UIM/X
Reference Manual

Integrated Computer
Solutions Incorporated

Integrated Computer Solutions, Inc.

54 Middlesex Turnpike, Bedford, MA 01730

Tel: 617.621.0060

Fax: 617.621.9555

E-mail: info@ics.com

WWW: http://www.ics.com

Printed in the United States of America.

October 2005

Preface

Overview

The Ux Convenience Library is a set of convenience functions for displaying and managing interfaces generated by UIM/X.

Swidget methods are the methods supported by the Connection Editor for each of the UIM/X swidgets.

Who Should Use this Manual

This manual is intended for programmers who want to create applications using UIM/X. It assumes the reader is familiar with the X Window system, Motif, and the issues involved in Graphical User Interface (GUI) development.

Related Documents

For more information on UIM/X, see the following documents, available at http://www.ics.com/support/docs/:

- *UIM/X Installation Guide*. Explains how to install and run UIM/X. Includes information on the files provided with UIM/X, backwards compatibility issues, and compiler considerations.
- *UIM/X Beginner's Guide*. Introduces UIM/X by presenting Novice Mode, the simplified Palette that enables new users to be productive immediately. Includes information on a number of important features for creating, testing and running applications.
- *UIM/X Tutorial Guide***.** A series of step-by-step tutorials, teaching tools and techniques that will greatly assist you in developing your own applications. Features tutorials in Novice Mode, Standard Mode, and on advanced topics.
- *UIM/X User's Guide*. Explores the UIM/X features common to both Motif and cross-platform development. Includes discussions of how to use UIM/X's editors to set properties, add behavior, etc.
- *UIM/X Motif Developer's Guide*. An in-depth guide to the widgets, features, and capabilities of UIM/X as they relate specifically to Motif development.
- *UIM/X Advanced Topics***.** Describes how to customize UIM/X, including integrating new widget and component classes into the executable. Includes reference information of an advanced technical nature.

For more information on designing user interfaces, see any of the following:

- *OSF/Motif Style Guide release 1.2* (Prentice Hall, 1993, ISBN 0-13-643123-2)
- *Visual Design with OSF/Motif* (by Shiz Kobara, Addison-Wesley, 1991, ISBN 0-201-56320-7)
- *The Windows Interface Guidelines for Software Design: An Application Design Guide* (Microsoft Corporation, 1995, ISBN 1-55615-679-0)
- *Human Interface Guidelines: The Apple Desktop Interface* (Addison-Wesley, 1987, ISBN 0-201-17753-6)

How This Manual Is Organized

This document comprises two chapters and an index, organized as follows:

- *Chapter 1*, "Ux Convenience Library," contains the reference pages for the convenience functions provided with UIM/X Ux Convenience Library swidgets.
- *Chapter 2*, "Swidget Methods," contains the reference pages for the methods supported by the Connection Editor for each of the UIM/X widgets.

The reference pages are presented in alphabetical order. Each reference page contains a synopsis, describes the associated arguments and return values, explains the features and usage of the function or method, and provides a list of related topics and reference pages.

Conventions Used in this Guide

Unless otherwise noted in the text, we use the following symbolic conventions:

Typeface or Symbol	**Meaning**
literal names	Bold words or characters in command descriptions represent words or values that you must use literally.
user-supplied values	Italic words or characters in command descriptions represent values that you must supply. Italic words in text also indicate the first use of a new term, or emphasis
`sample user input`	In interactive examples, information that you must enter appears in **`this typeface.`**

Typeface or Symbol	**Meaning**
`output/source code`	Information that the system displays appears in `this typeface.`
...	Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

Contents

Ux Convenience Library 1

Overview

The Ux Convenience Library provides a set of functions for displaying and managing interfaces generated by UIM/X. These convenience functions allow you to avoid some of the details associated with the Motif and Xt Intrinsics libraries. You can use Ux Convenience Library functions anywhere you can write code in UIM/X.

This chapter summarizes each function in alphabetical order. Most include a see also reference to related Ux functions. For more information on related Xm or Xt functions, consult the appropriate manual.

During design time, the objects UIM/X manipulates are swidgets. The widget's name is declared as a variable name of C type *`swidget`* (shadow widget); the swidget being a pointer to an opaque C structure. It maintains necessary state information about the corresponding widget and provides a more complete error checking and handling mechanism.

Note: When C++ bindings are enabled, Motif objects are not declared as type `swidget`, but instead are declared as objects of the appropriate Ux Convenience Library C++ wrapper class. This allows such objects to be manipulated with conventional C++ syntax. However, such objects can also be used in any context where a swidget is expected, since these objects can be implicitly converted to type `swidget`.

Ux Convenience Library functions generally use swidgets (not widgets) as one of their Arguments.

The *`UxGetWidget()`* function is available for retrieving the Xt widget pointer from a swidget for use in Xt functions.

When code is generated as Ux code, UIM/X includes calls to the Ux Convenience Library. For some Ux functions there is an equivalent Xt function, though Ux functions provide better error checking.

When the code is generated as Xt code, the Convenience Library is unavailable. Some Ux functions are available in the *UxXt.c* file. The include files *UxXtGets.h* and *UxXtPuts.h*, which reside in the uimx_directory*/contrib/XtCodePuts* subdirectory, offer as macros some of the otherwise unavailable *UxGet* and *UxPut* functions. These functions are the only Ux functions available when Xt code is generated.

Ux Convenience Library Cross Reference

Callbacks

UxAddCallback()	Adds a callback procedure to a callback list.

Event Handling

UxDispatchEvent()	Dispatches an X event.
UxMainLoop()	Enters the main event loop for the application.
UxNextEvent()	Returns the next X event form the event queue.
UxNotify()	Sets the notify flag to break from UxWaitForNotify() event loop.
UxWaitForNotify()	Executes an event loop until the notify flag is set using UxNotify

Initialization

UxAppInitialize()	Initializes user application.

Properties

UxDelayUpdate()	Suspends property updates so calls to UxPutProperty() does not immediately update the widget.
UxGet*Property*()	Retrieves property values.
UxPut*Property*()	Sets property values.
UxUpdate()	Updates the widget only once after all of the desired changes have been made.

Resource Management

UxGetAppDefault()	Queries the X resource database for a resource string of the form App-class.resource_string or argv[0].resource_string.
UxGetAppResource()	Queries the X resource database for a resource string of the form App-class.resource_string or argv[0].resource_string or argv[0].
UxGetDefault()	Queries the X resource database for a resource string of the form App-class.resource_string or program_name.resource_string.
UxGetResource()	Queries the X resource database for a resource string of the form App-class.resource_string or program_name.resource_string.
UxLoadResources()	Loads a resource file into the current resource database.
UxOverrideResources()	Loads a resource file into the current database.
UxGetAppDefault()	Queries the X resource database for a resource string of the form App-class.resource_string or argv[0].resource_string.
UxGetAppResource()	Queries the X resource database for a resource string of the form App-class.resource_string or argv[0].resource_string or argv[0].
UxGetDefault()	Queries the X resource database for a resource string of the form App-class.resource_string or program_name.resource_string.
UxGetResource()	Queries the X resource database for a resource string of the form App-class.resource_string or program_name.resource_string.
UxLoadResources()	Loads a resource file into the current resource database.
UxOverrideResources()	Loads a resource file into the current database.

Search Paths

UxAddPath()	Adds the directories in a string to the list of directories in the path list structure.
UxExpandBitmapFilename()	Expands a bitmap file name to its full path name.
UxExpandEnv()	Expands all environment variable references in a string recursively.

UxExpandFilename()	Expands a file name to its full path name if it is found in any of the directories in the path list structure.
UxExpandResourceFilename()	Expands a resource file name to its full path name.
UxFileExists()	Determines whether a file name exists.
UxFreePath()	Frees the memory used by a path list.
UxGetPath()	Returns the list of directories stored by a path list.
UxResetPath()	Replaces the list of directories stored in the a path list structure.

Subprocess Control

UxAppendTo()	An output handler that appends output from a subprocess to a text widget specified by UxSetSubprocClosure().
UxCreateSubproc()	Creates a subprocess object for running a particular subprocess command.
UxDelayedDeleteSubproc()	Sets up the subprocess associated with the passed handle for a delayed deletion.
UxDeleteSubproc()	Terminates a subprocess and deletes all data associated with its execution.
UxExecSubproc()	Executes the subprocess created by UxCreateSubproc().
UxExitSubproc()	Terminates a running subprocess, but maintains necessary data so that the process can be restarted by calling UxRunSubproc() or UxExecSubproc().
UxGetSubprocPid()	Checks if a subprocess associated with a given handle is running and returns its process id.
UxRunSubproc()	Executes a subprocess that was originally created by UxCreateSubproc().
UxSendSubproc() and UxSendSubprocNoCR()	Send a command string to the subprocess.
UxSetSubprocClosure()	The UxSetSubprocClosure() function is used to specify data that is to be passed to the output handler function for a given subprocess.
UxSetSubprocEcho()	Turns echoing of input on or off for the given subprocess handle.
UxSetSubprocExitCallback()	Specifies a function to be called when a subprocess is terminated or stopped.
UxSetSubprocFunction()	Specifies a function to be used to handle output from a subprocess.
UxTransferToBuffer()	Reads a buffer containing output from a subprocess.

Translations and Actions

UxAddActions()	Registers the specified action table.

Widget Lifecycle

UxCreateShadowWidget()	Creates a swidget.
UxCreateWidget()	Creates a widget associated with the specified swidget.
UxDestroyInterface()	Destroys the widgets in an interface.
UxDestroySwidget()	Deletes the specified swidget and its associated widget.
UxManage()	Manages the swidget passed to it.
UxMap()	Causes the specified swidget to re-appear on the screen.
UxPopdownInterface()	Pops down the interface associated with the specified swidget.
UxPopupInterface()	Pops up the interface associated with the specified swidget.
UxPostMenu()	Pops up a menu on the specified widget at the cursor position.
UxRealizeInterface()	Realizes all X widgets in the interface associated with the swidget.
UxUnmanage()	Unmanages the swidget passed to it.
UxUnmap()	Causes the specified swidget to disappear from the screen.

Utility Functions

UxCenterVisibly() and UxCenterWidgetVisibly()	Center the specified swidget under the pointer at its current location.
UxFindSwidget()	Returns the swidget handle of the named widget.
UxGetClass()	Returns the Motif widget class for a given swidget.
UxGetContext()	Returns a pointer to the current context of the specified swidget.
UxGetName()	Returns the name of the specified swidget.
UxGetParent()	Gets the parent swidget of the specified swidget.
UxGetWidget()	Returns the widget pointer for the specified swidget.
UxIsValidSwidget()	Checks the validity of the specified swidget.
UxNameToSwidget()	Searches for a swidget by name.
UxPutContext()	Uses a pointer to set the context of the specified swidget to the values in the context structure provided.
UxShellWidget()	Returns the widget ID for the shell widget of an interface.
UxTextAppend()	Appends a string to a text widget.
UxWidgetToSwidget()	Gets the swidget associated with the specified widget.

UxAddActions()

Function

Registers the specified action_table.

Synopsis

```
#include <UxLib.h>

void UxAddActions(XtActionList action_table,
   Cardinal number_of_actions);
```

Return Value

None.

Description

The UxAddActions() function registers the specified `action_table.` The UxAddActions() function passes its Arguments directly to XtAddActions(). The `number_of_actions` argument specifies the number of action entries recorded in the table.

See Also

UxAddCallback() , XtAddActions(), XtAppAddActions()

UxAddCallback()

Function Adds the callback procedure to the callback list.

Synopsis

```
#include <UxLib.h>

int UxAddCallback(swidget shadow_widget, char *name,
   void *procedure, void *client_data);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure.

Description The UxAddCallback() function adds the callback procedure to the callback list. The callback procedure should be written using the standard form for Xt callbacks. This function is the same as the Xt function XtAddCallback() except that its first parameter is a swidget instead of a widget.

See Also UxAddActions(), XtAddCallback()

UxAddPath()

Function Adds a new path to a list of search paths.

Synopsis

```
#include <pathlist.h>

#include <resload.h>

void UxAddPath(pathlist path, char *string);
```

Return Value None.

Description UxAddPath() adds the directories in `string` to the list of directories in `pathlist`. The directory names in `string` can be separated by spaces, colons, commas, newlines, or tabs.

You can also include the values of environment variables like `$XFILESEARCHPATH` in `string`:

```
UxAddPath(mySearchPath, "$XFILESEARCHPATH");
```

UIM/X provides two built-in path list variables: UxBitmapPath, which lists the paths to search for bitmaps, and UxResourcePath, which lists the paths to search for resource files. To use these global path list variables in your code, you must include `<resload.h>`.

Example Suppose you want your application to look for bitmaps in non-standard directories. To do this, you use the Program Layout Editor to add a call to `UxAddPath()` in the main program:

```
#include <resload.h>
   UxTopLevel=XtAppInitialize(&UxAppContext,
   "$PJ_APP_CLASS_NAME",NULL, 0, &argc, argv, NULL,
   NULL, 0);

   UxAppInitialize("$PJ_APP_CLASS_NAME", &argc,
      argv);

/*-----------------------------------------------

* Insert initialization code for your application
   here

*---------------------------------------------*/

UxAddPath( UxBitmapPath, "/usr/patrice/bitmaps" );
```

First, you include `<resload.h>` for the declarations of the global variables `UxBitmapPath and UxResourcePath`. Then you insert a call to `UxAddPath()` *after* the call to `UxAppInitialize()`, because `UxAppInitialize()` initializes both `UxBitmapPath` and `UxResourcePath`.

See Also

UxExpandBitmapFilename(), UxExpandEnv(), UxExpandFilename(), UxExpandResourceFilename(), UxFileExists(), UxFreePath(), UxGetPath(), UxInitPath()for an example of how to replace the default search paths in UxBitmapPath with your own search path, UxResetPath()

UxAddTabGroup()

This function is obsolete. Its behavior is replaced by setting the Specific resource Navigation Type to `exclusive_tab_group` via the Property Editor.

UxAppendTo()

Function Appends output from a subprocess to a text widget.

Synopsis

```
#include <UxSubproc.h>

void UxAppendTo(int file_descriptor, Widget
   text_widget);
```

Return Value None.

Description `UxAppendTo()` is an output handler that appends output from a subprocess to a text widget specified by `UxSetSubprocClosure()`. The `file_descriptor` parameter identifies the file descriptor for the subprocess output stream. If `text_widget` is `NULL`, then the output is sent to `stdout`.

Example The `UxAppendTo()` output handler is used in the example listed with the `UxCreateSubproc()` function.

See Also UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxAppInitialize()

Function Initializes a user application.

Synopsis

```
#include <UxLib.h>

void UxAppInitialize(char *App-class, int *argc,
   char **argv);
```

Return Value None.

Description The `UxAppInitialize()` function initializes UIM/X. This function must be called after the calls to `UxPreInitialize()` and `XtAppInitialize()`. The `App-class` argument is the name of a resource file in `/usr/lib/X11/app-defaults` where the application class resources are set.

Note: `UxAppInitialize()` and `UxPreInitialize()` now supercede `UxInitialize()` and `UxOptionInitialize()`, although the latter functions are still supported.

See Also UxGetDefault(), UxGetResource(), UxPreInitialize(), XtAppInitialize()

UxCenterVisibly() and UxCenterWidgetVisibly()

Function Centers the specified swidget under the pointer at its current location.

Synopsis

```
#include <UxLib.h>

void UxCenterVisibly(swidget shadow_widget, swidget
   top_shadow_widget);

void UxCenterWidgetVisibly(Widget widget, Widget
   top_widget);
```

Return Value None.

Description The `UxCenterVisibly()` function centers the specified `shadow_widget` under the pointer at its current location. If the interface's `top_shadow_widget` (usually a shell) is partially off the screen, the interface is moved as far as needed to make it completely visible.

The `UxCenterWidgetVisibly()` function works the same way, but is intended for interfaces that are not built with UIM/X's swidgets, such as convenience dialogs.

Example Here, an interface is displayed with its "printButton" automatically positioned under the pointer:

```
UxCenterVisibly(printButton, drawingAreaInterface);

UxPopupInterface(drawingAreaInterface, no_grab);
```

UxCreate*ShadowWidget*()

Function Creates the shadow widget—called a swidget.

Synopsis

```
#include <UxLib.h>

#include <UxShadowWidget.h> (Refer to the table that
   follows.)

swidget UxCreateShadowWidget(char *name, swidget
   parent);
```

Return Value Each of the `UxCreateShadowWidget()` functions returns the newly created shadow widget. The validity of the returned swidget may be checked using the `UxIsValidSwidget()` function.

Description The Ux Convenience Library `UxCreateShadowWidget()` functions simplify the creation of Motif widgets. For each widget class in Motif, UIM/X defines a shadow widget—called a *swidget*—class containing additional information relevant to interactive editing. `UxCreateShadowWidget()` creates the shadow widget. To create the actual widget, you must subsequently call `UxCreateWidget()`.

When using, replace *ShadowWidget* by the name of the widget.

Valid Function Names The following table lists all of the valid `UxCreateShadowWidget()` functions and the corresponding include file for each:

Type of Widget	Create Function	Include File
Application Shell	UxCreateApplicationShell()	UxApplSh.h
Arrow Button	UxCreateArrowButton()	UxArrB.h
Arrow Button Gadget	UxCreateArrowButtonGadget()	UxArrBG.h
Bulletin Board	UxCreateBulletinBoard()	UxBboard.h
Bulletin Board Dialog	UxCreateBulletinBoardDialog()	UxBbD.h
Cascade Button	UxCreateCascadeButton()	UxCascB.h
Cascade Button Gadget	UxCreateCascadeButtonGadget()	UxCascBG.h
Command Box	UxCreateCommand()	UxComm.h
Dialog Shell	UxCreateDialogShell()	UxDialSh.h

Type of Widget	Create Function	Include File
Drawing Area	UxCreateDrawingArea()	UxDrArea.h
Drawn Button	UxCreateDrawnButton()	UxDrawnB.h
Error Dialog	UxCreateErrorDialog()	UxErrorD.h
File Selection Box	UxCreateFileSelectionBox()	UxFsBox.h
File Selection Box Dialog	UxCreateFileSelectionBoxDialog()	UxFsBD.h
Form	UxCreateForm()	UxForm.h
Form Dialog	UxCreateFormDialog()	UxFormD.h
Frame	UxCreateFrame()	UxFrame.h
Information Dialog	UxCreateInformationDialog()	UxInforD.h
Label	UxCreateLabel()	UxLabel.h
Label Gadget	UxCreateLabelGadget()	UxLabelG.h
List	UxCreateList()	UxList.h
Main Window	UxCreateMainWindow()	UxMainW.h
Menu Shell	UxCreateMenuShell()	UxMenuSh.h
Message Box	UxCreateMessageBox()	UxMsgBox.h
Message Box Dialog	UxCreateMessageBoxDialog()	UxMsgBD.h
Override Shell	UxCreateOverrideShell()	UxOverSh.h
Paned Window	UxCreatePanedWindow()	UxPaneW.h
Prompt Dialog	UxCreatePromptDialog()	UxPromptD.h
Push Button	UxCreatePushButton()	UxPushB.h
Push Button Gadget	UxCreatePushButtonGadget()	UxPushBG.h
Question Dialog	UxCreateQuestionDialog()	UxQuestD.h
Row Column	UxCreateRowColumn()	UxRowCol.h
Scale	UxCreateScale()	UxScale.h
Scroll Bar	UxCreateScrollBar()	UxScrBar.h
Scrolled Window	UxCreateScrolledWindow()	UxScrW.h
Selection Box	UxCreateSelectionBox()	UxSelBox.h
Selection Box Dialog	UxCreateSelectionBoxDialog()	UxSelBD.h

Type of Widget	Create Function	Include File
Drawing Area	UxCreateDrawingArea()	UxDrArea.h
Drawn Button	UxCreateDrawnButton()	UxDrawnB.h
Error Dialog	UxCreateErrorDialog()	UxErrorD.h
File Selection Box	UxCreateFileSelectionBox()	UxFsBox.h
File Selection Box Dialog	UxCreateFileSelectionBoxDialog()	UxFsBD.h
Form	UxCreateForm()	UxForm.h
Form Dialog	UxCreateFormDialog()	UxFormD.h
Frame	UxCreateFrame()	UxFrame.h
Information Dialog	UxCreateInformationDialog()	UxInforD.h
Label	UxCreateLabel()	UxLabel.h
Label Gadget	UxCreateLabelGadget()	UxLabelG.h
List	UxCreateList()	UxList.h
Main Window	UxCreateMainWindow()	UxMainW.h
Menu Shell	UxCreateMenuShell()	UxMenuSh.h
Message Box	UxCreateMessageBox()	UxMsgBox.h
Message Box Dialog	UxCreateMessageBoxDialog()	UxMsgBD.h
Override Shell	UxCreateOverrideShell()	UxOverSh.h
Paned Window	UxCreatePanedWindow()	UxPaneW.h
Prompt Dialog	UxCreatePromptDialog()	UxPromptD.h
Push Button	UxCreatePushButton()	UxPushB.h
Push Button Gadget	UxCreatePushButtonGadget()	UxPushBG.h
Question Dialog	UxCreateQuestionDialog()	UxQuestD.h
Row Column	UxCreateRowColumn()	UxRowCol.h
Scale	UxCreateScale()	UxScale.h
Scroll Bar	UxCreateScrollBar()	UxScrBar.h
Scrolled Window	UxCreateScrolledWindow()	UxScrW.h
Selection Box	UxCreateSelectionBox()	UxSelBox.h
Selection Box Dialog	UxCreateSelectionBoxDialog()	UxSelBD.h

Type of Widget	Create Function	Include File
Separator	UxCreateSeparator()	UxSep.h
Separator Gadget	UxCreateSeparatorGadget()	UxSepG.h
Template Dialog	UxCreateTemplateDialog()	UxTempD.h
Text	UxCreateText()	UxText.h
Toggle Button	UxCreateToggleButton()	UxTogB.h
Toggle Button Gadget	UxCreateToggleButtonGadget()	UxTogBG.h
Top Level Shell	UxCreateTopLevelShell()	UxTopSh.h
Transient Shell	UxCreateTransientShell()	UxTranSh.h
Warning Dialog	UxCreateWarningDialog()	UxWarnD.h
Working Dialog	UxCreateWorkingDialog()	UxWorkD.h

Example

A number of convenience dialogs are created initially as either a Message Box or Selection Box. To configure the dialog more specifically to a certain type of dialog, set the `dialogType` resource:

```
{
      swidget exitDialog;
      exitDialog = UxCreateMessageBox("exitDialog",
      parent);


      UxPutX(exitDialog,(int)900);

      UxPutY(exitDialog,(int)900);

      UxPutDialogStyle(exitDialog,
      "dialog_application_modal");

      UxPutDialogTitle(exitDialog, "Keith’s New
      Stuff");

      UxPutMsgDialogType(exitDialog,
      "dialog_question");

      UxPutMessageString(exitDialog, "Exit Keith’s New
      Stuff?");

      UxCreateWidget(exitDialog);

   }
```

Note that you use `UxPutMsgDialogType()` to set the DialogType property of a MessageBox, not `UxPutDialogType()`.

See Also UxCreateWidget(), UxGetProperty(), UxPutProperty()

UxCreateSubproc()

Function Creates a subprocess object for running a particular subprocess command.

Synopsis

```
   #include "UxSubproc.h"
      handle UxCreateSubproc(char *command, char
      *default_args,void *output_handler);
```

Return Value Returns a `handle` to the subprocess. This UIM/X identifier is used by other subprocess functions to identify the subprocess. If the subprocess cannot be created, `UxCreateSubproc()` returns `ERROR` (which is defined as `-1` in `UxSubproc.h`).

Description The `UxCreateSubproc()` function creates a subprocess object for running a particular subprocess `command`. `UxCreateSubproc()` also identifies a `default_args` string and an `output_handler` function. Initializing a subprocess with this function does not start execution of the process.

The `default_args` parameter is passed with the `command` when either `UxRunSubproc()` or `UxExecSubproc()` is called if the second parameter to that function is `NULL`.

The `output_handler` parameter identifies a function to be called when output is generated by the subprocess. This is a user-supplied function. You can also identify an output handler function using `UxSetSubprocFunction()` call. If you plan to direct the output of the subprocess to a text widgets, use the `UxAppendTo()` handler function provided by UIM/X. If you plan to write your own output handler, refer to `UxTransferToBuffer()`.

Example This code fragment demonstrates how to create a subprocess for the `ls` system command. Since the second argument to `UxRunSubproc()` is `NULL`, the default argument, `"-F"`, is appended to the command string.

```
{

   /* Declare a handle for the subprocess. */
      handle lsHandle;

   /* Create a handle for the subprocess. */

   /* The command is "/bin/ls", the default */

   /* argument string is "-F", and the */

   /* output handler is UxAppendTo(). */
   lsHandle = UxCreateSubproc("/bin/ls", "-F",
      UxAppendTo);


   /* Direct the output of the subprocess */
   /* to the "outputText" text widget. */


   if(UxSetSubprocClosure(lsHandle,
   UxGetWidget(outputText)) == ERROR)
   {
   printf("Cannot set subprocess closure
      for/bin/ls.\n");
   return;
   }
   /* Run the subprocess using */
   /* the default argument string. */
   if (UxRunSubproc(lsHandle, NULL) == ERROR)
   {
   printf("Cannot start the /bin/ls
      subprocess.\n");return;
   }
}
```

See Also

UxAppendTo(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxCreateWidget()

Function

Creates a widget for a specified swidget.

Synopsis

```
#include <UxLib.h>

Widget UxCreateWidget(swidget shadow_widget);
```

Return Value

Returns a pointer to the newly created widget.

Description

The UxCreateWidget() function creates a widget (type Widget) associated with the specified shadow_widget (type swidget). Before executing this function, the shadow widget must have been created using UxCreate*ShadowWidget*().

Example

The following code segment creates a push button gadget as a child of shadow widget drawingArea1:

```
{
    swidget drawingArea1;

    swidget shadow_button;

    Widget button;


    shadow_button = UxCreatePushButtonGadget
        ("startButton", drawingArea1);

    UxPutLabelString(shadow_button, "Start");

    UxPutX(shadow_button, 10);

    UxPutY(shadow_button, 10);

    UxPutWidth(shadow_button, 60);

    UxPutHeight(shadow_button, 20);


    button = UxCreateWidget(shadow_button);
}
```

See Also

UxCreate*ShadowWidget*(), UxGet*Property*(), UxPut*Property*()

UxDelayedDeleteSubproc()

Function

Sets up the subprocess associated with the passed handle for a delayed deletion.

Synopsis

```
#include <UxSubproc.h>

int UxDelayedDeleteSubproc(handle index);
```

Return Value

Returns `NO_ERROR` for success, or `ERROR` for failure.

Description

The `UxDelayedDeleteSubproc()` function sets up the subprocess associated with the passed handle for a delayed deletion.

Deleting a subprocess terminates the subprocess and deletes all data associated with its execution.

Deletion takes place the *next* time `UxDelayedDeleteSubproc()` is called. To force deletion of a subprocess whose deletion was delayed, call `UxDelayedDeleteSubproc()` a second time, with an index of -1.

This function is especially useful for cases where you need to delete the data associated with the subprocess from the exit callback of the subprocess.

See Also

UxAppendTo(), UxCreateSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback() , UxSetSubprocFunction(), UxTransferToBuffer()

UxDelayUpdate() and UxUpdate()

Function Updates the widget only once after all of the desired changes have been made.

Synopsis

```
#include <UxLib.h>

void UxDelayUpdate(swidge shadow_widget);

void UxUpdate(swidge shadow_widget);
```

Return Value None.

Description Normally, all `UxPut`*`Property`*`()` calls immediately update the shadow widget and the widget. When changing more than one property for a shadow widget, it's more efficient to update the widget only once after all of the desired changes have been made. To do this, use the `UxDelayUpdate()` function to temporarily suspend the automatic updates until all changes have been made. Then use the `UxUpdate()` function to force a complete update of the widget.

Example To change the location of a widget named `exitDialog` without updating the display until all the related properties are set, use `UxDelayUpdate()` and `UxUpdate()` like this:

```
UxDelayUpdate(exitDialog);/* Inhibit screen update.
   */

UxPutX(exitDialog, 11);/* Set the new property
   values. */

UxPutY(exitDialog, 200);

UxUpdate(exitDialog);/* Update the widget with *//*
   the new values. */
```

If you remove the `UxDelayUpdate()` and `UxUpdate()` calls from this example, the `exitDialog` interface jumps around on the screen as each property is set individually.

See Also UxGet*Property*(), UxPut*Property*()

UxDeleteSubproc()

Function

Terminates a subprocess and deletes all data associated with its execution.

Synopsis

```
#include <UxSubproc.h>

int UxDeleteSubproc(handle index);
```

Return Value

Returns `NO_ERROR` for success, or `ERROR` for failure.

Description

The `UxDeleteSubproc()` function terminates a subprocess and deletes all data associated with its execution. The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`. This function cannot be called from within an exit callback function.

To restart the subprocess, use `UxCreateSubproc()`.

See Also

UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxDestroyInterface()

Function Deletes the interface associated with the specified swidget.

Synopsis

```
#include <UxLib.h>

int UxDestroyInterface(swidget shadow_widget);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure.

Description The `UxDestroyInterface()` function deletes the interface associated with the specified `shadow_widget`, which must be a Shell widget or have an implicit Shell widget as its parent. (Compare to `UxDestroySwidget()`, which operates on all widgets, but with less error checking for top-level widgets.)

In Test Mode, the `UxDestroyInterface()` function does not actually destroy the interface if it was created interactively—it merely unmaps it. Interfaces created dynamically (using function calls) *are* actually destroyed if you execute this function in Test Mode.

Example Suppose you use a function like this to create and display an interface:

```
printDialog = create_printDialog();

UxPopupInterface(printDialog, no_grab);
```

You hide the interface like this:

```
UxPopdownInterface(printDialog);
```

When your application no longer needs the interface, it destroys it using this call:

```
UxDestroyInterface(printDialog);
```

See Also UxDestroySwidget(), UxPopdownInterface(), UxPopupInterface(), UxMap(), UxRealizeInterface(), UxUnmap(), XtDestroyWidget()

UxDestroySwidget()

Function Destroys the specified swidget and the widget associated with it.

Synopsis

```
#include <UxLib.h>

void UxDestroySwidget(swidget shadow_widget);
```

Return Value None.

Description The `UxDestroySwidget()` function deletes the specified `shadow_widget` and the widget (type `Widget`) associated with it.

In Test Mode, the `UxDestroySwidget()` function does not actually destroy the widget—it merely unmaps it. Widgets created dynamically (using function calls) *are* actually destroyed if you execute this function in Test Mode.

See Also UxDestroyInterface(), UxMap(), UxPopdownInterface(), UxPopupInterface(), UxRealizeInterface(), UxUnmap(), XtDestroyWidget()

UxDispatchEvent()

Function

Dispatches an X event.

Synopsis

```
#include <UxLib.h>

void UxDispatchEvent(XEvent *event);
```

Return Value

None.

Description

The `UxDispatchEvent()` function dispatches an X event. In compiled code, it is equivalent to the function `XtDispatchEvent()`.

Example

You can combine the `UxNextEvent()` and `UxDispatchEvent()` functions to form a custom event loop.

```
XEvent event;

for (;;)
{
   /* Get the next event. */
   UxNextEvent(&event);

   switch(event.type)

   {
   /* Add a case for each special event you want to */
   /* trap for special processing. All other events */
   /* are dispatched in the "default" case. */
  default:
      UxDispatchEvent(&event);
      break;
      }
 }
```

See Also

UxDispatchEvent(), UxNextEvent(), UxMainLoop(), UxNotify() and UxWaitForNotify(), XtDispatchEvent()

UxExecSubproc()

Function

Executes the subprocess created by `UxCreateSubproc()`.

Synopsis

```
#include <UxSubproc.h>

int UxExecSubproc(handle sub_handle, char *args);
```

Return Value

Returns `NO_ERROR` for success, or `ERROR` for failure.

Description

The `UxExecSubproc()` function executes the subprocess created by `UxCreateSubproc()`. If the subprocess is already running, `UxExecSubproc()` terminates the running process, and executes it again.

The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`.

The `args` parameter is a string of whitespace-separated Arguments for the command. If you specify an argument string `(args)`, it overrides the `default_args` established with `UxCreateSubproc()`. If args is `NULL`, `default_args` is used.

Example

Suppose your application initialized a subprocess with these calls:

```
mailxHandle = UxCreateSubproc("mailx", NULL,
   UxAppendTo);

UxSetSubprocClosure(mailxHandle,
   UxGetWidget(outputText));
```

Your application could subsequently run the subprocess by executing this call:

```
UxExecSubproc(mailxHandle, "-s \"Re: Your mail\"
   valerierob");
```

See Also

UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc, UxDeleteSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocClosure(), UxSetSubprocFunction(), UxTransferToBuffer()

UxExitSubproc()

Function

Terminates a running subprocess.

Synopsis

```
#include <UxSubproc.h>

int UxExitSubproc(handle sub_handle);
```

Return Value

Returns `NO_ERROR` for success, or `ERROR` for failure.

Description

The `UxExitSubproc()` function terminates a running subprocess, but maintains necessary data so that the process can be restarted by calling `UxRunSubproc()` or `UxExecSubproc()`.

So, this sequence:

```
UxExitSubproc(procHandle);

UxRunSubproc(procHandle, NULL);
```

is equivalent to:

```
UxExecSubproc(procHandle, NULL);
```

The `handle` argument references the subprocess handle returned by **`UxCreateSubproc().`**

See Also

UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxExpandBitmapFilename()

Function

Expands the bitmap file name to its full path name.

Synopsis

```
#include <resload.h>

char *UxExpandBitmapFilename(char *file_name);
```

Return Value

Returns a pointer to a string containing the expanded string. If this value is to be used, it should be copied into local storage. If the file is not found, `UxExpandBitmapFilename()` returns `NULL`.

Description

`UxExpandBitmapFilename()` expands the bitmap `file_name` to its full path name if it is found in any of the directories maintained by the path list structure `UxBitmapPath`. That is, this call:

```
UxExpandBitmapFilename(file_name);
```

is equivalent to:

```
UxExpandFilename(UxBitmapPath, file_name);
```

See Also

UxAddPath(), UxExpandEnv(), UxExpandFilename(), UxExpandResourceFilename(), UxFileExists(), UxFreePath(), UxGetPath(), UxInitPath(), UxResetPath()

UxExpandEnv()

Function

Expands all environment variable references in the string.

Synopsis

```
   #include <pathlist.h>

   char *UxExpandEnv(char *string);
```

Return Value

Returns a character string containing the expanded string. If this value is to be used, it should be copied into local storage.

Description

`UxExpandEnv()` recursively expands all environment variable references in the string. That is, it continues until there are no “`$`” characters left in the string.

Example

Here’s how you might expand a path that includes environment variables:

```
   strcpy(savePath,
      UxExpandEnv("$HOME/$DISPLAY/current/"));
```

See Also

UxAddPath(), UxExpandBitmapFilename(), UxExpandFilename(), UxExpandResourceFilename(), UxFreePath(), UxGetPath(), UxInitPath(), UxResetPath()

UxExpandFilename()

Function Expands a file name to its full path name if it is found in a given search path.

Synopsis

```
#include <pathlist.h>

char *UxExpandFilename(pathlist pathlist, char
   *file_name);
```

Return Value Returns a pointer to a string containing the complete path and file name. If this value is to be used, it should be copied into local storage. Returns `NULL` if `file_name` cannot be expanded.

Description `UxExpandFilename()` expands `file_name` to its full path name if it is found in any of the directories in the `pathlist` structure.

See Also UxAddPath(), UxExpandBitmapFilename(), UxExpandEnv(), UxExpandResourceFilename(), UxFreePath(), UxGetPath(), UxInitPath(), UxResetPath()

UxExpandResourceFilename()

Function

Expands a resource file name to its full path name.

Synopsis

```
    #include <resload.h>

    char *UxExpandResourceFilename(char *file_name);
```

Return Value

Returns a pointer to a string containing the complete path and resource file name. If this value is to be used, it should be copied into local storage. This function returns a `NULL` if the `file_name` cannot be expanded.

Description

`UxExpandResourceFilename()` expands the resource file name `file_name` to its full path name if it is found in any of the directories maintained by the path list structure `UxResourcePath.` That is, this call:

```
    UxExpandResourceFilename(file_name);
```

is equivalent to:

```
    UxExpandFilename(UxResourcePath, file_name);
```

`UxExpandResourceFilename()` can be used inside of `UxLoadResources()`. For example:

```
    UxLoadResources( UxExpandResourceFilename (
        file_name ) );
```

See Also

UxAddPath(), UxExpandBitmapFilename(), UxExpandEnv(), UxExpandFilename(), UxFreePath(), UxGetPath(), UxInitPath(), UxLoadResources(), UxResetPath()

UxFileExists()

Function Determines whether or not a file exists.

Synopsis

```
#include <pathlist.h>

int UxFileExists(char *file_name);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure.

Description The `UxFileExists()` function determines whether `file_name` exists. Since no file name expansion is performed, `file_name` is typically a complete path to the file.

See Also UxAddPath(), UxExpandBitmapFilename(), UxExpandEnv(), UxExpandFilename(), UxExpandResourceFilename(), UxFreePath(), UxGetPath(), UxInitPath(), UxResetPath()

UxFindSwidget()

Function Returns the swidget handle of the named widget.

Synopsis

```
#include <UxLib.h>

swidget UxFindSwidget(char *name);
```

Return Value Returns the swidget. The validity of the Return Value may be checked using the `UxIsValidSwidget()` function.

Description The `UxFindSwidget()` function returns the swidget handle of the named widget. This function is similar to `UxNameToSwidget()`, without the reference widget.

See Also UxIsValidSwidget(), UxWidgetToSwidget()

UxFreePath()

Function Frees the memory used by a path list.

Synopsis

```
#include <pathlist.h>

void UxFreePath(pathlist path);
```

Return Value None.

Description `UxFreePath()` frees the memory used by the path list `path`. After calling `UxFreePath()` for a path list value, the value should not be subsequently referenced.

See Also UxAddPath(), UxExpandBitmapFilename(), UxExpandEnv(), UxExpandFilename(), UxExpandResourceFilename(), UxFileExists(), UxGetPath(), UxInitPath(), UxResetPath()

UxGetAppDefault()

Function

Queries the X resource database for a resource string of the form `App-class.resource_string` or `argv[0].resource_string`.

Synopsis

```
#include <resload.h>

char *UxGetAppDefault( char *resource_string, char
   *default);
```

Return Value

Returns a pointer to the resource value string. If the specified resource has not been set, the call returns `default`.

Description

The `UxGetAppDefault()` function queries the X resource database for a resource string of the form `App-class.resource_string` or `argv[0].resource_string` where `App-class` is the application's class name as passed to `UxAppInitialize()` and `argv[0]` is the command name used to start the application.

See Also

UxGetResource(), UxGetDefault(), UxGetAppResource()

UxGetAppResource()

Function

Queries the X resource database for a resource string of the form `App-class.resource_string or argv[0].resource_string` or `argv[0]`.

Synopsis

```
#include <resload.h>

char *UxGetAppResource(char *resource_string);
```

Return Value

Returns a pointer to the resource value, or `NULL` if the resource value was not found.

Description

The `UxGetAppResource()` function queries the X resource database for a resource string of the form `App-class.resource_string` or `argv[0].resource_string` where `App-class` is the application's class name as passed to `UxAppInitialize()` and `argv[0]` is the command name used to start the application. If this resource has not been set, the call returns `NULL`.

See Also

UxGetResource(), UxGetDefault(), UxGetAppDefault(), UxGetParent(), UxGetWidget()

UxGetClass()

Function

Returns the Motif widget class for the specified swidget.

Synopsis

```
#include <UxLib.h>

WidgetClass UxGetClass(swidget shadow_widget);
```

Return Value

Returns the widget class pointer.

Description

The `UxGetClass()` function returns the Motif widget class for the specified `shadow_widget`.

Example

Suppose you are writing a routine that sets a toggle button, but you don't know if the toggle button is a widget or a gadget (or you want to reserve the right to change it later without rewriting your code).You could test it like this to ensure that the correct "set state" call is executed:

```
{
   extern swidget viewOutlineToggle;


   if (UxGetClass(viewOutlineToggle) ==
      xmToggleButtonWidgetClass)

   XmToggleButtonSetState(UxGetWidget
      viewOutlineToggle), True, False);


   elseif (UxGetClass(viewOutlineToggle) ==
      xmToggleButtonGadgetClass)XmToggleButtonGadgetSet
      State(UxGetWidget (viewOutlineToggle), True,
      False);


   else

   printf("Hey, viewOutlineToggle isn't a toggle
      button!\n");
 }
```

See Also

UxGetName(), UxGetParent(), UxGetWidget()

UxGetContext()

Function Returns a pointer to the current context of the specified swidget.

Synopsis

```
#include <UxLib.h>

void *UxGetContext(swidget shadow_widget);
```

Return Value Returns a pointer to the context.

Description The `UxGetContext()` function returns a pointer to the current context of the specified `shadow_widget`. This pointer should be cast appropriately. Each interface has a unique context structure whose type is `_UxC`*interface*. Its name is `Ux`*Interface*`Context`, where *interface* is the name of the top-level shadow widget in the interface.

For typical applications, context manipulation is provided automatically within the C code generated by UIM/X. You should use this function only if you have a special need to manipulate context in a certain way. Using methods is more elegant, and avoids manipulation of the context.

Note that context structure functionality is not supported in design time. It can only be used and tested with generated code. One possible work-around is to surround the function call with the `#ifndef DESIGN_TIME` flag.

See Also UxPutContext()

UxGetDefault()

Function

Queries the X resource database for a resource string of the form `App-class.resource_string` or `program_name.resource_string`.

Synopsis

```
#include <resload.h>

char *UxGetDefault(char *program_name, char
   *resource_string, char *default);
```

Return Value

Returns a pointer to the resource value. If this resource has not been set, the call returns the `default` parameter.

Description

The `UxGetDefault()` function queries the X resource database for a resource string of the form `App-class.resource_string` or `program_name.resource_string` where `App-class` was passed as the first parameter to `UxInitialize()` and is the application's class name and `program_name` is the command name used to start the application.

See Also

UxGetAppDefault(), UxGetAppResource(), UxGetResource()

UxGetName()

Function Returns the name of the specified swidget.

Synopsis

```
#include <UxLib.h>

char *UxGetName(swidget shadow_widget);
```

Return Value Returns a pointer to the `shadow_widget`'s name string. If this value is to be used, it should be copied to local storage.

Description The `UxGetName()` function returns the name of the specified `shadow_widget`.

See Also UxGetClass(), UxGetParent(), UxGetWidget()

UxGetParent()

Function Gets the parent shadow widget for the specified swidget.

Synopsis

```
#include <UxLib.h>

swidget UxGetParent(swidget shadow_widget);
```

Return Value Returns the swidget of the parent. The validity of the returned shadow widget may be checked using the `UxIsValidSwidget()` function.

Description The `UxGetParent()` macro gets the parent shadow widget for the specified `shadow_widget.`

See Also UxGetClass(), UxGetName(), UxIsValidSwidget()

UxGetPath()

Function Returns the list of directories stored in a path list.

Synopsis

```
#include <pathlist.h>

char *UxGetPath(pathlist path);
```

Return Value Returns a pointer to the first character of a null-terminated string. Do not free the memory used by this string.

Description `UxGetPath()` returns the list of directories stored by `path`. The directories in the returned string are separated by colons.

See Also UxAddPath(), UxExpandBitmapFilename(), UxExpandEnv(), UxExpandFilename(), UxExpandResourceFilename(), UxFileExists(), UxFreePath(), UxInitPath(), UxResetPath()

UxGet*Property*() and UxPut*Property*()

Function

Allows you to directly access any property of the named widget.

Synopsis

```
#include <UxLib.h>

return_type UxGetProperty(swidget sw);

int UxPutProperty(swidget sw, value_type value);
```

Return Value

The *return_type* for `UxGet`*Property*`()` and the *value_type* passed to `UxPut`*Property*`()` depends on which *Property* is being retrieved or set.

The `UxPut`*Property*`()` functions return `NO_ERROR` for success, or `ERROR` for failure (perhaps caused by a conversion error). A failure may also cause a message to be printed to standard error (`stderr`).

Description

The `UxGet`*Property*`()` and `UxPut`*Property*`()` functions allow you to directly access any property of the named widget.

The `UxPut`*Property*`()` functions set the specified property for the widget. The `UxGet`*Property*`()` functions return the specified value or a pointer to the value. `UxGet`*Property*`()` cannot be used in the Property Editor until the referenced widget exists. Setting a resource in the property table through `UxGet`*Property*`()` will work under run time only if the widget exists for the swidget parameter to the `UxGet`*Property*`()` call.

Property is replaced by the name of the property For example, `UxPutBackground()` or `UxGetBackground()`, where `Background` is the property name.

When C++ bindings are enabled, UIM/X provides a C++ wrapper class for each of the Motif widgets, and declares Motif objects of an interface as objects of these classes. Each of these classes has member-function equivalents of the `UxGet`*Property*`()` and `UxPut`*Property*`()` functions for the properties that apply to them. The member-function equivalents are of the form:

```
return_type GetProperty();

void SetProperty(value_type value);
```

Note that no `swidget` parameter is needed, since C++ member functions have an implicit `this` parameter.

Valid Function Names and Property Values

The following table lists all possible completions for *Property*.

Note the following:

- The term *Property* corresponds to Motif resources.
- Not all properties apply to all widgets.
- The Get and Put functions use UIM/X's simplified data types for most properties. If you use Xm and Xt code to get or set property values, you must use the data types listed in the *OSF/Motif Programmer's Reference* for each property.
- To manipulate the string within a text widget using Get and Put functions, you must use the Text property, *not* the Valueproperty. This difference is necessary because the Value property is an integer (type int) for all other widgets.
- The asterisk, *, indicates that the Get function of the property returns a temporary buffer that is automatically freed. If the returned value is to be used later, you must make a copy of it. You should never free a value returned by a UxGet*Property* call.
- If no Return Type appears in the table, the UxGet*Property* does not exist.
- If no Value Type appears in the table, the UxPut*Property* does not exist.
- The Xt column specifies whether or not Xt versions of the UxGet and UxPut functions are defined in *uimx_directory*/contrib/XtCodePuts. This contrib provides two files, UxXtGets.h and UxXtPuts.h, that define Xt versions of many the UxGet and UxPut functions. C++ bindings are not provided by this contrib.

Property	Get Function Return Type		Put Function Value Type	Xt
Accelerator	string	*	string	x
Accelerators	string	*	string	
AcceleratorText	string	*	string	x
AdjustLast	string	*	string	x
AdjustMargin	string	*	string	x
Alignment	string	*	string	
AllowOverlap	string	*	string	x
AllowResize	string	*	string	x
AllowShellResize	string	*	string	x

Property	Get Function Return Type		Put Function Value Type	Xt
AncestorSensitive	string	*	string	x
ApplyLabelString	string	*	string	x
Argc	integer		integer	x
Argv	stringTable		stringTable	
ArmColor	string	*	string	
ArmPixmap	string	*	string	
ArrowDirection	string	*	string	
AudibleWarning	string	*	string	
AutomaticSelection	string	*	string	x
AutoShowCursorPosition	string	*	string	x
AutoUnmanage	string	*	string	x
Background	string	*	string	
BackgroundPixmap	string	*	string	
BaseHeight	integer		integer	x
BaseWidth	integer		integer	x
BlinkRate	integer		integer	x
BorderColor	string	*	string	
BorderPixmap	string	*	string	
BorderWidth	integer		integer	x
BottomAttachment	string	*	string	
BottomOffset	integer		integer	
BottomPosition	integer		integer	x
BottomShadowColor	string	*	string	
BottomShadowPixmap	string	*	string	
BottomWidget	string	*	string	
ButtonFontList	string	*	string	
CancelButton	string	*	string	

Property	Get Function Return Type		Put Function Value Type	Xt
CancelLabelString	string	*	string	x
CascadePixmap	string	*	string	
ChildHorizontalAlignment	string	*	string	
ChildHorizontalSpacing	integer		integer	
ChildPlacement	string	*	string	
Children	stringTable			
ChildType	string	*	string	
ChildVerticalAlignment	string	*	string	
ClipWindow	string	*		
ColorMap	integer		integer	
Columns	short		short	x
Command	string	*	string	x
CommandWindow	string	*		
CommandWindowLocation	string	*	string	
CreateManaged	string		voidFunction	
CreatePopupChildProc	voidFunction		voidFunction	
CursorPosition	integer		integer	x
CursorPositionVisible	string	*	string	x
DecimalPoints	short		short	x
DefaultButton	string	*	string	
DefaultButtonShadowThickness	integer		integer	x
DefaultButtonType	string	*	string	
DefaultFontList	string	*	string	
DefaultPosition	string	*	string	x
DeleteResponse	string	*	string	
DialogStyle	string	*	string	

Property	Get Function Return Type		Put Function Value Type	Xt
DialogTitle	string	*	string	x
DialogType*	string	*	string	
Directory	string	*	string	x
DirectoryValid	string	*	string	x
DirListItemCount	integer		integer	x
DirListItems	string	*	string	
DirListLabelString	string	*	string	x
DirMask	string	*	string	x
DirSearchProc	voidFunction		voidFunction	
DirSpec	string	*	string	x
DoubleClickInterval	integer		integer	x
Editable	string	*	string	x
EditMode	string	*	string	
EntryAlignment	string	*	string	
EntryBorder	integer		integer	x
EntryClass	string	*	string	
EntryVerticalAlignment	string	*	string	
FileListItemCount	integer		integer	x
FileListLabelString	string	*	string	x
FileListItems	string	*	string	
FileSearchProc	voidFunction		voidFunction	
FileTypeMask	string	*	string	
FillOnArm	string	*	string	x
FillOnSelect	string	*	string	x
FilterLabelString	string	*	string	x
FontList	string	*	string	
Foreground	string	*	string	

Property	Get Function Return Type		Put Function Value Type	Xt
FractionBase	integer		integer	x
Geometry	string	*	string	
Height	integer		integer	x
HeightInc	integer		integer	x
HelpLabelString	string	*	string	x
HighlightColor	string	*	string	
HighlightOnEnter	string	*	string	x
HighlightPixmap	string	*	string	
HighlightThickness	string	*	string	x
HistoryItemCount	integer		integer	x
HistoryItems	string	*	string	
HistoryMaxItems	integer		integer	x
HistoryVisibleItemCount	integer		integer	x
HorizontalScrollBar	string	*	string	
HorizontalSpacing	integer		integer	x
Iconic	string	*	string	x
IconMask	string	*	string	
IconName	string	*	string	x
IconNameEncoding	string	*	string	
IconPixmap†	string	*	string	
IconWindow	string	*	string	
IconX	integer		integer	x
IconY	integer		integer	x
Increment	integer		integer	x
IndicatorOn	string	*	string	x
IndicatorSize	integer		integer	x
IndicatorType	string	*	string	
InitialDelay	integer		integer	

Property	Get Function Return Type		Put Function Value Type	Xt
InitialFocus	string	*	string	
InitalState	string	*	string	
Input	string	*	string	
InputMethod	string	*	string	
InsertPosition	integer		integer	
IsAligned	string	*	string	x
IsHomogeneous	string	*	string	x
ItemCount	integer		integer	x
Items	string	*	string	
KeyboardFocusPolicy	string	*	string	
LabelFontList	string	*	string	
LabelInsensitivePixmap	string	*	string	
LabelPixmap	string	*	string	
LabelString	string	*	string	x
LabelType	string	*	string	
LeftAttachment	string	*	string	
LeftOffset	integer		integer	x
LeftPosition	integer		integer	x
LeftWidget	string	*	string	
ListItemCount	integer		integer	x
ListItems	string	*	string	x
ListLabelString	string	*	string	x
ListMarginHeight	integer		integer	x
ListMarginWidth	integer		integer	x
ListSizePolicy	string	*	string	
ListSpacing	integer		integer	x
ListUpdated	string	*	string	x
ListVisibleItemCount	integer		integer	x

Property	Get Function Return Type		Put Function Value Type	Xt
MainWindowMarginHeight	integer		integer	x
MainWindowMarginWidth	integer		integer	x
MappedWhenManaged	string	*	string	x
MappingDelay	integer		integer	x
Margin	integer		integer	x
MarginBottom	integer		integer	x
MarginHeight	integer		integer	x
MarginLeft	integer		integer	x
MarginRight	integer		integer	x
MarginTop	integer		integer	x
MarginWidth	integer		integer	x
MaxAspectX	integer		integer	x
MaxAspectY	integer		integer	x
MaxHeight	integer		integer	x
Maximum	integer		integer	x
MaxLength	integer		integer	x
MaxWidth	integer		integer	x
MenuAccelerator	string	*	string	x
MenuBar	string	*		
MenuHelpWidget	string	*	string	
MenuHistory	string	*	string	
MenuPost	integer		integer	
MessageAlignment	string	*	string	
MessageString	string	*	string	x
Message Window	string	*	string	
MinAspectX	integer		integer	x
MinAspectY	integer		integer	x
MinHeight	integer		integer	x

Property	Get Function Return Type		Put Function Value Type	Xt
MinimizeButtons	string	*	string	x
Minimum	integer		integer	x
MinWidth	integer		integer	x
Mnemonic	string	*	string	
MnemonicCharSet	string	*	string	x
MsgDialogType‡	string	*	string	
MultiClick	string	*	string	
MustMatch	string	*	string	x
MwmDecorations	integer		integer	x
MwmFunctions	integer		integer	x
MwmInputMode	string	*	string	x
MwmMenu	string	*	string	x
NavigationType	string	*	string	
NoMatchstring	string	*	string	x
NoResize	string	*	string	
NumChildren	integer		integer	
NumColumns	integer		integer	
OkLabelString	string	*	string	x
Orientation	string	*	string	
OverrideRedirect	string	*		x
Packing	string	*	string	
PageIncrement	integer		integer	x
PaneMaximum	integer		integer	x
PaneMinimum	integer		integer	x

Property	Get Function Return Type		Put Function Value Type	Xt
Pattern	string	*	string	x
PendingDelete	string	*	string	x
PopupEnabled	string	*	string	x
PositionIndex	integer		integer	
PreeditType	string	*	string	
ProcessingDirection	string	*	string	
PromptString	string	*	string	x
PushButtonEnabled	string	*	string	x
QualifySearchDataProc	voidFunction		voidFunction	
RadioAlwaysOne	string	*	string	x
RadioBehavior	string	*	string	x
RecomputeSize	string	*	string	x
RefigureMode	string	*	string	x
RepeatDelay	integer		integer	x
Resizable	string	*	string	x
ResizeHeight	string	*	string	x
ResizePolicy	string	*	string	
ResizeWidth	string	*	string	x
RightAttachment	string	*	string	
RightOffset	integer		integer	x
RightPosition	integer		integer	x
RightWidget	string	*	string	
RowColumnType	string	*	string	
Rows	short		short	x
RubberPositioning	string	*	string	x
SashHeight	integer		integer	x

Property	Get Function Return Type		Put Function Value Type	Xt
SashIndent	integer		integer	x
SashShadowThickness	integer		integer	x
SashWidth	integer		integer	x
SaveUnder	string	*	string	x
ScaleHeight	integer		integer	x
ScaleMultiple	integer		integer	x
ScaleWidth	integer		integer	x
ScrollBarDisplayPolicy	string	*	string	
ScrollBarPlacement	string	*	string	
ScrolledWindowMarginHeight	integer		integer	x
ScrolledWindowMarginWidth	integer		integer	x
ScrollHorizontal	string	*	string	x
ScrollingPolicy	string	*	string	
ScrollLeftSide	string	*	string	x
ScrollTopSide	string	*	string	x
ScrollVertical	string	*	string	x
SelectColor	string	*	string	
SelectedItemCount	integer		integer	x
SelectedItems	string	*	string	
SelectInsensitivePixmap	string	*	string	
SelectionArray	string	*	string	
SelectionArrayCount	integer		integer	x
SelectionLabelString	string	*	string	x
SelectionPolicy	string	*	string	
SelectPixmap	string	*	string	
SelectThreshold	integer		integer	x
Sensitive	string	*	string	x

Property	Get Function Return Type		Put Function Value Type	Xt
SeparatorOn	string	*	string	x
SeparatorType	string	*	string	
Set	string	*	string	x
ShadowThickness	integer		integer	x
ShadowType	string	*	string	
ShellUnitType	string	*	string	
ShowArrows	string	*	string	x
ShowAsDefault	integer		integer	x
ShowSeparator	string	*	string	x
ShowValue	string	*	string	x
SkipAdjust	string	*	string	x
SliderSize	integer		integer	x
Source	XmTextSourc e		XmTextSource	
Spacing	integer		integer	x
StringDirection	string	*	string	
SubMenuId	string	*	string	
SymbolPixmap	string	*	string	
TearOffModel	string	*	string	
Text	string	*	string	x
TextAccelerators			string	
TextColumns	short		short	x
TextFontList	string	*	string	
TextString	string	*	string	x
TextTranslations			string	
Title	string	*	string	x
TitleEncoding	string	*	string	
TitleString	string	*	string	x

Property	Get Function Return Type		Put Function Value Type	Xt
TopAttachment	string	*	string	
TopCharacter	integer		integer	x
TopItemPosition	integer		integer	x
TopOffset	integer		integer	x
TopPosition	integer		integer	x
TopShadowColor	string	*	string	
TopShadowPixmap	string	*	string	
TopWidget	string	*	string	
Transient	string	*	string	x
TransientFor	string	*	string	
Translations	string	*	string	
TraversalOn	string	*	string	x
TroughColor	string	*	string	
UnitType	string	*	string	
UseAsyncGeometry	string	*	string	x
UserData	pointer		pointer	x
Value	integer		integer	x
ValueWcs	string	*	string	
VerifyBell	string	*	string	x
VerticalScrollBar	string	*		
VerticalSpacing	integer		integer	x
VisibleItemCount	integer		integer	x
VisibleWhenOff	string	*	string	x
Visual	visualPointer		visualPointer	
VisualPolicy	string	*	string	
WaitForWm	string	*	string	x
WhichButton	integer		integer	x
Width	integer		integer	x

Property	Get Function Return Type		Put Function Value Type	Xt
WidthInc	integer		integer	x
WinGravity	string	*	string	
\WindowGroup	string	*	string	
WmTimeout	integer		integer	x
WordWrap	string	*	string	x
X	integer		integer	x
Y	integer		integer	x

*. Use `UxPutDialogType()` and `UxGetDialogType()` to set and retrieve the DialogType property of SelectionBox widgets.

Use `UxPutMsgDialogType()` and `UxGetMsgDialogType()` to set and retrieve the DialogType property of MessageBox widgets. Both the MessageBox and SelectionBox widget classes have a DialogType property, but UIM/X renames DialogType to MsgDialogType for the MessageBox widget class.

†. If you plan to change a Shell widget's IconPixmap property via a call to `UxPutIconPixmap()` or `XtSetValues()`, set IconPixmap when you create the widget. Otherwise, the calls to `UxPutIconPixmap()` and `XtSetValues()` have no effect. This is due to a bug in Motif 1.2.2.

‡. MsgDialogType corresponds to the `XmNDialogType` resource of a MessageBox widget.

UxGetResource()

Function

Queries the X resource database for a resource string of the form `App-class.resource_string` or `program_name.resource_string`.

Synopsis

```
#include <resload.h>

char *UxGetResource(char *program_name, char
   *resource_string);
```

Return Value

A pointer to the resource value, or `NULL` if the resource value was not found.

Description

The `UxGetResource()` function queries the X resource database for a resource string of the form `App-class.resource_string` or `program_name.resource_string` where `App-class` is the application's class name (passed as the first parameter) and `program_name` is the command name used to start the application. If this resource has not been set, the call returns `NULL`.

See Also

UxGetAppDefault(), UxGetAppResource(), UxGetDefault()

UxGetSubprocPid()

Function

Checks if a subprocess associated with a given handle is still running and returns its process id.

Synopsis

```
#include <UxSubproc.h>

int UxGetSubprocPid(handle object_handle);
```

Return Value

Returns the process id of the subprocess if it is running, or `ERROR` otherwise.

Description

The `UxGetSubprocPid()` function checks if a subprocess associated with a given `handle` is still running and returns its process id. If the subprocess was improperly terminated, it performs error checking and closes open file descriptors associated with the running subprocess. The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`.

See Also

UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxGetWidget()

Function Returns the widget pointer for the specified swidget.

Synopsis

```
#include <UxLib.h>

Widget UxGetWidget(swidget shadow_widget);
```

Return Value Returns the X widget pointer to the corresponding swidget. The pointer is `NULL` when no X widget is associated with the specified `shadow_widget`.

Description The `UxGetWidget()` function returns the widget pointer for the specified `shadow_widget`. Use this call whenever you need an actual widget ID, type `Widget`.

See Also UxFindSwidget(), UxGetClass(), UxGetName(), UxWidgetToSwidget()

UxInitPath()

Function Initializes a search path.

Synopsis

```
#include <pathlist.h>

pathlist UxInitPath(char *path);
```

Return Value Returns a path list structure which maintains an ordered list of directories.

Description The `UxInitPath()` function initializes a search path given an initial search path in path. The directories within path can be separated by any white space (spaces, newlines or tabs), colons, or commas. The directory names can contain environment variables.

Example UIM/X uses two variables, `UxResourcePath` and `UxBitmapPath`, to store the search path for resource and bitmap files, respectively. These variables are set for you automatically. If you want your application to search for files in nonstandard locations, however, you can create your own search paths.

The `UxInitPath()` function creates an initial path list. To replace the default search paths in `UxBitmapPath` with your own, you add the following code in the Ux Main Program (via the Program Layout Editor):

```
#include <resload.h>

...

UxTopLevel = XtAppInitialize(&UxAppContext,
   "$PJ_APP_CLASS_NAME",

NULL, 0, &argc, argv, NULL, NULL,

0);

UxAppInitialize("$PJ_APP_CLASS_NAME", &argc, argv);


/*------------------------------------------------

* Insert initialization code for your application
   here

*---------------------------------------------*/


UxFreePath(UxBitmapPath);

UxBitmapPath = UxInitPath("/usr/mike/bitmaps");
```

```
UxAddPath(UxBitmapPath, "/usr/steve/bitmaps");
```

First, you include `<resload.h>` for the declarations of the global variable `UxBitmapPath`. Then you call `UxFreePath()` to free the old value of `UxBitmapPath`.

See Also

UxAddPath(), UxExpandBitmapFilename(), UxExpandEnv(), UxExpandFilename(), UxExpandResourceFilename(), UxFileExists(), UxFreePath(), UxGetPath(), UxResetPath()

UxIsValidSwidget()

Function Checks the validity of the specified swidget.

Synopsis

```
#include <UxLib.h>

int UxIsValidSwidget(swidget shadow_widget);
```

Return Value Returns a `1` if the `shadow_widget` handle is valid, or `0` (zero) otherwise.

Description The `UxIsValidSwidget()` function checks the validity of the specified `shadow_widget`.

See Also UxGetWidget(), UxName(), UxParent(), UxWidgetToSwidget()

UxLoadResources()

Function Loads a resource file into the current resource database.

Synopsis

```
#include <resload.h>

void UxLoadResources(char *file_name);
```

Return Value None.

Description `UxLoadResources()` loads a resource file into the current resource database. Values loaded with this call are appended to the database such that they may be overridden by previously loaded resources, including those loaded automatically by the Xt Intrinsics.

`file_name` is the name of the resource file in the current directory. If the resource does not reside in the current directory, specify the full path name. If the resource file resides in one of the search paths listed in `UxResourcePath`, use `UxExpandResourceFilename()` to get the full path name of the resource file:

```
UxLoadResources(UxExpandResourceFilename
    (file_name));
```

See Also UxAddPath()for a discussion of the UxResourcePath global path list variables, UxAppInitialize(), UxExpandResourceFillename, UxPreInitialize(), UxOverrideResources(), XtAppInitialize()

UxMainLoop()

Function Enters the main event loop for the application.

Synopsis

```
#include <UxLib.h>

void UxMainLoop();
```

Return Value None.

Description The `UxMainLoop()` function enters the main event loop for the application. Applications are expected to exit in response to some user action. `UxMainLoop()` calls `XtMainLoop()`, so this call can be substituted by a call to a customized event loop, if desired.

See Also UxDispatch(), UxNextEvent(), UxNotify() and UxWaitForNotify(), XtMainLoop()

UxManage()

Function Manages the swidget passed to it.

Synopsis

```
#include <UxLib.h>

void UxManage(swidget shadow_widget);
```

Return Value None.

Description The `UxManage()` function manages the swidget passed to it. `UxManage()` converts the swidget into a Widget and then calls `XtPopup` if the widget is a shell or `XtManageChild` for composite widgets.

See Also UxUnmanage()

UxMap()

Function Causes the specified swidget to re-appear on the screen.

Synopsis

```
#include <UxLib.h>

void UxMap(swidget shadow_widget);
```

Return Value None.

Description The `UxMap()` function causes the specified `shadow_widget` to re-appear on the screen, after it has been removed from the screen by a call to `UxUnmap()`. For top-level widgets, `XtPopup()` is used; for gadgets, `XtManageChild()` is used; for all other widget classes, `XtMapWidget()` is used.

To unmap interfaces, use `UxPopdownInterface()`.

See Also UxDestroySwidget(), UxPopdownInterface(), UxPopupInterface(), UxRealizeInterface(), UxUnmap(), XtManageChild(), XtManageChildren(), XtMapWidget(), XtPopdown(), XtPopup(), XtUnmanageChild(), XtUnmanageChildren(), XtUnmapWidget()

UxNameToSwidget()

Function Searches for a swidget by name.

Synopsis

```
#include <UxLib.h>

swidget UxNameToSwidget(swidget reference, char
   *name);
```

Return Value Returns the first swidget found with a matching name. If there is more than one swidget with the same `name`, the Return Value may be unpredictable.

Returns `NULL` if no match is found.

The validity of the returned value may be checked using the `UxIsValidSwidget()` function.

Description The `UxNameToSwidget()` function searches for a shadow widget by `name`. It first traverses the siblings and children of the `reference`shadow widget, then goes to the top of the widget hierarchy and begins searching there.

See Also UxFindSwidget()

UxNextEvent()

Function

Returns the next X event form the event queue via the parameter event.

Synopsis

```
#include <UxLib.h>

void UxNextEvent(XEvent *event);
```

Return Value

None.

Description

The `UxNextEvent()` function returns the next X event form the event queue via the parameter `event`. In compiled code, it is equivalent to the function `XtAppNextEvent()`. (Refer to the example with `UxDispatchEvent()`.)

See Also

UxDispatchEvent(), UxMainLoop(), UxNotify(), XtAppNextEvent()

UxNotify() and UxWaitForNotify()

Function Controls a nested event loop.

Synopsis

```
#include <UxLib.h>

void UxNotify();

void UxWaitForNotify();
```

Return Value None.

Description The `UxNotify()` function sets the notify flag to break from `UxWaitForNotify()` event loop. The `UxWaitForNotify()` function executes an event loop until the notify flag is set using `UxNotify()`.

These functions are intended for use only when adding a graphical user interface to an existing flow-controlled terminal application.

See Also UxDispatchEvent(), UxMainLoop(), UxNextEvent()

UxOverrideResources()

Function Loads a resource file into the current database.

Synopsis

```
#include <resload.h>

void UxOverrideResources(char *file_name);
```

where `file_name` is the name of the resource file

Return Value None.

Description The `UxOverrideResources()` function loads a resource file into the current database. Resource values loaded with this call override all previously loaded resources, including those loaded automatically by the Xt Intrinsics.

See Also UxLoadResources()

UxPopdownInterface()

Function Pops down the interface associated with the specified swidget.

Synopsis

```
#include <UxLib.h>

int UxPopdownInterface(swidget shadow_widget);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure.

Description `UxPopdownInterface()` function pops down the interface associated with the specified `shadow_widget`. It is a function designed specifically for Shell widgets and widget with implicit Shells as their parent—otherwise, `UxPopdownInterface()` is ignored. (Compare to `UxUnmap()`, which operates on all widgets., but with less error-checking for top-level widgets.)

See Also UxPopupInterface(), UxUnmap(), XtManageChild(), XtManageChildren(), XtMapWidget(), XtPopdown(), XtPopup(), XtUnmanageChild(), XtUnmanageChildren(), XtUnmapWidget()

UxPopupInterface()

Function

Pops up the interface associated with the specified swidget.

Synopsis

```
#include <UxLib.h>

int UxPopupInterface(swidget shadow_widget, int
   grabtype);
```

Return Value

Returns `NO_ERROR` for success, or `ERROR` for failure.

Description

The `UxPopupInterface()` function pops up the interface associated with the specified `shadow_widget`. The function is for Shell widgets and widgets that have an implicit Shell as parent. (Compare to `UxMap()`, which operates on all widgets, but with less error-checking.)

The `grabtype` parameter determines how events are processed while the interface is displayed:

- `no_grab`—all events are processed normally. That is, all windows within the application remain active.
- `nonexclusive_grab`—application events are passed to the specified interface, or any other interface displayed using this grab type (or no_grab). This allows an application to display multiple dialogs in which the user's response to the dialogs can occur in any order.
- `exclusive_grab`—application events are passed only to the specified interface. This grab type allows an application to display "cascading" dialogs. Cascading dialogs require the user to respond to the most recently displayed dialog within an application.

The grab type mechanism is provided at the Xt Intrinsics level. In general, you can use `no_grab` for all of your interfaces, and rely on the properties `DialogStyle` and `MwmInputMode` to achieve the desired behavior.

In Test Mode, the values `exclusive_grab` and `nonexclusive_grab` are mapped to `no_grab`. This prevents you from locking up your UIM/X session. To test `exclusive_grab`, you must generate and compile the source code for the interface.

Example

The `MotifMain.prj` example (located in the *uimx_directory*`/contrib/MotifMain` directory) demonstrates how to create, display, and hide "application modal" dialogs. It creates the appropriate dialog behavior by properly reparenting the dialogs to the application's main window and setting the `DialogStyle` property to `"dialog_primary_application_modal"`.

Refer to the example listed with `UxPopdownInterface()`.

See Also

UxDestroyInterface(), UxMap(), UxPopupInterface(), UxRealizeInterface(), UxUnmap(), XtManageChild(), XtManageChildren(), XtMapWidget(), XtUnmapWidget(), XtPopup(), XtPopdown(), XtUnmanageChild(), XtUnmanageChildren()

UxPostMenu()

Function Pops up a menu on the specified widget at the cursor position.

Synopsis

```
#include <UxLib.h>

void UxPostMenu(Widget widget, XEvent *event, String
   *argv, Cardinal *argc);
```

Return Value None.

Description The `UxPostMenu()` function pops up a menu on the specified `widget` at the cursor position. This function is normally used as an action, so the Arguments follow the standard protocol for action procedures. If this function is used as an action procedure, then `widget` and `event` specify where and how the action was invoked.

`UxPostMenu` relies on the uniqueness of the menu name.

See Also UxAddActions()

UxPreInitialize()

Function

Performs all the initialization needed by UIM/X prior to initializing the Xt Intrinsics.

Synopsis

```
#include <UxLib.h>

void UxPreInitialize();
```

Return Value

None.

Description

The `UxPreInitialize()` function performs all the initialization needed by UIM/X prior to initializing the Xt Intrinsics. This function must be called before any other UIM/X or Xt functions.

Note: `UxAppInitialize()` and `UxPreInitialize()` now supercede `UxInitialize()` and `UxOptionInitialize()`, although the latter functions are still supported.

See Also

UxGetDefault(), UxGetResource(), XtAppInitialize(), UxAppInitialize()

UxPutContext()

Function Uses a pointer to set the context of the specified swidget.

Synopsis

```
#include <UxLib.h>

int UxPutContext(swidget shadow_widget, void
   *context);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure.

Description The `UxPutContext()` macro uses a pointer to set the context of the specified `shadow_widget` to the values in the context structure provided. (Casting context pointers is done via `UxGetContext()`.) Each interface has a unique context structure whose type is `_UxC`*interface*, where *interface* is the name of the top-level shadow widget in the interface.

For typical applications, context manipulation is provided automatically within the C code generated by UIM/X. You should use this function only if you have a special need to manipulate contexts in a certain way.

See Also UxGetContext()

UxPut*Property*()

Refer to `UxGet`*Property*`()`.

UxRealizeInterface()

Function Realizes all X widgets in the interface associated with swidget.

Synopsis

```
#include <UxLib.h>

int UxRealizeInterface(swidget shadow_widget);
```

Return Value Returns NO_ERROR for success, or ERROR for failure.

Description The `UxRealizeInterface()` function realizes all X widgets in the interface associated with `shadow_widget`. This function is automatically called within the C code generated by UIM/X for each interface. Generally, you should need it only when writing your own interface code by hand.

Example The following code segment shows how to create a simple interface:

```
{
   /* Declare the shadow widgets. */
   swidget manager, button;
   /* Create the shadow widget structures */
   /* for the manager and the button. */
   manager = UxCreateRowColumn("rowColumn", NO_PARENT);
   button = UxCreatePushButton("pushButton", manager);
   /* Set the initial state of the manager. */
   UxPutWidth(manager, 50);
   UxPutHeight(manager, 100);
   /* Set the initial state of the button. */
   UxPutLabelString(button, "Push Me");
   /* Create the actual widgets (type Widget). */
   UxCreateWidget(manager); UxCreateWidget(button);
   /* Realize the interface. */
   UxRealizeInterface(manager);
}
```

UxRegisterFunction()

Function Registers a function with UIM/X's built-in Interpreter in the design-time executable.

Synopsis

```
void UxRegisterFunction(char *name, void
   *function_ptr);
```

Return Value None.

Description The `UxRegisterFunction()` function is used to register a function with UIM/X's built-in Interpreter in the design-time executable. The function is identified by a character string `name` and a function pointer `function_ptr`.

Note: `UxRegisterFunction()` is intended exclusively for use within UIM/X during development. Therefore, it is not available for use in generated code and does not appear in `libuimx.a`.

`UxRegisterFunction()` is useful for two reasons. First, registering functions makes them known to the Interpreter, so there is minimal delay when they are first encountered. Second, `UxRegisterFunction()` ensures that desired library functions are available during development.

To use `UxRegisterFunction()`, you must create a new `uimx` executable.

UxRegisterGlobal()

Function Makes the address of an external variable known to the Interpreter.

Synopsis

```
void UxRegisterGlobal(char *name, char *gptr);
```

Return Value None.

Description `UxRegisterGlobal()` makes the address of an external variable known to the Interpreter, eliminating the delay associated with looking up the variable the first time it is encountered.

The commented examples at the end of `uimx_main.c` and `uimx_main.cc` illustrate its use.

UxRemoveTabGroup()

This function is obsolete. Its behavior is replaced by setting the resource Navigation Type to `none` using the Property Editor.

UxResetPath()

Function Replaces the list of directories stored in a path list.

Synopsis

```
#include <pathlist.h>

void UxResetPath(pathlist path, char *new_paths);
```

Return Value None.

Description `UxResetPath()` replaces the list of directories stored in `pathlist` by the directories in the `new_paths` string. The `new_paths` string lists directories separated by spaces, colons, commas, newlines, or tabs.

See Also UxAddPath(), UxExpandBitmapFilename(), UxExpandEnv(), UxExpandFilename(), UxExpandResourceFilename(), UxFileExists(), UxFreePath(), UxGetPath(), UxInitPath()

UxRunSubproc()

Function

Executes a subprocess that was originally created by `UxCreateSubproc()`.

Synopsis

```
#include <UxSubproc.h>

int UxRunSubproc(handle sub_handle, char *args);
```

Return Value

Returns `NO_ERROR` for success, or `ERROR` for failure. An error occurs if the subprocess is already running.

Description

The `UxRunSubproc()` function executes a subprocess that was originally created by `UxCreateSubproc()`. If the process is already running, `UxRunSubproc()` returns `ERROR`. If a `non-NULL` argument string `(args)` is provided, it overrides the `default_args` string passed to `UxCreateSubproc()`. The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`.

Example

Refer to the example listed with `UxCreateSubproc()`.

See Also

UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxSendSubproc() and UxSendSubprocNoCR()

Function

Sends a command string to the subprocess.

Synopsis

```
#include <UxSubproc.h>

int UxSendSubproc(handle sub_handle, char *command);

int UxSendSubprocNoCR(handle sub_handle, char
   *command);
```

Return Value

Returns `NO_ERROR` for success, or `ERROR` for failure.

Description

The `UxSendSubproc()` function sends a command string to the subprocess. A carriage return is automatically appended to the command string.

The `UxSendSubprocNoCR()` function is identical to `UxSendSubproc()`, except it *does not* append a carriage return to the command string. The `handle` argument in both `UxSendSubproc()` and `UxSendSubprocNoCR()` references the subprocess handle returned by `UxCreateSubproc()`.

Example

Suppose you had started a subprocess to run the `vi` text editor. This code fragment could be used as a “save callback” to send a command to the subprocess to save the current file:

```
{
   /* The application keeps track of vi’s current mode.
      */
   extern Boolean insertMode;


   /* This is the handle to the vi subprocess. */
   extern handle viHandle;


   if (insertMode)
   {
      /* Turn off insert mode by sending */
      /* an "escape" character.
      */UxSendSubprocNoCR(viHandle, "\033");
   }


   /* Send the "write" command */
   /* (with a carriage return).
      */UxSendSubproc(viHandle, ":w");


   if (insertMode)
   {
      /* Return to insert mode, if necessary.
      */UxSendSubprocNoCR(viHandle, "i");
   }
}
```

The `vi` command to write the current file to disk is `:w` followed by a carriage return. Before issuing the command, however, the application must make sure `vi` is not in “insert mode”—it does this by sending an “escape” character. Presumably, `insertMode` is a variable that the application maintains to keep track of `vi's` insert mode. After saving the file, insert mode is restored, if necessary.

See Also

UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxSendSubprocNoCR()

Refer to `UxSendSubproc()`.

UxSetSubprocClosure()

Function Specifies data passed to the output handler function for a given subprocess.

Synopsis

```
#include <UxSubproc.h>

int UxSetSubprocClosure(handle handle, char
   *closure);
```

Note: The `closure` parameter is shown here as a type char*. However, if the subprocess is using `UxTransferToBuffer()` with a customized output handler function, you are free to declare and use `closure` as whatever type you need. Of course, the output handler function should expect the same type.

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure. An error may occur if the subprocess `handle` is invalid.

Description The `UxSetSubprocClosure()` function is used to specify data that is to be passed to the output handler function for a given subprocess. The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`. If the subprocess is using the `UxAppendTo()` output handler, the `closure` parameter must be a widget ID (type `Widget`) identifying a valid text widget.

See Also UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxSetSubprocEcho()

Function Turns echoing of input on or off for the given subprocess handle.

Synopsis

```
#include <UxSubproc.h>

int UxSetSubprocEcho(handle sub_handle, int echo);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure. An error may occur if the subprocess handle is invalid.

Description The `UxSetSubprocEcho()` function turns echoing of input on or off for the given subprocess `handle`. The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`.

When echoing is turned on, data sent to a subprocess using `UxSendSubproc()` or `UxSendSubprocNoCR()` is echoed back to the application.

To turn echoing on, the `echo` parameter should be `1`. To turn echoing off, the echo parameter should be `0`. By default, echoing is turned off when you create a new subprocess.

For `UxSetSubprocEcho()` to take effect, it must be called after the subprocess handle is created, but before calling `UxExecSubproc()` or `UxRunSubproc()`.

See Also UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocExitCallback(), UxSetSubprocFunction(), UxTransferToBuffer()

UxSetSubprocExitCallback()

Function Specifies a function to be called when a subprocess is terminated or stopped.

Synopsis

```
#include <UxSubproc.h>

int UxSetSubprocExitCallback(handle sub_handle, void
   *function);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure.

Description The `UxSetSubprocExitCallback()` function specifies a `function` to be called when a subprocess is terminated or stopped. The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`.

The call to `UxSetSubprocExitCallback()` must be made *after* the call to `UxCreateSubproc()`, and *before* the call to either the functions `UxRunSubproc()` or `UxExecSubproc()`.

The `function` parameter is a pointer to the exit callback function. Since the function is not a true Xt callback, it should be written as shown here (not as an Xt callback):

```
void function(int pid, int status, handle sub_handle)
```

This function receives the process ID (`pid`) of the terminated subprocess.

The `status` parameter is set to the value set by the `wait()` system call, and can be used to determine the exit value if the subprocess was exited, or the signal number if it was stopped. The `handle` value points to the terminated subprocess.

Do not call the `UxRunSubproc()` function from within any exit callback function set by the `UxSetSubprocExitCallback()` function. Although, in UNIX terms, the process has in fact stopped; UIM/X may not have cleared its internal data structures. Subsequently, the message: `The subprocess is already active` may be displayed.

In addition, do not call `UxDeleteSubproc` from within the exit callback function you register here. `UxDeleleteSubproc` destroys the subprocess structure.

Example Suppose you provided the following exit callback function:

```
void MyExitCallback(pid, status, sub_handle)

int pid;

int   status;
```

```
handle s ub_handle;
{
   printf("Subprocess terminated.\n");
   printf("              PID: %d\n", pid);
   printf("     Exit status: %d\n", status);
}
```

You can add the exit callback to the subprocess as shown here (after setting the subprocess closure, but before running the subprocess):

```
{
/* Declare a handle for the subprocess. */
handle lsHandle;


/* Create a handle for the subprocess. */
lsHandle = UxCreateSubproc("/bin/ls", NULL,
   UxAppendTo);


/* Direct the output of the subprocess */
/* to the "outputText" text widget. */
if(UxSetSubprocClosure(lsHandle,
   UxGetWidget(outputText))==ERROR)
   {
   printf("Cannot set subprocess closure for
   /bin/ls.\n");
   return;
   }


/* Create an exit callback (to be executed */
/* when the subprocesses terminates).
   */if(UxSetSubprocExitCallback(lsHandle,
   MyExitCallback)
{
printf("Cannot set subprocess exit callback
   for/bin/ls.\n");
   return;
   }
   /* Run the subprocess using the default argument
      string. */
   if (UxRunSubproc(lsHandle, NULL) == ERROR)
   {
      printf("Cannot start the /bin/ls subprocess.\n");
      return;
   }
}
```

See Also UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocFunction(), UxTransferToBuffer()

UxSetSubprocFunction()

Function Specifies a function to be used to handle output from a subprocess.

Synopsis

```
#include <UxSubproc.h>

int UxSetSubprocFunction(handle sub_handle, void
   *function);
```

Return Value Returns `NO_ERROR` for success, or `ERROR` for failure.

Description The `UxSetSubprocFunction()` function specifies a `function` to be used to handle output from a subprocess. This allows you to change the output handler for a subprocess after `UxCreateSubproc()` has been called. The `handle` argument references the subprocess handle returned by `UxCreateSubproc()`.

UIM/X provides one output handler function, `UxAppendTo()`, for appending the output from the subprocess to a text widget. If you plan to write your own output handler, refer to `UxTransferToBuffer()`.

See Also UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxTransferToBuffer()

UxShellWidget()

Function Returns the shell widget at the root of the widget hierarchy.

Synopsis

```
#include <UxLib.h>

Widget UxShellWidget(swidget shadow_widget);
```

Return Value Returns an X widget pointer. Returns `NULL` if `shadow_widget` is null or has been deleted.

Description The `UxShellWidget()` function returns the shell widget ID (type `Widget`) for the shell widget at the root of the widget hierarchy for the interface which includes the specified `shadow_widget`. This function is useful for accessing a shell widget when you are unsure whether the shell was created explicitly during development, or automatically by UIM/X.

See Also UxGetWidget(), UxIsValidSwidget(), UxParent()

UxTextAppend()

Function Appends a string to a text widget.

Synopsis

```
#include <UxLib.h>

void UxTextAppend(Widget text_widget, char *string);
```

Return Value None.

Description The `UxTextAppend()` function appends a `string` to a `text widget`. This function is used within the `UxAppendTo()` output handler.

See Also UxAppendTo()

UxTransferToBuffer()

Function Reads a buffer containing output from a subprocess.

Synopsis

```
#include <UxSubproc.h>

char *UxTransferToBuffer(int file_descriptor, int
   *status);
```

Return Value Returns a character pointer to the first character of a null-terminated string of length at most 2048 bytes.

Description The `UxTransferToBuffer()` function is used to read a buffer containing output from a subprocess. You write the output handler function so that it redirects output to a buffer.

The general form of an output handler function is as follows:

```
   void function(int file_descriptor, type *closure)
   {
      int status;
      char *buffer;

      do {
         buffer=UxTransferToBuffer(file_descriptor,
         &status);

      /* Add your code here to process the buffer data.
         */
         .
         .
         .

      } while (status);
 }
```

The closure data can be any type, as long as it matches the type of closure specified as a parameter to `UxSetSubprocClosure()`. (The `UxAppendTo()` output handler uses the closure parameter for the text widget ID.) `file_descriptor` is the file descriptor that has output on it.

In the body of the output function, `UxTransferToBuffer()` is used to obtain a pointer to the first character of a null-terminated string contained in a 2048-byte static text buffer. The output from the subprocess is deposited in 2048 byte blocks into this buffer. Successive calls to `UxTransferToBuffer()` return successive blocks. The `status` variable is non-zero (`1`) if there is more output data present. It is up to the output handler function to read from the buffer until `status` is `0` (zero).

See Also

UxAppendTo(), UxCreateSubproc(), UxDelayedDeleteSubproc(), UxDeleteSubproc(), UxExecSubproc(), UxExitSubproc(), UxGetSubprocPid(), UxRunSubproc(), UxSendSubproc(), UxSetSubprocClosure(), UxSetSubprocEcho(), UxSetSubprocExitCallback(), UxSetSubprocFunction()

UxUnmanage()

Function Unmanages a swidget.

Synopsis

```
#include <UxLib.h>

void UxUnmanage(swidget shadow_widget);
```

Return Value None.

Description The `UxUnmanage()` function unmanages the swidget passed to it. `UxUnmanage()` converts the swidget into a Widget and then calls `XtPopdown` if the widget is a shell or `XtUnmanageChild()` for composite widgets.

See Also UxManage()

UxUnmap()

Function Causes the specified swidget to disappear from the screen.

Synopsis

```
#include <UxLib.h>

void UxUnmap(swidget shadow_widget);
```

Return Value None.

Description The `UxUnmap()` function causes the specified `shadow_widget` to disappear from the screen. If the widget is a top-level widget, `XtPopdown()` is used; if the widget is a gadget, `XtUnmanageChild()` is used; otherwise, `XtUnmapWidget()` is used.

For interfaces, use `UxPopdownInterface()`.

See Also UxDestroySwidget(), UxMap(), UxPopdownInterface(), UxPopupInterface(), UxRealizeInterface(), XtManageChild(), XtManageChildren(), XtMapWidget(), XtPopdown(), XtPopup(), XtUnmanageChild(), XtUnmanageChildren(), XtUnmapWidget()

UxUpdate()

Refer to `UxDelayUpdate()`.

UxWaitForNotify()

Refer to `UxNotify()`.

UxWidgetToSwidget()

Function Gets the swidget associated with the specified widget.

Synopsis

```
#include <UxLib.h>

swidget UxWidgetToSwidget(Widget widget);
```

Return Value Returns the swidget corresponding to the given widget.

Description The `UxWidgetToSwidget()` function gets the shadow widget associated with the specified `widget`.

Since it is possible to create widgets dynamically using Xm and Xt calls, you should test the validity of the returned swidget using `UxIsValidSwidget()`.

See Also UxFindSwidget(), UxGetWidget(), UxIsValidSwidget()

Swidget Methods

Overview

This chapter contains the reference pages for the methods supported by the Connection Editor for each of the UIM/X widgets.

Each reference page describes the associated arguments and return values, explains the usage of the method, and, where applicable, provides a list of related topics and reference pages.

Note: For all swidgets that accept an ItemList: Itemlist is a comma-separated list of items. Any item which contains a comma or a backslash must have that character preceded by a backslash. The same is true for all swidget methods that return a list of items. Note also that all backslashes must be protected from the C preprocessor. Thus, as an extreme example, to specify an item with the name "\", the argument to the swidget must be specified as "\\\\".

AddItemsToBeginning Method

Description	Adds items to the beginning of a List or ScrolledList.
Arguments	char *ItemList
Return Value	None.
Applies To	List, ScrolledList
See Also	AddItemsToEnd DeleteItemsAtBeginning DeleteItemsAtEnd

AddItemsToEnd Method

Description	Adds items to the end of a List or ScrolledList.
Arguments	char *ItemList
Return Value	None.
Applies To	List, ScrolledList
See Also	AddItemsToBeginning, DeleteItemsAtBeginning, DeleteItemsAtEnd

Deiconify Method

Description	Deiconifies a window if it is currently iconified.
Arguments	None.
Return Value	None.
Applies To	ApplicationWindow, SecondaryWindow
See Also	Iconify

DeleteAllItems Method

Description	Deletes all items from a List or ScrolledList.
Arguments	None.
Return Value	None.
Applies To	List, ScrolledList
See Also	DeleteItems, DeleteItemsAtBeginning, DeleteItemsAtEnd, DeleteSelectedItems

DeleteItems Method

Description	Deletes a specified set of items from a List or ScrolledList.
Arguments	char *ItemList
Return Value	None.
Applies To	List, ScrolledList
See Also	DeleteAllItems, DeleteItemsAtBeginning, DeleteItemsAtEnd, DeleteSelectedItems

DeleteItemsAtBeginning Method

Description	Deletes a specified number of items from the beginning of a List or ScrolledList.
Arguments	int Number
Return Value	None.
Applies To	List, ScrolledList
See Also	DeleteAllItems, DeleteItems, DeleteItemsAtEnd, DeleteSelectedItems

DeleteItemsAtEnd Method

Description	Deletes a specified number of items from the end of a List or ScrolledList.
Arguments	int Number
Return Value	None.
Applies To	List, ScrolledList
See Also	DeleteAllItems, DeleteItems, DeleteItemsAtBeginning, DeleteSelectedItems

DeleteSelectedItems Method

Description	Deletes the selected items from a List or ScrolledList.
Arguments	None.
Return Value	None.
Applies To	List, ScrolledList
See Also	DeleteAllItems, DeleteItems, DeleteItemsAtBeginning, DeleteItemsAtEnd

DeselectAllItems Method

Description	Deselects all items in a List or ScrolledList.
Arguments	None.
Return Value	None.
Applies To	List, ScrolledList
See Also	SelectAllItems

DeselectItems Method

Description	Deselects a specified set of items in a List or ScrolledList.
Arguments	char *ItemList
Return Value	None.
Applies To	List, ScrolledList
See Also	SelectItems

Exit Method

Description	Exits the application by calling the system exit() function.
Arguments	int status The exit status.
Return Value	Does not return.
Comments	In test mode the system exit() function is not called. Instead, the tester is informed that the call has been made.
Applies To	ApplicationShell, ApplicationWindow

GetDirectory Method

Description	Retrieves the current directory in a FileSelectionBox or FileSelectionBoxDialog.
Arguments	None.
Return Value	char * The directory name.
Applies To	FileSelectionBox, FileSelectionBoxDialog
See Also	GetDirectory, SetDirectory, SetPattern

GetItemCount Method

Description	Returns the number of items in a List or ScrolledList.
Arguments	None.
Return Value	int *
Applies To	List, ScrolledList
See Also	GetSelectedItemCount

GetItems Method

Description	Returns the items in a List or ScrolledList.
Arguments	None.
Return Value	char *
Applies To	List, ScrolledList
See Also	GetItemCount, GetSelectedItems, SetItems

GetPattern Method

Description	Gets the current file search pattern.
Arguments	None.
Return Value	char * The file search pattern.
Applies To	FileSelectionBox, FileSelectionBoxDialog
See Also	GetDirectory, SetDirectory, SetPattern

GetSelectedItemCount Method

Description	Returns the number of selected items in a List or ScrolledList.
Arguments	None.
Return Value	int *
Applies To	List, ScrolledList
See Also	GetItemCount

GetSelectedItems Method

Description	Returns the selected items of a List or ScrolledList.
Arguments	None.
Return Value	char *
Applies To	List, ScrolledList
See Also	GetItems

GetText Method

Description	Gets the text contained in a ScrolledText, Text, or TextField object.
Arguments	None.
Return Value	char *The text.
Applies To	ScrolledText, Text, TextField
See Also	SetText

GetTextString Method

Description	Gets the string that appears in the text edit selection field of a FileSelectionBox or FileSelectionBoxDialog.
Arguments	None.
Return Value	char *The string.
Applies To	FileSelectionBox, FileSelectionBoxDialog
See Also	SetTextString

GetToggleState Method

Description	Gets the current state of a CheckButton or RadioButton.
Arguments	None.
Return Value	char *The current state, either "true" or "false".
Applies To	CheckButton, RadioButton
See Also	SetToggleState

GetValue Method

Description	Gets the current value of a slider's position in a HorizScale or VertScale.
Arguments	None.
Return Value	int The value
Applies To	HorizScale, VertScale
See Also	SetValue

GoToBeginning Method

Description	Scrolls the list so that the first item, if not already visible, becomes visible.
Arguments	None.
Return Value	None.
Applies To	List, ScrolledList
See Also	GoToEnd, GoToItem

GoToEnd Method

Description	Scrolls the list so that the last item, if not already visible, becomes visible.
Arguments	None.
Return Value	None.
Applies To	List, ScrolledList
See Also	GoToBeginning, GoToItem

GoToItem Method

Description	Scrolls the list so that the specified item, if not already visible, becomes visible.
Arguments	char *Item
Return Value	None.
Applies To	List, ScrolledList
See Also	GoToBeginning, GoToEnd

Hide Method

Description	Hides an object.
Arguments	None.
Return Value	None.
Applies To	All components.
See Also	Show

Iconify Method

Description	Iconifies a window if it is not already iconified.
Arguments	None.
Return Value	None.
Applies To	ApplicationWindow, SecondaryWindow
See Also	Deiconify

Insensitive Method

Description	Makes an object insensitive to input.
Arguments	None.
Return Value	None.
Applies To	All components
See Also	Sensitive

ReplaceItems Method

Description	Replaces a set of items in a List or ScrolledList with the corresponding items of a second set.
Arguments	char *Replace char *With
Return Value	None.
Applies To	List, ScrolledList
See Also	ReplaceSelectedItems

ReplaceSelectedItems Method

Description	Replaces selected items in a List or ScrolledList with the corresponding items in the specified set.
Arguments	char *With
Return Value	None.
Applies To	List, ScrolledList
See Also	ReplaceItems

SelectAllItems Method

Description	Selects all items in a List or ScrolledList.
Arguments	None.
Return Value	None.
Applies To	List, ScrolledList
See Also	DeselectAllItems

SelectItems Method

Description	Selects a specified set of items in a List or ScrolledList.
Arguments	char *ItemList
Return Value	None.
Applies To	List, ScrolledList
See Also	DeselectItems

Sensitive Method

Description	Makes an object sensitive to input.
Arguments	None.
Return Value	None.
Applies To	All components.
See Also	Insensitive

SetBackground Method

Description	Sets the background color of an object.
Arguments	char * Color The desired background color
Return Value	None.
Applies To	All components.
See Also	SetForeground

SetDialogTitle Method

Description	Sets the title to display in the title bar of a dialog box.
Arguments	char * Title The desired title
Return Value	None.
Applies To	FileSelectionBoxDialog, MessageBoxDialog

SetDirectory Method

Description	Sets the directory in a FileSelectionBox or FileSelectionBoxDialog.
Arguments	char * Directory The directory name.
Return Value	None.
Applies To	FileSelectionBox, FileSelectionBoxDialog
See Also	GetDirectory, GetPattern, SetPattern

SetForeground Method

Description	Sets the foreground color of an object.
Arguments	char * Color The desired foreground color.
Return Value	None.
Applies To	All components except ApplicationWindow, RadioBox, and SecondaryWindow
See Also	SetBackground

SetIconName Method

Description	Sets the icon name used for a window when it is minimized.
Arguments	char * IconNameThe desired icon name
Return Value	None.
Applies To	ApplicationShell, ApplicationWindow, SecondaryWindow, TopLevelShell

SetItems Method

Description	Sets the items in a List or ScrolledList.
Arguments	char *ItemList
Return Value	None.
Applies To	List, ScrolledList
See Also	GetItems

SetLabelPixmap Method

Description	Sets the pixmap to display on an object when its LabelType is "pixmap".
Arguments	char * PixmapnameThe path/filename of the desired pixmap.
Return Value	None.
Applies To	CheckButton, DefaultButton, Label, PushButton, RadioBox
See Also	SetLabelString

SetLabelString Method

Description	Sets the text string of an object if its LabelType is "string".
Arguments	char * LabelString The desired text string
Return Value	None.
Applies To	CheckButton, DefaultButton, Label, Pushbutton, RadioButton
See Also	SetLabelPixmap

SetMessageString Method

Description	Sets the text message to display in a MessageBox or MessageBoxDialog.
Arguments	char * MessageString The desired message.
Return Value	None.
Applies To	MessageBox, MessageBoxDialog

SetPattern Method

Description	Sets the file search pattern.
Arguments	char * Pattern The file search pattern.
Return Value	None.
Applies To	FileSelectionBox, FileSelectionBoxDialog
See Also	GetDirectory, GetPattern, SetDirectory

SetText Method

Description	Sets the text contained in a ScrolledText, Text, or TextField object.
Arguments	char * Text The text.
Return Value	None.
Applies To	ScrolledText, Text, TextField
See Also	GetText

SetTextString Method

Description	Sets the string that appears in the text edit selection field of a FileSelectionBox or FileSelectionBoxDialog.
Arguments	char * TextString The string.
Return Value	None.
Applies To	FileSelectionBox, FileSelectionBoxDialog
See Also	GetTextString

SetTitle Method

Description	Sets the text to display in the title bar of a window
Arguments	char * Title The desired title.
Return Value	None.
Applies To	ApplicationShell, ApplicationWindow, SecondaryWindow, TopLevelShell

SetTitleString Method

Description	Sets the title to display with a HorizScale or VertScale.
Arguments	char *TitleStringThe desired title.
Return Value	None.
Applies To	HorizScale, VertScale

SetToggleState Method

Description	Sets the state of a CheckButton or RadioButton.
Arguments	char * StateThe desired state, either "true" or "false".
Return Value	None.
Applies To	CheckButton, RadioButton
See Also	GetToggleState

SetValue Method

Description	Sets the current value of a slider’s position in a HorizScale or VertScale.
Arguments	int ValueThe value.
Return Value	None.
Applies To	HorizScale, VertScale
See Also	GetValue

Show Method

Description	Shows an object.
Arguments	None.
Return Value	None.
Applies To	All components.
See Also	Hide

Index

Index

Index

Index

Index