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddCallback** subroutine, **XtAddCallbacks** subroutine, **XtRemoveCallback** subroutine, **XtRemoveCallbacks** subroutine, **XtCallCallbacks** subroutine.

The **XtHasCallbacks** procedure.

The **XtCallbackProc** data type.

XtRemoveCallback Subroutine

Purpose

Deletes a callback procedure from a specified widget callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveCallback(*WidgetID*, *CallbackName*,
Callback, *ClientData*)
Widget *WidgetID*;
String *CallbackName*;
XtCallbackProc *Callback*;
caddr_t *ClientData*;

Description

The **XtRemoveCallback** subroutine deletes a callback procedure from a specified widget callback list only if both the procedure and the client data match.

Parameters

WidgetID	Specifies the widget.
CallbackName	Specifies the callback list from which the callback procedure will be removed.
Callback	Specifies the callback procedure.
ClientData	Specifies the client data to match on the registered callback procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddCallback** subroutine, **XtAddCallbacks** subroutine, **XtRemoveCallback** subroutine, **XtRemoveAllCallbacks** subroutine, **XtCallCallbacks** subroutine.

The **XtHasCallbacks** procedure.

The **XtCallbackProc** data type.

XtRemoveCallbacks Subroutine

Purpose

Deletes a list of callback procedures from a specified widget callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveCallbacks(*WidgetID*, *CallbackName*,
Callbacks)
Widget *WidgetID*;
String *CallbackName*;
XtCallbackList *Callbacks*;

Description

The **XtRemoveCallbacks** subroutine deletes a list of callback procedures from a specified widget callback list.

Parameters

WidgetID	Specifies the widget.
CallbackName	Specifies the callback list from which the callback procedures will be removed.
Callbacks	Specifies the null–terminated list of callback procedures and corresponding client data.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddCallback** subroutine, **XtAddCallbacks** subroutine, **XtRemoveCallback** subroutine, **XtRemoveAllCallbacks** subroutine, **XtCallCallbacks** subroutine.

The **XtHasCallbacks** procedure.

The **XtCallbackProc** data type.

XtRemoveEventHandler Subroutine

Purpose

Removes a previously registered event handler.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveEventHandler(*WidgetID*, *EventMask*,
Nonmaskable, *Procedure*,
ClientData)

Widget *WidgetID*;
XtEventMask *EventMask*;
Boolean *Nonmaskable*;
XtEventHandler *Procedure*;
caddr_t *ClientData*;

The **XtRemoveEventHandler** subroutine removes a previously registered event handler. It stops the specified procedure from receiving the specified events. The request is ignored if the value for the *ClientData* parameter does not match the value given in the call to the **XtAddEventHandler** subroutine.

If the widget is realized, the **XtRemoveEventHandler** subroutine calls the **XSelectInput** subroutine, if necessary. If the specified procedure has not been registered or if it has been registered with a different value for the *ClientData* parameter, the **XtRemoveEventHandler** subroutine returns without reporting an error.

To stop a procedure from receiving any events that entirely remove it from the widget event table, use the **XtAllEvents** subroutine with the *EventMask* parameter set to the **XtAllEvents** value and the *Nonmaskable* parameter set to the value of **True**.

Parameters

WidgetID	Specifies the widget to register for this procedure.
EventMask	Specifies the event mask to unregister for this procedure.
Nonmaskable	Specifies a Boolean value that indicates whether this procedure should be removed on the nonmaskable events. The nonmaskable events include the following: **ClientMessage** **SelectionClear** **GraphicsExpose** **SelectionNotify** **MappingNotify** **SelectionRequest** **NoExpose**
Procedure	Specifies the event handler procedure to be removed.
ClientData	Specifies the client data registered for this procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddEventHandler** subroutine, **XtAddRawEventHandler** subroutine, **XtBuildEventMask** subroutine.

The **XtEventHandler** data type.

XtRemoveGrab Subroutine

Purpose

Removes the redirection of user input to a modal widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveGrab(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtRemoveGrab** subroutine removes the redirection of user input to a modal widget. It removes widgets from the modal cascade starting at the most recent widget and up to and including the specified widget. If the specified widget is not on the modal cascade, it issues an error message.

Parameter

WidgetID Specifies the widget to remove from the modal cascade.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAddGrab** subroutine.

XtRemoveInput Subroutine

Purpose

Removes a source of input.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveInput(*ID***)**
XtInputId ****ID*;**

Description

The **XtRemoveInput** subroutine discontinues a source of input by causing the Intrinsics read subroutine to stop watching for input from the input source.

Parameter

ID	Specifies the ID returned from the corresponding **XtAppAddInput** subroutine.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddInput** subroutine.

The **XtInputCallbackProc** data type.

XtRemoveRawEventHandler Subroutine

Purpose

Removes a previously registered raw event handler.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveRawEventHandler(*WidgetID*, *Mask*,
Nonmaskable, *Procedure*,
ClientData)

Widget *WidgetID*;
EventMask *Mask*;
Boolean *Nonmaskable*;
XtEventHandler *Procedure*;
caddr_t *ClientData*;

Description

The **XtRemoveRawEventHandler** subroutine removes a previously registered raw event handler. It stops the specified procedure from receiving the specified events. This subroutine does not affect the widget masks or cause a call to the **XSelectInput** subroutine because it is a raw event handler.

Parameters

WidgetID	Specifies the widget to register for this procedure.
Mask	Specifies the event mask to unregister for this procedure.
Nonmaskable	Specifies a Boolean value that indicates whether this procedure should be removed on the nonmaskable events. The nonmaskable events include the following:

ClientMessage	**SelectionClear**
GraphicsExpose	**SelectionNotify**
MappingNotify	**SelectionRequest**
NoExpose	

Procedure	Specifies the event handler procedure to be registered.
ClientData	Specifies the client data registered.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRemoveEventHandler** subroutine, **XtAddRawEventHandler** subroutine, **XtAddEventHandler** subroutine, **XtBuildEventMask** subroutine, **XtRealizeWidget** subroutine.

XtRemoveTimeOut Subroutine

Purpose

Removes the time-out value.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveTimeOut(*Timer***)**
XtIntervalId *Timer*;

Description

The **XtRemoveTimeout** subroutine clears a time-out value by removing the time-out value. Time-out values are removed automatically once they are triggered.

Parameter

Timer Specifies the ID for the time-out request to be destroyed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddTimeOut** subroutine.

The **XtTimerCallbackProc** procedure.

XtRemoveWorkProc Subroutine

Purpose

Removes a background work procedure.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtRemoveWorkProc(*ID***)**
XtWorkProcId *ID*;

Description

The **XtRemoveWorkProc** subroutine removes the specified background work procedure.

An alternate method to remove a work procedure is to return to the value of **True** when it exits.

Parameter

ID Specifies the work procedure to remove.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtAppAddWorkProc** subroutine.

The **XtWorkProc** data type.

XtResizeWidget Subroutine

Purpose

Resizes a sibling widget of the child widget making the geometry request.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtResizeWidget(*WidgetID*, *Width*, *Height*,
BorderWidth)
Widget *WidgetID*;
Dimension *Width*;
Dimension *Height*;
Dimension *BorderWidth*;

Description

The **XtResizeWidget** subroutine resizes a sibling widget of the child widget making the geometry request. It returns immediately if the values for the *Width, Height,* and *BorderWidth* parameters are the same as the old values. Otherwise, it writes the new values into the widget.

If the widget is realized, the **XtResizeWidget** subroutine issues an **XConfigureWindow** call on the window of the widget.

If the value for the *Width* or *Height* parameters are different from the old values, the **XtResizeWidget** subroutine calls the resize procedure of the widget to notify it of the size change.

Parameters

WidgetID	Specifies the widget.
Width	Specifies the width of the new widget.
Height	Specifies the height of the new widget.
BorderWidth	Specifies the size of the new widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtResizeWindow** subroutine, **XtConfigureWidget** subroutine, **XtMoveWidget** subroutine.

XtResizeWindow Subroutine

Purpose

Resizes a child widget that already has the new values for its *width*, *height*, and *border_width* fields.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtResizeWindow(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtResizeWindow** subroutine resizes a child widget that already has the new values for its *width*, *height*, and *border_wldth* fields. It calls the **XConfigureWindow** subroutine to make the window of the specified widget match the values specified in its *width*, *height*, and *border_width* fields. The configure request is done unconditionally because there is no check on the values. The widget's resize procedure is not called. Use the **XtResizeWidget** subroutine under most normal conditions.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtMoveWidget** subroutine, **XtConfigureWidget** subroutine, **XtResizeWidget** subroutine.

XtScreen Macro

Purpose

Returns a pointer to the screen.

Library

Intrinsics Library (**libXt.a**)

Syntax

Screen *XtScreen(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtScreen** macro returns the screen pointer for the specified widget.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtParent** subroutine, **XtWindow** subroutine.

The **XtDisplay** macro.

XtSetArg Subroutine

Purpose

Sets values in an argument list.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtSetArg(*Argument*, *Name*, *Value*)
Arg *Argument*;
String *Name*;
XtArgVal *Value*;

Description

The **XtSetArg** subroutine sets values in an argument list. This subroutine is usually specified in a stylized manner to minimize the probability of making a mistake. Do not use auto-increment or auto-decrement within the first argument to the **XtSetArg** subroutine. The **XtSetArg** subroutine can be implemented as a macro that dereferences the first argument twice.

The **XtSetArg** subroutine is usually used in a highly stylized manner to minimize the probability of making a mistake. For example:

```
Arg args[20];
int n;

n=0;
XtSetArg(args[n], XtNheight, 100); n++;
XtSetArg(args[n], XtNwidth, 200); n++;
XtSetValues(widget,args,n);
```

An application could also declare the argument list and use the **XtNumber** subroutine. For example:

```
static Args args[]={
{XtNheight, (XtArgVal) 100},
{XtNwidth, (XtArgVal) 200},
};
XtSetValues(Widget, args, XtNumber(args));
```

Note: Do not use auto–increment or auto–decrement within the first argument to the **XtSetArg** subroutine. The **XtSetArg** subroutine can be implemented as a macro that de–references the first argument twice.

Parameters

Argument	Specifies the *Name-Value* parameter pair to set.
Name	Specifies the name of the resource.
Value	Specifies the value of the resource if less than or equal to the size of an **XtArgVal** structure. Otherwise, the *Value* parameter specifies the address of the resource.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtMergeArgLists** subroutine.

The **ArgList** data structure.

XtSetErrorHandler Subroutine

Purpose

Registers a procedure to call under fatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetErrorHandler(*Handler*)
XtErrorHandler *Handler*;

Description

The **XtSetErrorHandler** subroutine registers a procedure to call under fatal error conditions. The default error handler provided by the Intrinsics library is the **_XtError** error handler. On UNIX systems, it prints the message to standard error and ends the application.

Fatal error message handlers should not return. If one does return, subsequent toolkit behavior is undefined.

Note: This subroutine exists only as a convenience for users converting earlier versions of the toolkit.

Parameters

Handler Specifies the new fatal error procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetErrorDatabase** subroutine, **XtGetErrorDatabaseText** subroutine, **XtSetErrorMsgHandler** subroutine, **XtErrorMsg** subroutine, **XtSetWarningMsgHandler** subroutine, **XtWarningMsg** subroutine, **XtError** subroutine, **XtSetWarningHandler** subroutine, **XtWarning** subroutine.

XtSetErrorMsgHandler Subroutine

Purpose

Registers a procedure to call under fatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetErrorMsgHandler(*MessageHandler***)**
XtErrorMsgHandler *MessageHandler*;

Description

The **XtSetErrorMsgHandler** subroutine registers a procedure to call under fatal error conditions. The default error handler provided by the Intrinsics library constructs a string from the error resource database and calls the **XtError** subroutine.

Fatal error message handlers should not return. If one does return, subsequent toolkit behavior is undefined.

Note: This subroutine exists only as a convenience for users converting earlier versions of the toolkit.

Parameters

MessageHandler	Specifies the new fatal error procedure.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetErrorDatabase** subroutine, **XtGetErrorDatabaseText** subroutine, **XtSetErrorHandler** subroutine, **XtErrorMsg** subroutine, **XtSetWarningMsgHandler** subroutine, **XtWarningMsg** subroutine, **XtError** subroutine, **XtSetWarningHandler** subroutine, **XtWarning** subroutine.

XtSetKeyTranslator Subroutine

Purpose

Registers a key translator.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetKeyTranslator(*DisplayPtr*, *Procedure*)
Display **DisplayPtr*;
XtKeyProc **Procedure*;

Description

The **XtSetKeyTranslator** subroutine registers a key translator. Its sets the specified procedure as the current key translator. The default translator is the **XtTranslateKey** subroutine, an **XtKeyProc** procedure that uses the **Shift** and **Lock** modifiers with the interpretations defined by the core protocol.

The **XtSetKeyTranslator** subroutine is provided for new translators to obtain default **KeyCode**-to-**KeySym** translations and for reinstalling the default translator.

Parameters

DisplayPtr	Specifies the display from which to translate the events.
Procedure	Specifies the procedure that is to perform key translations.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtTranslateKeycode** subroutine, **XtRegisterCaseConverter** subroutine, **XtConvertCase** subroutine.

The **XtKeyProc** data type.

XtSetKeyboardFocus Subroutine

Purpose

Redirects keyboard input to a child widget of a composite widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtSetKeyboardFocus(*Subtree*, *Descendant*)
Widget *Subtree*, *Descendant*;

Description

The **XtSetKeyboardFocus** subroutine redirects keyboard input to a child widget of a composite widget without calling the **XSetInputFocus** subroutine.

If a future **KeyPress** or **KeyRelease** event occurs within the specified subtree, the **XtSetKeyboardFocus** subroutine causes the **XtDispatchEvent** subroutine to remap and send the event to the specified descendant widget.

When there is a modal cascade, keyboard events can occur within a widget in one of three ways:

- The widget has the X input focus.
- The widget has the keyboard focus of one of its ancestors, and the event occurs within the ancestor or one of the ancestor's descendants.
- No ancestor of the widget has a descendant within the keyboard focus, and the pointer is within the widget.

When there is a modal cascade, a widget receives keyboard events if an ancestor of the widget is in the active subset of the modal cascade and one or more of the previous conditions is the value of **True**.

When a subtree or one of its descendants acquires the X input focus or the pointer moves into the subtree such that keyboard events would now be delivered to the subtree, a **FocusIn** event is generated for the descendant in **FocusNotify** events have been selected by the descendant. Similarly, when the widget loses the X input focus or the keyboard focus for one of its ancestors, a **FocusOut** event is generated for the descendant if the **FocusNotify** events have been selected by the descendant.

Parameters

Subtree	Specifies the subtree of the hierarchy for which the keyboard focus is to be set.
Descendant	Specifies the widget in the subtree structure which will receive the keyboard event. Otherwise, this parameter should be the value of **None**.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCallAcceptFocus** subroutine, **XSetInputFocus** subroutine.

The **XtAcceptFocusProc** data type.

XtSetMappedWhenManaged Subroutine

Purpose

Changes the managed state of the specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetMappedWhenManaged(*WidgetID*, *MapWhenManaged*)
Widget *WidgetID*;
Boolean *MapWhenManaged*;

Description

The **XtSetMappedWhenManaged** subroutine changes the *map_when_managed* field of the specified widget.

- If the widget is realized and managed and the new value of the *map_when_managed* field is the value of **True**, the **XtSetMappedWhenManaged** subroutine maps the window.
- If the widget is realized and managed and the new value of the *map_when_managed* field is the value of **False**, the **XtSetMappedWhenManaged** subroutine unmaps the window.

As an alternative to the **XtSetMappedWhenManaged** subroutine to control mapping, a client can set the *map_when_managed* field to the value of **False** and use the **XtMapWidget** and **XtUnmapWidget** subroutines.

Parameters

WidgetID	Specifies the widget.
MapWhenManaged	Specifies a Boolean value which indicates the new value of this parameter.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtMapWidget** subroutine, **XtUnmapWidget** subroutine.

XtSetSelectionTimeout Subroutine

Purpose

Sets the value for the selection time-out.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetSelectionTimeout(*TimeOut***)**
unsigned long *TimeOut*;

Description

The **XtSetSelectionTimeout** subroutine sets the selection time-out. The selection time-out is the time during which two communicating applications must respond to one another. If the applications do not respond within this interval, the execution of the selection request is terminated. The default value of the selection time-out is 5 seconds.

Parameter

TimeOut Specifies the selection time-out in milliseconds.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetSelectionTimeout** subroutine.

XtSetSensitive Subroutine

Purpose

Sets the sensitivity state of a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetSensitive(*WidgetID*, *Sensitive***)**
Widget *WidgetID*;
Boolean *Sensitive*;

Description

The **XtSetSensitive** subroutine sets the sensitivity state of a widget in the following manner:

1. Calls the **XtSetValues** subroutine on the current widget with an argument list specifying that the *Sensitive* parameter should change to the new value.
2. Then, it recursively propagates the new value down the managed children tree by calling the **XtSetValues** subroutine on each child widget to set the *ancestor_sensitive* field to the new value if the new values for the *sensitive* field (of the current widget) and the *ancestor_sensitive* field (of the child widget) are not the same.

The **XtSetSensitive** subroutine calls the **XtSetValues** subroutine to change the *sensitive* and *ancestor_sensitive* fields. With these changes, the procedure specified in the *set_values* field of the widget takes the display actions necessary.

If the parent widget has the *sensitive* field or the *ancestor_sensitive* field set to the value of **False**, then all the children widgets also have the *ancestor_sensitive* field set to the value of **False**.

Parameters

WidgetID	Specifies the widget.
Sensitive	Specifies a Boolean value that indicates if the widget should receive keyboard and pointer events.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtIsSensitive** subroutine.

XtSetSubvalues Subroutine

Purpose

Sets the current value of a non-widget resource associated with a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetSubvalues(*BaseAddress*, *Resources*, *NumberResources*,
Arguments, *NumberArguments*)
caddr_t *BaseAddress*;
XtResourceList *Resources*;
Cardinal *NumberResources*;
ArgList *Arguments*;
Cardinal *NumberArguments*;

Description

The **XtSetSubvalues** subroutine sets the current value of a non-widget resource associated with a widget instance. It stores resources into the structure identified by the base address.

Parameters

BaseAddress	Specifies the base address of the subpart data structure where the resources should be written.
Resources	Specifies the current non-widget resource values.
NumberResources	Specifies the number of resources in the resource list.
Arguments	Specifies a variable length argument list of *Name-Value* parameter pairs that contain the resources to be modified and their new values. The resources and values used are dependent on the subpart of the widget being modified.
NumberArguments	Specifies the number of resources in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetValues** subroutine, **XtGetSubvalues** subroutine, **XtSetValues** subroutine.

The **XtSetValuesFunc** data type.

XtSetValues Subroutine

Purpose

Modifies the current value of a resource associated with a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetValues(*WidgetID*, *Arguments*, *NumberArguments*)
Widget *WidgetID*;
ArgList *Arguments*;
Cardinal *NumberArguments*;

Description

The **XtSetValues** subroutine modifies the current value of a resource associated with a widget instance. It starts with the resources specified for the **Core** widget fields and proceeds down the subclass chain to the widget. At each stage, it writes the new value (if specified by one of the arguments) or the existing value (if no new value is specified) to a new widget data record.

The **XtSetValues** subroutine then calls the procedures specified in the **set_values** field of the widget in superclass-to-subclass order. If the widget does not have any values of **NULL** for the *set_values_hook* fields, these are called immediately after the corresponding *set_values* field. This procedure permits subclasses to set non-widget data for the **XtSetValues** subroutine.

If the parent of the widget is a subclass of the **constraintWidgetClass**, the **XtSetValues** subroutine also updates the widget's constraints. It starts with the constraint resources specified for the **constraintWidgetClass** and proceeds down the subclass chain to the class of the parent. At each stage, the **XtSetValues** subroutine writes the new value or the existing value to a new constraint record. Then, it calls the constraint *set_values* field from the **constraintWidgetClass** down to the class of the parent. The constraint *set_values* field are called with widget parameters so that adjustments to the desired values can be made based on complete information about the widget.

The **XtSetValues** subroutine determines if a geometry request is needed by comparing the current widget to the new widget. If any geometry changes are required, the **XtSetValues** subroutine makes the request and the geometry manager returns one of the following values:

XtGeometryAlmost — Indicates that the **XtSetValues** subroutine calls the procedure specified in the **set_values_almost** procedure of the widget, which determines what should be done and writes new values for the geometry fields into the new widget.

Then, the **XtSetValues** subroutine repeats the process, deciding once more whether the geometry manager should be called.

XtGeometryNo — Indicates that the **XtSetValues** subroutine resets the geometry fields to their original values.

XtGeometryYes Indicates that the **XtSetValues** subroutine calls the procedure specified in the **resize** procedure of the widget.

Finally, if any of the *set_values* fields returned the value of **True**, the **XtSetValues** subroutine calls the **XClearArea** subroutine on the widget window.

Parameters

WidgetID Specifies the widget.

Arguments Specifies a variable-length argument list of *Name-Value* parameter pairs that contain the resources to be modified and their new values. The resources and values used are dependent on the widget being modified.

NumberArguments Specifies the number of resources in the argument list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetValues** subroutine, **XtGetSubValues** subroutine, **XtSetSubvalues** subroutine.

XtSetWarningHandler Subroutine

Purpose

Registers a procedure to be called under non-fatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetWarningHandler(*Handler***)**
XtErrorHandler *Handler*;

Description

The **XtSetWarningHandler** subroutine registers a procedure to be called under non-fatal error conditions.

The default warning handler is the **_XtWarning** subroutine, which, on UNIX based systems, prints the message to standard error and returns to the caller.

Note: This subroutine exists only as a convenience for users converting earlier versions of the toolkit.

Parameter

Handler Specifies the new non-fatal error procedure, which usually returns.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetErrorDatabase** subroutine, **XtGetErrorDatabaseText** subroutine, **XtSetErrorHandler** subroutine, **XtErrorMsg** subroutine, **XtSetWarningMsgHandler** subroutine, **XtWarningMsg** subroutine, **XtError** subroutine, **XtSetErrorMsgHandler** subroutine, **XtWarning** subroutine.

XtSetWarningMsgHandler Subroutine

Purpose

Registers a procedure to be called under non-fatal error conditions.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtSetWarningMsgHandler(*MessageHandler***)**
XtErrorMsgHandler *MessageHandler*;

Description

The **XtSetWarningMsgHandler** subroutine registers a procedure to be called under nonfatal error conditions. It constructs a string from the error resource database and calls the **XtWarning** subroutine.

Note: This subroutine exists only as a convenience for users converting earlier versions of the toolkit.

Parameter

MessageHandler Specifies the new non-fatal error procedure, which usually returns.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetErrorDatabase** subroutine, **XtGetErrorDatabaseText** subroutine, **XtSetErrorHandler** subroutine, **XtErrorMsg** subroutine, **XtSetWarningHandler** subroutine, **XtWarningMsg** subroutine, **XtError** subroutine, **XtSetErrorMsgHandler** subroutine, **XtWarning** subroutine.

XtStringConversionWarning Subroutine

Purpose

Issues a warning message.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtStringConversionWarning(*Source*, *DestinationType*)
String *Source*, *DestinationType*;

Description

The **XtStringConversionWarning** subroutine is a convenience routine for new resource converters that convert from strings. The **XtStringConversionWarning** subroutine issues a warning message with name "conversionError," type "string," class "XtToolkitError," and the default message string "Cannot convert *Source* to type *DestinationType.*"

Parameters

DestinationType	Specifies the string that could not be converted.
Source	Specifies the name of the type to which the string could not be converted.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtSuperclass Macro

Purpose

Obtains the superclass of a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

WidgetClass XtSuperclass(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtSuperclass** macro obtains the superclass of a widget by returning a pointer to the superclass structure of the widget.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtClass** subroutine, **XtIsSubclass** subroutine, **XtCheckSubclass** subroutine.

XtToolkitInitialize Subroutine

Purpose

Initializes the X toolkit internals.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtToolkitInitialize();

Description

The **XtToolkitInitialize** subroutine initializes the X toolkit internals. If this routine is called more than once, the behavior of the toolkit is undefined.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCreateApplicationContext** subroutine, **XtDestroyApplicationContext** subroutine, **XtWidgetToApplicationContext** subroutine, **XtDisplayInitialize** subroutine, **XtOpenDisplay** subroutine, **XtCloseDisplay** subroutine.

XtTranslateCoords Subroutine

Purpose

Translates an x, y coordinate pair from widget coordinates to root coordinates.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtTranslateCoords(*WidgetID*, *X*, *Y*,
RootXReturn,
RootYReturn)
Widget *WidgetID*;
Position *X*, *Y*;
Position **RootXReturn*, **RootYReturn*;

Description

The **XtTranslateCoords** subroutine translates an x, y coordinate pair from widget coordinates to root coordinates. It does not generate a server request because the required information is already in the data structure of the widget.

Parameters

WidgetID	Specifies the widget.
X	Specifies the x coordinate of the widget.
Y	Specifies the y coordinate of the widget.
RootXReturn	Returns the x coordinate of the root.
RootYReturn	Returns the y coordinate of the root.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtTranslateKeycode Subroutine

Purpose

Calls the currently registered **key_code**-to-**key_sym** translator.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtTranslateKeycode(*DisplayPtr*, *Keycode*,
ModifiersMask,
ModifiersMaskReturn,
KeySymReturn)
Display **DisplayPtr*;
KeyCode *Keycode*;
Modifiers *ModifiersMask*;
Modifiers **ModifiersMaskReturn*;
KeySym **KeySymReturn*;

Description

The **XtTranslateKeycode** subroutine invokes the **key_code**-to-**key_sym** translator that is registered currently. It passes the specified arguments directly to the translator.

Parameters

DisplayPtr	Specifies the display that the key code is from.
Keycode	Specifies the key code to translate.
ModifiersMask	Specifies the modifiers to the key code.
ModifiersMaskReturn	Returns a mask that indicates the modifiers actually used to generate the key symbol.
KeySymReturn	Returns the resulting key symbol.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRegisterCaseConverter** subroutine, **XtSetKeyTranslator** subroutine.

The **XtKeyProc** data type.

XtUngrabKey Subroutine

Purpose

Cancels a passive grab on a key combination.

Library

Intrinsics Library (**libXt.a**)

Syntax

#include <X11/PassivGrab.h>

void XtUngrabKey(*WidgetID*, *Keycode*, *Modifiers***)**
Widget *WidgetID*;
int *Keycode*;
unsigned int *Modifiers*;

Description

The **XtUngrabKey** subroutine cancels the passive grab on the key combination on the specified widget and allows the client application to redirect the specified key event to the root widget of a hierarchy.

Parameters

WidgetID	Specifies the root widget to the **XtUngrabKey** call.
Keycode	Specifies the key code. This maps to the specific key to be grabbed.
Modifiers	Specifies the set of key masks. This mask is the bitwise-inclusive OR of these key mask bits: the **ShiftMask**, **LockMask**, **ControlMask**, **Mod1Mask**, **Mod2Mask**, **Mod3Mask**, **Mod4Mask**, and **Mod5Mask** bits. You can also pass the **AnyModifier** bit, which is equivalent to issuing the ungrab key request for all possible modifier combinations, including the combination of no modifiers.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGrabKey** subroutine.

XtUngrabKeyboard Subroutine

Purpose

Releases any active grab on the keyboard.

Library

Intrinsics Library (**libXt.a**)

Syntax

#include <X11/PassivGrab.h>

void XtUngrabKeyboard(*WidgetID*, *TimeStamp*)
Widget *WidgetID*;
Time *TimeStamp*;

Description

The **XtUngrabKeyboard** subroutine releases any active grab on the keyboard.

Parameters

WidgetID	Specifies the root widget to the **XtUngrabKeyboard** call.
TimeStamp	Specifies the time. Either a time stamp, expressed in milliseconds can be passed, or the **CurrentTime** value can be passed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGrabKeyboard** subroutine.

XtUninstallTranslations Subroutine

Purpose

Removes existing translations.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtUninstallTranslations(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtUninstallTranslations** subroutine causes the entire translation table for the specified widget to be removed.

Parameter

WidgetID Specifies the widget from which the translations are to be removed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtParseTranslationTable** subroutine, **XtAugmentTranslations** subroutine, **XtOverrideTranslations** subroutine.

XtUnmanageChild Subroutine

Purpose

Removes a single child widget from the managed set of its parent widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtUnmanageChild(*Child***)**
Widget *Child*;

Description

The **XtUnmanageChild** subroutine removes a single child widget from the managed set of its parent widget. It constructs a widget list with a length of 1 and calls the **XtUnmanageChildren** subroutine.

Parameter

Child Specifies the child widget to be removed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtCreateManagedWidget** subroutine, **XtIsManaged** subroutine, **XtManageChildren** subroutine, **XtUnmanageChildren** subroutine.

XtUnmanageChildren Subroutine

Purpose

Removes a list of child widgets from the managed list of the parent widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtUnmanageChildren(*Children*, *NumberChildren*)
WidgetList *Children*;
Cardinal *NumberChildren*;

Description

The **XtUnmanageChildren** subroutine removes a list of child widgets from the managed list of the parent widget, but does not destroy the children widgets. The **XtUnmanageChildren** subroutine:

- Issues an error if all the children widgets do not have the same parent widget or if the parent widget is not a subclass of the **compositeWidgetClass**.
- Returns immediately if the common parent widget is being destroyed.

For each unique child widget on the list, the **XtUnmanageChildren** subroutine does the following:

- Ignores the child widget if the child widget is already unmanaged or is being destroyed. Otherwise, this subroutine marks the child widget.
- Makes the child widget non–visible by unmapping it if it is realized.
- Calls the subroutine specified in the **changed_managed** field of the parent widget after all children widgets have been marked if the parent widget is realized.

Parameters

Children	Specifies the list of child widgets to be removed.
NumberChildren	Specifies the number of the child widgets to be removed.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtUnmanageChild** subroutine, **XtCreateManagedChildren** subroutine, **XtIsManaged** subroutine, **XtManageChildren** subroutine.

XtUnmapWidget Subroutine

Purpose

Unmaps a widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtUnmapWidget(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtUnmapWidget** subroutine unmaps a widget explicitly.

Parameter

WidgetID Specifies the widget to unmap.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtMapWidget** subroutine, **XtSetMappedWhenManaged** subroutine.

XtUnrealizeWidget Subroutine

Purpose

Destroys the windows associated with a widget and its descendants.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtUnrealizeWidget(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtUnrealizeWidget** subroutine destroys the windows associated with a widget and its descendants (recursively down the widget tree). To recreate the windows at a later time, call the **XtRealizeWidget** subroutine again.

If the widget was managed, it will be unmanaged automatically before its window is freed.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtRealizeWidget** subroutine.

XtWarning Subroutine

Purpose

Calls the installed non-fatal error procedure.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtWarning(*Message***)**
String *Message*;

Description

The **XtWarning** subroutine calls the installed non-fatal error procedure.

Note: This subroutine exists only as a convenience for users converting earlier versions of the toolkit.

Parameter

Message	Specifies the non-fatal error message to report.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetErrorDatabase** subroutine, **XtGetErrorDatabaseText** subroutine, **XtSetErrorHandler** subroutine, **XtErrorMsg** subroutine, **XtSetWarningHandler** subroutine, **XtWarningMsg** subroutine, **XtError** subroutine, **XtSetErrorMsgHandler** subroutine, **XtSetWarningMsgHandler** subroutine.

XtWarningMsg Subroutine

Purpose

Is the high-level warning handler.

Library

Intrinsics Library (**libXt.a**)

Syntax

void XtWarningMsg(*Name*, *Type*, *Class*,
Default, *Parameters*,
NumberParameters)
String *Name*;
String *Type*;
String *Class*;
String *Default*;
String **Parameters*;
Cardinal **NumberParameters*;

Description

The **XtWarningMsg** subroutine displays messages based on input parameters.

Note: This subroutine exists only as a convenience for users converting earlier versions of the toolkit.

Parameters

Name	Specifies the general kind of error.
Type	Specifies the detailed name of the error.
Class	Specifies the resource class. This parameter has the **XtToolkitError** value for the Intrinsics internal warnings.
Default	Specifies the default message to use if an error database entry is not found.
Parameters	Specifies a pointer to a list of values to be stored in the message.
NumberParameters	Specifies the number of values in the parameter list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtGetErrorDatabase** subroutine, **XtGetErrorDatabaseText** subroutine, **XtSetErrorHandler** subroutine, **XtErrorMsg** subroutine, **XtSetWarningHandler** subroutine, **XtWarning** subroutine, **XtError** subroutine, **XtSetErrorMsgHandler** subroutine, **XtSetWarningMsgHandler** subroutine.

XtWidgetCallCallbacks Subroutine

Purpose

Calls the entries on a callback list.

Library

Intrinsics Library (**libXt.a**)

Syntax

#include <Xm/Xm.h>

void XtWidgetCallCallbacks(*Callbacks*, *CallData*)
XtCallbackList *Callbacks*;
Opaque *CallData*;

Description

The **XtWidgetCallCallbacks** subroutine calls the entries on a callback list. The widget knows the address of the callback list and avoids extra processing by using this function. The external version of this routine is the **XtCallCallbacks** subroutine.

Parameters

Callbacks	Specifies the callback list to execute.
CallData	Specifies a callback list specific data value to pass to each of the callback procedures in the list.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

XtWidgetToApplicationContext Subroutine

Purpose

Gets the application context for a specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

XtAppContext XtWidgetToApplicationContext(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtWidgetToApplicationContext** subroutine gets and returns the application context for a specified widget.

Parameter

WidgetID Specifies the widget with the desired application context.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtToolkitInitialize** subroutine, **XtCreateApplicationContext** subroutine, **XtDestroyApplicationContext** subroutine, **XtDisplayInitialize** subroutine, **XtOpenDisplay** subroutine.

XtWindow Macro

Purpose

Returns the window of the specified widget.

Library

Intrinsics Library (**libXt.a**)

Syntax

Window XtWindow(*WidgetID***)**
Widget *WidgetID*;

Description

The **XtWindow** macro returns the window of the specified widget.

Parameter

WidgetID Specifies the widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Related Information

The **XtDisplay** subroutine, **XtParent** subroutine, **XtScreen** subroutine.

XtWindowToWidget Subroutine

Purpose

Translates a window and display pointer into a widget instance.

Library

Intrinsics Library (**libXt.a**)

Syntax

Widget XtWindowToWidget(*DisplayPtr*, *WindowID*)
Display **DisplayPtr*;
Window *WindowID*;

Description

The **XtWindowToWidget** subroutine translates a window and display pointer into a widget instance.

Parameters

DisplayPtr Specifies the display on which the window is defined.

WindowID Specifies the window for the desired widget.

Implementation Specifics

This toolkit subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

_XAllocScratch Extension Subroutine

Purpose

Returns a scratch buffer.

Library

Enhanced X-Windows Library (**libXext.a**)

Syntax

char ***_XAllocScratch**(*DisplayID*, *NumberBytes*)
Display **DisplayID*;
unsigned long *NumberBytes*;

Description

The **_XAllocScratch** extension subroutine returns a scratch buffer. If you need a single scratch buffer inside a critical section to pack and unpack data to and from wire protocol, the general memory allocators may be too expensive to use (particularly in output routines, which are performance critical). Use of a routine with this storage must only be used inside the critical section of your stub.

Parameters

DisplayID	Specifies the ID of the display.
NumberBytes	Specifies the length, in bytes, of the buffer to be allocated.

Implementation Specifics

This extension subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

_XReply Extension Subroutine

Purpose

Flushes the output buffer, waits for a reply packet, and copies the contents into the specified *Reply* parameter.

Library

Enhanced X-Windows Library (**libXext.a**)

Syntax

Status _XReply(*DisplayPtr*, *Reply*, *Extra*, *Discard***)**
**Display ***DisplayPtr*:
**xReply ***Reply*:
int *Extra*:
Bool *Discard*;

Description

The **_XReply** extension subroutine flushes the output buffer, waits for a reply packet, and copies the contents into the specified *Reply* parameter. If other events arrive during this time, the **_XReply** extension subroutine queues these events for later use. This extensionhandles error and event packets that occur before the reply is received.

Most reply structures are 32 bytes long; therefore, the *Extra* parameter is usually the the value of 0. In the Core protocol, only the following are longer than 32 bytes: the **GetWindowAttributes**, **QueryFont**, **QueryKeymap**, and **GetKeyboardControl** reply structures.

If a reply is not followed by variable-length data, use the **_XReply** extension subroutine as follows:

```
_XReply (Display, (xReply *)&Rep, 0, xTrue);
*ret1 = rep.ret1;
*ret2 = rep.ret2;
*ret3 = rep.ret3;
UnlockDisplay(dpy);
SyncHandle();
return (rep.ret4);}
```

If a reply has variable-length data, change the **xTrue** value to the **xFalse** value and use the **_XRead** extension subroutine to read the variable-length data.

Parameters

Discard Specifies a Boolean value that tells the **_XReply** extension subroutine to discard any additional bytes beyond those it was told to read. The *Discard* parameter can be one of the following values:

xFalse	The reply structure is followed by additional variable-length data, such as a list or string.
xTrue	There is no variable-length data.

DisplayPtr Specifies the display structure.

Extra Specifies the number of additional bytes (beyond `sizeof(xReply) = 32bytes`) in the reply structure. This is the number of words expected after the reply.

Reply Specifies a pointer to the parameter structure indicated by the **xReply** subroutine.

Return Values

In this case, the **_XReply** extension returns one of the following values:

True A reply was received successfully.

False The reply was not successful. An **XError** message accompanies this value.

Implementation Specifics

This extension subroutine is part of AIXwindows Run Time Environment in AIXwindows Environment/6000.

Index

Symbols

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Reader's Comment Form

AIX Calls and Subroutines Reference: User Interface for IBM RISC System/6000

SC23–2198–00

Please use this form only to identify publication errors or to request changes in publications. Your comments assist us in improving our publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page	Comments

Please contact your IBM representative or your IBM–approved remarketer to request additional publications.

Please print

Date _____

Your Name _____

Company Name _____

Mailing Address _____

_____

_____

Phone No. ( ) _____
Area Code

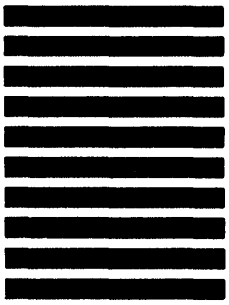
No postage necessary if mailed in the U.S.A

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE



International Business Machines Corporation
Department 997, Building 997
11400 Burnet Rd.
Austin, Texas 78758–3493

Fold

Fold

Cut or Fold Along Line

Fold and Tape

Please Do Not Staple

Fold and Tape

International Business Machines
Corporation
11400 Burnet Road
Austin, Texas 78758–3493

Printed in the
United States of America

SC23-2198-00

SC23-2198-00