Section 3 - Mrm Functions

This page describes the format and contents of each reference page in Section 3, which covers the Motif Resource Manager (Mrm) functions.

Name

Function – a brief description of the function.

Synopsis

This section shows the signature of the function: the names and types of the arguments, and the type of the return value. The header file *<Mrm/MrmPublic.h>* declares all of the public Mrm functions.

Inputs

This subsection describes each of the function arguments that pass information to the function.

Outputs

This subsection describes any of the function arguments that are used to return information from the function. These arguments are always of some pointer type, so you should use the C address-of operator (**&**) to pass the address of the variable in which the function will store the return value. The names of these arguments are sometimes suffixed with *_return* to indicate that values are returned in them. Some arguments both supply and return a value; they will be listed in this section and in the "Inputs" section above. Finally, note that because the list of function arguments is broken into "Input" and "Output" sections, they do not always appear in the same order that they are passed to the function. See the function signature for the actual calling order.

Returns

This subsection explains the return values of the function. Mrm functions typically return one of the following values: MrmSUCCESS, MrmPARTIAL_SUCCESS, MrmBAD_HIERARCHY, MrmNOT_FOUND, MrmWRONG_TYPE, MrmNOT_VALID, MrmDISPLAY_NOT_OPENED, or MrmFAILURE. To be safe, you should check the return value against MrmSUCCESS or MrmPARTIAL_SUCCESS, and then check for specific errors on non-success. When an error occurs, the functions call `XtWarning`() with a descriptive error message.

Availability

This section appears for functions that were added in Motif 2.0 or later, and also for functions that are now superseded by other, preferred, functions.

Description

This section explains what the function does and describes its arguments and return value. If you've used the function before and are just looking for a refresher, this section and the synopsis above should be all you need.

Usage

This section appears for most functions and provides less formal information about the function: when and how you might want to use it, things to watch out for, and related functions that you might want to consider.

Example

This section provides an example of the use of the function. It also shows the corresponding UIL code needed for the example.

Structures

This section shows the definition of any structures, enumerated types, typedefs, or symbolic constants used by the function.

Procedures

This section shows the syntax of any prototype procedures used by the function.

See Also

This section refers you to related functions, UIL file format sections, and UIL data types. The numbers in parentheses following each reference refer to the sections of this book in which they are found.

Name

MrmCloseHierarchy – close an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmCloseHierarchy (MrmHierarchy hierarchy)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().

Returns

MrmSUCCESS	On success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmFAILURE	On failure.

Description

`MrmCloseHierarchy`() closes an Mrm hierarchy that has been previously opened with a call to `MrmOpenHierarchy`() or `MrmOpenHierarchyPerDisplay`(). The UID files associated with the *hierarchy* are closed and the memory used by the *hierarchy* is freed. However, as of Motif 1.2, the memory used by Mrm to register any values or procedures with `MrmRegisterNamesInHierarchy`() is not freed.

Usage

An application calls `MrmCloseHierarchy`() when it is done accessing an Mrm hierarchy in order to free file descriptions and memory consumed by the hierarchy. As of Motif 1.2, this function cannot fail; it always returns MrmSUCCESS or MrmBAD_HIERARCHY.

Example

The following code fragment illustrates the use of `MrmCloseHierarchy`():

```
...
extern MrmHierarchy hierarchy; /* Previously opened Mrm hierarchy. */

if (MrmCloseHierarchy (hierarchy) != MrmSUCCESS)
    error_handler();
hierarchy = NULL;          /* Protect from future misuse. */
...
```

See Also

`MrmOpenHierarchy`(3), `MrmOpenHierarchyPerDisplay`(3).

Name

MrmFetchBitmapLiteral – retrieve an exported bitmap from an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmFetchBitmapLiteral[1] (MrmHierarchy hierarchy,
                                   String       name,
                                   Screen       *screen,
                                   Display      *display,
                                   Pixmap       *pixmap,
                                   Dimension    *width,
                                   Dimension    *height)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
name	Specifies the name of an icon to retrieve as a bitmap.
screen	Specifies the screen of the display on which the pixmap is created.
display	Specifies the display on which the pixmap is created.

Outputs

pixmap	Returns the specified bitmap as a pixmap of depth 1 on the specified screen of the specified display.
width	Returns the width of the pixmap.
height	Returns the height of the pixmap.

Returns

MrmSUCCESS	On success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmNOT_FOUND	If the icon is not found.
MrmWRONG_TYPE	If the named value is not an icon.
MrmNOT_VALID	If the icon uses a color table which contains colors other than foreground- color and back- ground- color.
MrmFAILURE	On failure.

Availability

Motif 1.2 and later.

Description

1.Erroneously given as MrFetchBitmapLiteral in 1st edition.

`MrmFetchBitmapLiteral`() retrieves the named icon and converts it to a pixmap of depth 1 on the specified *screen* of the specified *display*. The icon must be defined as an exported value in a UIL source module. Foreground color pixels in the icon are set to 1 in the pixmap and background color pixels in the icon are set to 0 (zero) in the pixmap. The application is responsible for freeing the pixmap using `XmFreePixmap`().

Usage

An icon retrieved with `MrmFetchBitmapLiteral`() can only use the special colors foreground color and background color in its color table. If the color table contains any other colors, `MrmFetchBitmapLiteral`() fails and returns MrmNOT_VALID.

As of Motif 1.2, values of type xbitmapfile cannot be converted to a pixmap using this function. xbitmapfile values can only be retrieved using `MrmFetch-IconLiteral`().

Example

The following UIL and C code fragments show the retrieval of a bitmap from an Mrm hierarchy:

```
UIL:
    ...
    ! Declare a cursor icon using the default color table.
     value
        resize_down : exported icon (   '********',
                                        "   **   ',
                                         '   **   ',
                                        '** ** **',
                                        ' ****** ',
                                        '   **   ');
        ...
C:
    ...
    extern MrmHierarchy    hierarchy;   /* Previously opened hierarchy. */
    extern Widget          w;           /* Previously created widget. */

    Pixmap          cursor_bits;
    Dimension       width, height;
    Cardinal        status;
    static XColor   white = { 0, ~0, ~0, ~0, DoRed | DoGreen | DoBlue };
    static XColor   black = { 0, 0, 0, 0, DoRed | DoGreen | DoBlue };

    /* Get the icon as a pixmap of depth 1. */
```

```
status = MrmFetchBitmapLiteral (hierarchy, "resize_down", XtScreen
                                (w), XtDisplay (w), &cursor_bits,
                                &width, &height);
if (status != MrmSUCCESS)
    error_handler();
else {
    /* Create a cursor using the pixmap. */
    cursor = XCreatePixmapCursor (XtDisplay (w), cursor_bits,
                                  cursor_bits, &black, &white,
                                  width/2, height-1);
    /* Set the cursor in the widget. */
    XDefineCursor (XtWindow (w), cursor);
}
...
```

See Also

`MrmFetchIconLiteral`(3), `MrmFetchLiteral`(3), `value`(5), `color_table`(6), `icon`(6), `xbitmapfile`(6).

Name

MrmFetchColorLiteral – retrieve an exported color value from an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmFetchColorLiteral (MrmHierarchy    hierarchy,
                               String          name,
                               Display         *display,
                               Colormap        colormap,
                               Pixel           *pixel)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
name	Specifies the name of the color to retrieve.
display	Specifies the display.
colormap	Specifies the colormap in which the color is allocated.

Outputs

pixel	Returns a pixel value for the named color.

Returns

MrmSUCCESS	On success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmNOT_FOUND	If the specified color is not found or cannot be allocated.
MrmWRONG_TYPE	If the named value is not a color or rgb value.
MrmFAILURE	On failure.

Description

`MrmFetchColorLiteral`() retrieves a named color value and attempts to allocate a color cell containing it. The color must be defined as an exported value in a UIL source module. The color cell is allocated with `XAllocColor`() if the type of the value is rgb or with XAllocNamedColor if the type of the value is color. The *colormap* argument is used as a parameter to these functions. If *colormap* is NULL, Mrm uses the colormap returned by the `DefaultColormap`() macro.

Usage

If the color cannot be allocated because the specified *colormap* is full, `MrmFetchColorLiteral`() fails and returns MrmNOT_FOUND, not MrmFAILURE. The OSF documentation claims that when a color cannot be allocated, black or white is substituted. This was not true of Motif 1.2 variants: this translation did not take place, and you had to handle the error yourself. In Motif 2.1, however, MrmFetchColorLiteral() most certainly does substitute `XBlackPixelOfScreen`() if `XAllocColor`() fails; it does not use `XWhitePixelOfScreen`().

Example

The following UIL and C code fragments show the retrieval of color values from an Mrm hierarchy:

UIL:

```
...
value
foreground : exported rgb (255, 167, 0);
background : exported color ('mutant ninja turtle');
...
```

C:

```
Widget          toplevel;       /* Previously created widget. */
MrmHierarchy    hierarchy;      /* Previously opened Mrm hierarchy. */
Pixel           foreground, background;
Cardinal        status;
...
status = MrmFetchColorLiteral (hierarchy, "foreground", XtDisplay
(toplevel),
                                NULL, &foreground);
if (status != MrmSUCCESS)
    error_handler();
status =    MrmFetchColorLiteral (hierarchy, "background", XtDisplay
(toplevel),
                                    NULL, &background);
if (status != MrmSUCCESS)
    error_handler();
...
```

See Also

`MrmFetchBitmapLiteral`(3), `MrmFetchIconLiteral`(3), `MrmFetchLiteral`(3), `value`(5), `color`(6), `color_table`(6), `rgb`(6).

Name

MrmFetchIconLiteral – retrieve an exported icon from an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmFetchIconLiteral ( MrmHierarchy   hierarchy,
                               String         name,
                               Screen         *screen,
                               Display        *display,
                               Pixel          foreground,
                               Pixel          background,
                               Pixmap         *pixmap)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
name	Specifies the name of an icon or xbitmapfile to retrieve.
screen	Specifies the screen of the display on which the pixmap is created.
display	Specifies the display.
foreground	Specifies the foreground color to use for the pixmap.
background	Specifies the background color to use for the pixmap.

Outputs

pixmap	Returns a pixmap created on the specified screen and display.

Returns

MrmSUCCESS	On success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmNOT_FOUND	If the specified icon or xbitmapfile is not found or a color in the icon's color table cannot be allocated.
MrmWRONG_TYPE	If the named value is not an icon or xbitmapfile value.
MrmFAILURE	On failure.

Description

`MrmFetchIconLiteral`() retrieves the named icon or xbitmapfile value and attempts to convert it to a pixmap on the specified *screen* of the display. The icon or xbitmap-file must be defined as an exported value in a UIL source module. The *foreground* pixel argument is used as the color for foreground pixels in an icon and pixels set to 1 in an xbitmapfile. The *background* pixel argument is used as the color for background pixels in an icon and pixels set to 0 (zero) in an xbitmapfile. Additional colors used by an icon are allocated in the colormap returned by the `DefaultColormap()` macro. The application is responsible for freeing the pixmap using `XFreePixmap()`.

Usage

If a color cannot be allocated because the specified colormap is full, `MrmFetchIconLiteral()` fails and returns MrmNOT_FOUND, not MrmFAILURE. The OSF documentation claims that when a color cannot be allocated, black or while is substituted, but in Motif 1.2 this translation did not take place, so you had to handle the error yourself. In Motif 2.1, `XBlackPixelOfScreen()` is used as a substitute if `XAllocColor()` fails; it does not use a corresponding `XWhitePixelOfScreen()`.

Example

The following UIL and C code fragments illustrate the retrieval of a pixmap from an Mrm hierarchy:

```
UIL:
    ...
    ! Declare an icon using the default color table
    value
        box : exported icon (   '****',
                                '*  *',
                                '*  *',
                                '****');
        ...

C:
    extern MrmHierarchy   hierarchy;          /* Previously opened   */
                                              /*     hierarchy. */
    extern Widget         drawing_area;       /* Previously created  */
                                              /*     widget.   */
    extern GC             drawing_area_gc;    /* Previously defined   */
                                              /*    graphics context. */
    Pixel                 foreground, background;
    Pixmap                box_pixmap;
    unsigned int          box_width, box_height;
    unsigned int          dont_care;
    Cardinal              status;

    /* Get values to use for pixmap foreground and background. */
    XtVaGetValues (drawing_area, XmNforeground, &foreground,
                                 XmNbackground, &background,
                                 NULL);
    /* Create the pixmap from the box icon in the hierarchy. */
```

```
status = MrmFetchIconLiteral (hierarchy, "box", XtScreen
(drawing_area),
                              XtDisplay (drawing_area), fore-
                              ground, background, &box_pixmap);
if (status != MrmSUCCESS)
    error_handler();
else {
    /* Get the size of the pixmap. */
    XGetGeometry (XtDisplay (drawing_area), box_pixmap, (Window
    *) &dont_care,
                 (int *) &dont_care, (int *) &dont_care,
                 &box_width, &box_height, &dont_care,
                 &dont_care);
    /* Draw the box in the drawing area. */
    XCopyArea (XtDisplay (drawing_area), box_pixmap, XtWindow
               (drawing_area),         drawing_area_gc, 0, 0,
               box_width, box_height, 10, 10);
    /* Free the pixmap. */
    XFreePixmap (box_pixmap);
}
```

See Also

`MrmFetchBitmapLiteral`(3), `MrmFetchColorLiteral`(3), `MrmFetchLiteral`(3), `value`(5), `color`(6), `color_table`(6), `icon`(6), `rgb`(6), `xbitmapfile`(6).

Name

MrmFetchLiteral – retrieve an exported value from an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmFetchLiteral (MrmHierarchy    hierarchy,
                          String          name,
                          Display         *display,
                          XtPointer       *value,
                          MrmCode         *type)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
name	Specifies the name of the value to retrieve.
display	Specifies the display.

Outputs

value	Returns a pointer to the value with the specified name.
type	Returns the type of the value retrieved.

Returns

MrmSUCCESS	On success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmNOT_FOUND	If the specified value is not found.
MrmWRONG_TYPE	If the type of the value specified cannot be converted by this procedure.

Description

`MrmFetchLiteral`() retrieves the named value and its type from the specified Mrm hierarchy. The value must be defined as an exported value in a UIL source module. The *display* argument is used to convert values of type font, fontset, and font_table. On success, this routine returns a pointer to the named value and the type of the value. The possible type values begin with MrmR*type* and are defined in the include file <*Mrm/MrmPublic.h*>. The application is responsible for freeing the returned value, except when it is a font or a fontset. font and fontset values are cached by Mrm and freed when the display is closed.

Usage

`MrmFetchLiteral`() cannot be used to retrieve values of certain types. You should retrieve icon values with `MrmFetchIconLiteral`() or `MrmFetchBitmapLiteral`(), xbitmapfile values with MrmFetchIconLiteral(), and color or rgb values with `MrmFetchColorLiteral`().

The storage allocated by Mrm for a boolean value is sizeof(int) not sizeof(Boolean). Because sizeof(Boolean) is less than sizeof(int) on many systems, applications should use an int pointer rather than a Boolean pointer as the value argument when retrieving a boolean.

Example

The following UIL and C code fragments illustrate the use of MrmFetchLiteral() to fetch various values from an Mrm hierarchy:

UIL:

```
    ...
    value
        int_val       : 10;
        string_val  : ‘okemo’;
        ...
```

C:

```
    ...
    extern MrmHierarchy       hierarchy;       /* Previously opened hierarchy. */
    extern Display                *display;        /* Previously opened display.   */
    int                                *int_ptr;
    String                           string;
    MrmCode                       type;
    Cardinal                         status;

    status = MrmFetchLiteral (hierarchy, "int_val", display, (XtPointer *)
    &int_ptr, &type);

    if (status != MrmSUCCESS || type != MrmRtypeInteger)
        error_handler();
    else
        printf ("Fetched integer %d\n", *int_ptr);

    status = MrmFetchLiteral (hierarchy, "string_val", display, (XtPointer[1] *)
    &string, &type);

    if (status != MrmSUCCESS || type != MrmRtypeCString)
        error_handler();
    else
        printf ("Fetched string ’%s’\n", string);
    ...
```

1.Erroneously given as XtPoitner in 1st edition.

See Also

`MrmFetchBitmapLiteral`(3), `MrmFetchColorLiteral`(3), `MrmFetchIconLiteral`(3), `MrmFetchSetValues`(3), `value`(5), `asciz_string_table`(6), `boolean`(6), `class_rec_name`(6), `color`(6), `compound_string`(6), `compound_string_table`(6), `float`(6), `font`(6), `font_table`(6), `fontset`(6), `icon`(6), `integer`(6), `integer_table`(6), `keysym`(6), `rgb`(6), `single_float`(6), `string`(6), `translation_table`(6), `wide_character`(6), `widget`(6), `xbitmapfile`(6).

Name

MrmFetchSetValues – set widget resources to values retrieved from an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmFetchSetValues ( MrmHierarchy    hierarchy,
                             Widget          widget,
                             ArgList         arg_list,
                             Cardinal        num_args)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
widget	Specifies the object whose resources are modified.
arg_list	Specifies an array of name/UID-value pairs to be set.
num_args	Specifies the number of elements in arg_list.

Returns

MrmSUCCESS	On success.
MrmPARTIAL_SUCCESS	On partial success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmFAILURE	On failure.

Description

`MrmFetchSetValues`() sets the resources for an widget to named values obtained from the specified Mrm *hierarchy.* If a named value is not found or cannot be converted, the resource corresponding to that value is not set. If all the named values in *arg_list* are successfully retrieved, `MrmFetchSetValues`() returns MrmSUCCESS. If some values are successfully retrieved and others are not, MrmPARTIAL_SUCCESS is returned. If no values are successfully retrieved, MrmFAILURE is returned. When at least one value is successfully retrieved, `XtSetValues`() is called to modify the resources of object.

Usage

`MrmFetchSetValues`() sets the resources named in the name member of each item in *arg_list* to the value from the Mrm hierarchy named by the value member. This use differs from `XtSetValues`(), in that value member contains the name of a value to retrieve, not the value itself. Each named value must be defined as an exported value in a UIL source module.

The conversion of certain types may require a display pointer, screen pointer, background color, or foreground color. When these values are needed, Mrm obtains them from widget. If foreground and background colors are needed for a conversion and widget does not have a background or foreground resource, Mrm uses black or white instead. If foreground and background colors are needed for a conversion and the XmNbackground or XmNforeground resources are specified in arg_list, they are used instead of the foreground and background of widget. As a result, if both an icon and foreground and/or background values are specified in the same argument list, the icon uses the colors specified in the list, rather than the colors of the widget.

Example

The following UIL and C code fragments illustrate the use of `MrmFetchSet-Values`() to fetch a resource value from an Mrm hierarchy:

UIL:

```
...
value
! English language version of the confirm quit message:
confirm_quit_msg : ’Do you really want to quit?’;
...
```

C:

```
extern MrmHierarchy    hierarchy;         /* Previously opened Mrm   */
                                          /*     hierarchy. */
extern Widget          yes_no_dialog;     /* Previously created yes/no  */
                                          /*     dialog. */

void DisplayConfirmQuit (void)
{
    static Arg args[] = {
        { XmNmessageString, (XtArgVal) "confirm_quit_msg" }
    };

    /* Set the message string for confirm quit. */
    MrmFetchSetValues (hierarchy, yes_no_dialog, args, XtNumber (args));
    /* Make the dialog appear. */
    XtManageChild (yes_no_dialog);
}
```

Structures

ArgList is defined as follows:

```
typedef struct {
    String        name;
    XtArgVal      value;
} Arg, *ArgList;
```

See Also

`MrmFetchBitmapLiteral`(3), `MrmFetchColorLiteral`(3), `MrmFetchIconLiteral`(3), `MrmFetchLiteral`(3), `value`(5).

Name

MrmFetchWidget – create the widget tree rooted at a named widget.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmFetchWidget ( MrmHierarchy    hierarchy,
                          String          name,
                          Widget          parent,
                          Widget          *widget,
                          MrmType         *class)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
name	Specifies the name of the root widget of the widget tree to create.
parent	Specifies the parent of the root widget.

Outputs

widget	Returns the widget ID of the root widget.
class	Returns the UID class code for the widget class of the root widget.

Returns

MrmSUCCESS	On success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmNOT_FOUND	If the specified widget is not found.
MrmFAILURE	On failure.

Description

`MrmFetchWidget`() creates the named *widget* and recursively creates all of its children. Each child is managed by Mrm, unless declared unmanaged in the UIL source module. The root widget should be defined as exported in a UIL source module. Mrm supports the MrmNcreateCallback, which if defined, is called after a widget is created. The prototype of an MrmNcreateCallback is the same as any other Xt callback procedure. The call_data passed to the callback is an XmAnyCallbackStruct.

Usage

Each successful call to `MrmFetchWidget`() results in the creation of a new widget tree, even if *name* has been fetched previously. As a result, you can use a widget tree definition from an Mrm hierarchy as a template for creating multiple instances of the same widget tree. The widget at the root of the tree is not managed by Mrm, so your application must manage this widget to make the tree visible.

In Motif 1.2 and earlier, `MrmFetchWidget`() returns MrmSUCCESS if the root widget is retrieved successfully, even if one or more of its children are not. As of Motif 1.2.1, if `MrmFetchWidget`() cannot find a child widget, it returns MrmNOT_FOUND and does not create any widgets.

As of Motif 1.2, the possible MrmType values returned in class are not defined in any of the Mrm include files, although the OSF documentation claims that they are defined in <*Mrm/Mrm.h*>. If you need to check the widget class of a widget created with `MrmFetchWidget`(), use `XtClass`() or one of the XmIs*() macros.

Example

The following UIL and C code fragments illustrate the retrieval of a widget hierarchy from an Mrm hierarchy:

UIL:

```
...
! Define a simple widget tree, with form at the root.
object label      : XmLabel { };
object button     : XmPushButton { };
object form       : exported XmForm {
      controls {
            XmLabel           label;
            XmPushButton      button;
      };
};
...
```

C:

```
extern Widget            toplevel;         /* Previously defined widget. */
extern MrmHierarchy      hierarchy;        /* Previously opened hierarchy. */
Widget                   form;
MrmType                  class;
Cardinal                 status;

status = MrmFetchWidget (hierarchy, "form", toplevel, &form, &class);

if (status != MrmSUCCESS)
      error_handler();
...
```

Structures

The MrmNcreateCallback function is passed an XmAnyCallbackStruct, which is defined as follows:

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

See Also

`MrmFetchWidgetOverride`(3), `MrmOpenHierarchy`(3), `MrmOpenHierarchyPerDisplay`(3), `object`(5), `widget`(6).

Name

MrmFetchWidgetOverride – create the widget tree rooted at a named widget and override the resources set in the UID file.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmFetchWidgetOverride (MrmHierarchy    hierarchy,
                                 String          name,
                                 Widget          parent,
                                 String          override_name,
                                 ArgList         arg_list,
                                 Cardinal        num_args,
                                 Widget          *widget,
                                 MrmType         *class)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
name	Specifies the name of the root widget of the widget tree to create.
parent	Specifies the parent of the root widget.
override_name	Specifies the name to use when creating the root widget. If NULL, name is used.
arg_list	Specifies an array of resource/value pairs to set on the root widget when it is created. If NULL, no resources are set.
num_args	Specifies the number of elements in arg_list. Must be 0 (zero) if arg_list is NULL.

Outputs

widget	Returns the widget ID of the root widget.
class	Returns the UID class code for the widget class of the root widget.

Returns

MrmSUCCESS	On success.
MrmBAD_HIERARCHY	If hierarchy is NULL or does not point to a valid Mrm hierarchy.
MrmNOT_FOUND	If the specified widget is not found.
MrmFAILURE	On failure.

Description

`MrmFetchWidgetOverride`() creates the named *widget* and recursively creates all of its children. The root widget should be defined as exported in a UIL source module. *arg_list* is used to specify additional resource/value pairs that override those specified in the widget definition in a UIL source module. Each child is managed by Mrm unless declared unmanaged in the UIL source module. Mrm supports the MrmNcreateCallback, which if defined, is called after a widget is created. The prototype of an MrmNcreateCallback is the same as any other Xt callback procedure. The call_data passed to the callback is an XmAnyCallback-Struct.

Usage

`MrmFetchWidgetOverride`() allows an application to create a widget defined in an Mrm hierarchy while specifying application-defined resource values that can supplement or override those specified in the UIL definition. The function sets the resources of the root widget that are named in the name member of each item in *arg_list* to value specified in the value member. The resource of any children of the root widget are not affected.

Each successful call to `MrmFetchWidgetOverride`() results in the creation of a new widget tree, even if *name* has been fetched previously. As a result, you can use a widget tree definition from an Mrm hierarchy as a template for creating multiple instances of the same widget tree. The widget at the root of the tree is not managed by Mrm, so your application must manage this widget to make the tree visible.

In Motif 1.2 and earlier, `MrmFetchWidget`() returns MrmSUCCESS if the root widget is retrieved successfully, even if one or more of its children are not. As of Motif 1.2.1, if `MrmFetchWidget`() cannot find a child widget, it returns MrmNOT_FOUND and does not create any widgets.

As of Motif 1.2, the possible MrmType values returned in class are not defined in any of the Mrm include files, although the OSF documentation claims that they are defined in <*Mrm/Mrm.h*>. If you need to check the widget class of a widget created with `MrmFetchWidgetOverride`(), *use* `XtClass`*()* or one of the XmIs*() macros.

Example

The following UIL and C code fragments illustrate the retrieval of a widget hierarchy from an Mrm hierarchy using MrmFetchWidgetOverride()[1]:

```
UIL:
    ...
    object error_dialog: exported XmErrorDialog {
        arguments {
            XmNmessageString = "If you can read this, file a bug report.";
            XmNdialogStyle =
            XmDIALOG_FULL_APPLICATION_MODAL;
```

1.Erroneously given as MwmFetchWidgetOverride() in 1st edition.

```
        };
    };
    ...
C:
    extern Widget          toplevel;     /* Previously created widget. */
    extern MrmHierarchy    hierarchy;    /* Previously opened hierarchy. */

    void display_error (String message)
    {
        Arg         arg_list[1];
        XmString    s;
        Cardinal    status;
        Widget      error_dialog;
        MrmType     class;

        s = XmStringCreateLocalized (message);
        XtSetArg (arg_list[0], XmNmessageString, s);
        status = MrmFetchWidgetOverride (hierarchy, "error_dialog",
                                         toplevel, "error_dialog",
                                         arg_list, 1, &error_dialog,
                                         &class);
        XmStringFree (s);

        if (status != MrmSUCCESS)
            handle_error();
        else
            XtManageChild (error_dialog);
    }
```

Structures

ArgList is defined as follows:

```
typedef struct {
    String      name;
    XtArgVal    value;
} Arg, *ArgList;
```

The MrmNcreateCallback function is passed an XmAnyCallbackStruct, which is defined as follows:

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

See Also

`MrmFetchWidget`(3), `MrmOpenHierarchy`(3),
`MrmOpenHierarchyPerDisplay`(3), `object`(5), `widget`(6).

Name

MrmInitialize – prepare the Mrm library for use.

Synopsis

```
#include <Mrm/MrmPublic.h>

void MrmInitialize (void)
```

Description

`MrmInitialize`() initializes the Mrm library. As part of the initialization, all Motif widget classes are registered in the Mrm widget class database with `MrmRegisterClass`().

Usage

Applications should call `MrmInitialize`() before the Xt Toolkit is initialized and before calling any other Mrm functions. If the routine is not called before `MrmOpenHierarchyPerDisplay`(), future calls to `MrmFetchWidget`() and `MrmFetchWidgetOverride`() will fail. Applications should only call `MrmInitialize`() once.

Example

The following code fragment illustrates the use of `MrmInitialize`(): [1]

```
...
Widget          toplevel;
XtAppContext    app_context;
MrmHierarchy    hierarchy;
Cardinal        status;

XtSetLanguageProc (NULL, (XtLanguageProc) NULL, NULL);

MrmInitialize();

toplevel = XtVaOpenApplication (&app_context, "App", NULL, 0, (Cardi-
                                nal *) &argc, &argv, NULL, session-
                                ShellWidgetClass, NULL);
...
```

See Also

`MrmFetchWidget`(3), `MrmFetchWidgetOverride`(3), `MrmOpenHierarchy`(3), `MrmOpenHierarchyPerDisplay`(3), `MrmRegisterClass`(3).

1.From X11R6, XtAppInitialize() is marked as obsolete. The SessionShell is only available from X11R6 onwards, and it replaces the deprecated ApplicationShell widget class.

Name

MrmOpenHierarchy – open an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmOpenHierarchy ( MrmCount          num_files,
                            String            file_name_list[],
                            MrmOsOpenParamPtr *os_params,
                            MrmHierarchy      *hierarchy)
```

Inputs

num_files	Specifies the number of files in file_name_list.
file_name_list	Specifies an array of UID file names to associate with the hierarchy.
os_params	Specifies operating system dependent settings.

Outputs

hierarchy	Returns an open Mrm hierarchy consisting of the specified files.

Returns

MrmSUCCESS	On success.
MrmNOT_FOUND	If one or more files cannot be opened.
MrmNOT_VALID	If the version of Mrm is older than the version of any UID file.
MrmDISPLAY_NOT_OPENED	If a display pointer cannot be found.
MrmFAILURE	On failure.

Availability

In Motif 1.2, `MrmOpenHierarchy`() is obsolete. It has been superseded by `MrmOpenHierarchyPerDisplay`().

Description

`MrmOpenHierarchy`() opens an Mrm hierarchy consisting of one or more UID files. This routine is similar to `MrmOpenHierarchyPerDisplay`(), except that it does not take a display parameter. `MrmOpenHierarchy`() is retained for compatibility with Motif 1.1 and should not be used in newer applications.

Usage

`MrmOpenHierarchy`() relies on the Motif widget library to locate a display pointer. To ensure that a display pointer can be found, an application must create an ApplicationShell before calling `MrmOpenHierarchy`(). The display pointer is used as a parameter to `XtResolvePathname`(), which locates the files in *file_name_list*. If an application creates multiple ApplicationShells on different displays, the display pointer chosen by this routine is undefined.

See the `MrmOpenHierarchyPerDisplay`() manual page for a full explanation of the process of opening an Mrm hierarchy, including the search path that is used to find the UID files.

See Also

`MrmCloseHierarchy`(3), `MrmOpenHierarchyPerDisplay`(3).

Name

MrmOpenHierarchyFromBuffer – open an Mrm hierarchy from a buffer

Synopsis

#include <Mrm/MrmPublic.h>

Cardinal MrmOpenHierarchyFromBuffer (unsigned char **buffer*, MrmHierarchy **hierarchy_id*)

Inputs

buffer	Specifies a stream of bytes representing a UID file contents.

Outputs

hierarchy_id	Returns an open Mrm hierarchy.

Returns

MrmSUCCESS	On Success.
MrmNOT_VALID	If the version of Mrm is older than the data contained within the buffer.
MrmDISPLAY_NOT_OPENED	If a display pointer cannot be found.
MrmFAILURE	On failure.

Availability

Motif 2.0 and later.

Description

`MrmOpenHierarchyFromBuffer`() opens an Mrm hierarchy using the stream of data specified by buffer, which is presumably the contents of a previously opened UID file loaded into memory. It could, however, be dynamically constructed.

Usage

`MrmOpenHierarchyFromBuffer`() relies on the Motif widget library to locate a display pointer using internal default values. A pointer is only found if a ApplicationShell has been created before calling `MrmOpenHierarchyFromBuffer`().

See Also

`MrmOpenHierarchy`(3), `MrmOpenHierarchyPerDisplay`(3), `MrmCloseHierarchy`(3).

Name

MrmOpenHierarchyPerDisplay – open an Mrm hierarchy.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmOpenHierarchyPerDisplay (Display          *display,
                                     MrmCount         num_files,
                                     String           file_name_list[],
                                     MrmOsOpenParamPtr os_params_list[],
                                     MrmHierarchy     *hierarchy)
```

Inputs

display	Specifies the display.
num_files	Specifies the number of files in file_name_list.
file_name_list	Specifies an array of file names to associate with the hierarchy.
os_params_list	Specifies an array of operating system dependent settings.

Outputs

hierarchy	Returns an open Mrm hierarchy consisting of the specified files.

Returns

MrmSUCCESS	On success.
MrmNOT_FOUND	If one or more files cannot be opened.
MrmNOT_VALID	If the version of Mrm is older than version of the any UID file.
MrmDISPLAY_NOT_OPENED	If a display pointer cannot be found.
MrmFAILURE	On failure.

Availability

Motif 1.2 and later.

Description

`MrmOpenHierarchyPerDisplay`() opens an Mrm hierarchy consisting of one or more UID files. An Mrm hierarchy must be opened before any values are retrieved or widgets created with the MrmFetch*() routines. When an Mrm hierarchy is successfully opened, each UID file specified in *file_name_list* is opened and consumes a file descriptor. No files are opened if a value other than MrmSUCCESS is returned. The UID files are subsequently closed when the hierarchy is closed with `MrmCloseHierarchy`(). As of Motif 1.2, settings in the *os_params_list* parameter are only useful to the UIL compiler. Application programs should always specify NULL for this argument.

Usage

The MrmFetch*() routines retrieve a named value or widget by searching the UID files for a hierarchy in the order that they are specified in *file_name_list*. If a named value or widget occurs in more than one of the UID files, the value is retrieved from the file that occurs first in the array. Once an Mrm hierarchy has been opened, the UID files associated with the hierarchy must not be modified or deleted until the hierarchy is closed.

Files specified in *file_name_list* may be full or partial path names. When a file name starts with a slash (/), it specifies a full path name and `MrmOpenHierarchyPerDisplay`() opens the file. Otherwise, the file name specifies a partial path name which causes `MrmOpenHierarchyPerDisplay`() to look for the file using a search path.

`XtResolvePathname`() is used to locate the file in the search path. The UIDPATH environment variable specified the search path for UID files. Each directory in the search path can contain the substitution character %U; the partial path name specified by file_name_list is substituted for %U.   In addition, the path can also use the substitution characters accepted by `XtResolvePathname()`. The path is first searched with %S mapped to *.uid*. If the file is not found the path is searched again with %S mapped to NULL.

If UIDPATH is not set, `MrmOpenHierarchyPerDisplay`() uses a default search path. If the XAPPLRESDIR environment variable is set, the routine searches the following path; the class name of the application is substituted for %N, the language string of the display argument is substituted for %L, and the language component of the language string is substituted for %l.

$XAPPLRESDIR/%L/uid/%N/%U%S
$XAPPLRESDIR/%l/uid/%N/%U%S
$XAPPLRESDIR/uid/%N/%U%S
$XAPPLRESDIR/%L/uid/%U%S
$XAPPLRESDIR/%l/uid/%U%S
$XAPPLRESDIR/uid/%U%S
$HOME/uid/%U%S
$HOME/1%U%S
/usr/lib/X11/%L/uid/%N/%U%S
/usr/lib/X11/%l/uid/%N/%U%S
/usr/lib/X11/uid/%N/%U%S
/usr/lib/X11/%L/uid/%U%S
/usr/lib/X11/%l/uid/%U%S
/usr/lib/X11/uid/%U%S
/usr/include/X11/uid/%U%S

If XAPPLRESDIR is not set, `MrmOpenHierarchyPerDisplay`() searches the same path, except that XAPPLRESDIR is replaced by HOME. These paths are vendor-dependent and a vendor may use different directories for /usr/lib/X11 and /usr/include/X11.

Example

The following code fragment illustrates the use of `MrmOpenHierarchyPer-Display`():[1]

```
...
MrmHierarchy    hierarchy;
XtAppContext    app_context;
Widget          toplevel;
String          uid_files[] = { "/usr/lib/app/app", "strings" };
Cardinal        status;

XtSetLanguageProc (NULL, NULL, NULL);

MrmInitialize();
toplevel = XtVaOpenApplication (&app_context, "App", NULL, 0, &argc,
                    argv, NULL, sessionShellWidgetClass, NULL);

status = MrmOpenHierarchyPerDisplay (XtDisplay (toplevel),
                                      XtNumber (uid_files), uid_files,
                                      NULL, &hierarchy);

if (status != MrmSUCCESS)
    error_handler();
...
```

See Also

`MrmCloseHierarchy`(3), `MrmFetchBitmapLiteral`(3), `MrmFetchColorLiteral`(3), `MrmFetchIconLiteral`(3), `MrmFetchLiteral`(3), `MrmFetchWidget`(3), `MrmFetchWidgetOverride`(3).

1. From X11R6, XtAppInitialize() is marked as obsolete. The SessionShell is only available from X11R6 onwards, and it replaces the deprecated ApplicationShell widget class.

Name

MrmRegisterClass – register a widget creation function for a non-Motif widget.

Synopsis

```
#include <Mrm/MrmPublic.h>

Cardinal MrmRegisterClass (MrmType       class_code,
                           String        class_name,
                           String        create_proc_name,
                           Widget        (*create_proc) (Widget, char *,
ArgList, Cardinal),
                           WidgetClass   widget_class)
```

Inputs

class_code	This argument is obsolete and is ignored.
class_name	This argument is obsolete and is ignored.
create_proc_name	Specifies the case-sensitive name of the widget creation function.
create_proc	Specifies a pointer to the widget creation procedure.
widget_class	Specifies a pointer to the widget class record or NULL.

Returns

MrmSUCCESS	On success.
MrmFAILURE	On failure.

Description

`MrmRegisterClass`() supplies Mrm with information it needs to create a user-defined widget, which is any widget that is not built into UIL and Mrm. A user-defined widget cannot be created until its class is registered.

Usage

A user-defined widget is defined in a UIL source module by specifying the *create_proc_name* in its declaration. *create_proc_name* must be all uppercase characters when used in a UIL module compiled with case-insensitive names because this setting causes the UIL compiler to store procedure name references in all uppercase characters.

If `MrmRegisterClass`() is called with a *class_name* that has been registered previously, the new *`create_proc`* and *widget_class* replace the previous values. There is no way to unregister a previously registered class. As of Motif 1.2, a small amount of memory may be leaked when a class is registered multiple times.

The *widget_class* argument allows Mrm to convert a class name specified in a UIL *class_rec_name* literal into a widget class pointer. If NULL is specified, the widget class pointer is not accessible with the *class_rec_name* type.

Example

The following UIL and C code fragments illustrate the creation of an instance of the Athena panner widget from UIL. Like any other widget defined in a UIL module, it is created with a call to MrmFetchWidget() or `MrmFetchWidget-Override`():

UIL:

```
...
procedure XawCreatePannerWidget;

object panner : user_defined procedure XawCreatePannerWidget { };
...
```

C:

```
 Widget XawCreatePannerWidget (Widget parent, String name, ArgList
                    args, Cardinal num_args)
{
     return XtCreateWidget (name, pannerWidgetClass, parent, args,
                    num_args);
}
...
MrmRegisterClass (0, NULL,
                         "XawCreatePannerWidget",
                         XawCreatePannerWidget,
                         &pannerWidgetClass);
...
```

Procedures

The create_proc parameter has the following syntax:

Widget create_proc (Widget *parent*, String *name*, ArgList *args*,

Cardinal *num_args*)

The procedure takes four arguments. The first, *parent*, is the parent of the widget that is being created. *name* is the name of the widget. The last two arguments, *args* and *num_args*, specify the initial resource settings for the widget. The procedure returns the widget ID of the newly created widget.

See Also

`MrmInitialize`(3), `MrmFetchWidget`(3), `MrmFetchWidgetOverride`(3), `object`(5), `class_rec_name`(6).

Name

MrmRegisterNames – register application-defined values and procedures.

Synopsis

#include <Mrm/MrmPublic.h>

Cardinal MrmRegisterNames (MrmRegisterArgList *name_list*, MrmCount *count*)

Inputs

name_list	Specifies an array of name/value pairs to be registered with Mrm.
count	Specifies the number of elements in name_list.

Returns

MrmSUCCESS	On success.
MrmFAILURE	On failure.

Description

`MrmRegisterNames`() registers an array of name/value pairs that are used as identifiers and procedures in a UIL source module. Names registered with this routine are accessible from any open Mrm hierarchy. By contrast, names registered with `MrmRegisterNamesInHierarchy`() are only accessible from the hierarchy in which they are registered.

If `MrmRegisterNames`() is called with a name that has been registered previously, the old value associated with the name is replaced by the new value. There is no way to unregister a previously registered name.

Usage

The MrmRegisterArg structure consists of a name and an associated value. The case of name is significant. name must be in all uppercase characters if name is used in a UIL module compiled with case-insensitive names, because this setting causes the UIL compiler to store procedure and identifier name references in all uppercase characters.

The *name_list* array can contain names that represent both callback procedures and identifier values. A procedure value in *name_list* should be a pointer to a function of type XtCallbackProc. An identifier value is any application-defined value that is exactly sizeof (XtPointer). Mrm makes no distinction between procedures and identifiers, although an application may organize them in two separate arrays for clarity. A distinction is made in a UIL source module, where any name used must be declared as either a procedure or an identifier.

Procedures and identifiers must be registered with `MrmRegisterNames`() or `MrmRegisterNamesInHierarchy`() before an application attempts to create a widget that references them. Mrm converts a procedure or identifier reference to a value by first searching hierarchy-local names registered with `MrmRegisterNamesInHierarchy`(). If the value is not found, the search continues with global names registered with `MrmRegisterNames`().

Example

The following UIL and C code fragments illustrate the use of `MrmRegisterNames`():

```
UIL:
    ...
    identifier user_id;
    procedure activate();

    object button : XmPushButton {
        callbacks {
            XmNactivateCallback = procedure activate;
        };
    };
    ...

C:
    ...
    extern XtCallbackProc activate;
    int                 user_id = getuid();
    MrmRegisterArg      names[2];

    names[0].name = "activate";
    names[0].value = (XtPointer) activate;
    names[1].name = "user_id";
    names[1].value = (XtPointer) user_id;

    MrmRegisterNames (names, XtNumber (names));
    ...
```

Structures

MrmRegisterArgList is defined as follows:

```
typedef struct {
    String      name;     /* case-sensitive name */
    XtPointer   value;    /* procedure/value to associate with name */
} MrmRegisterArg, *MrmRegisterArglist;
```

See Also

`MrmRegisterNamesInHierarchy`(3), `identifier`(5), `procedure`(5).

Name

MrmRegisterNamesInHierarchy – register application-defined values and procedures for use in a specific UIL hierarchy.

Synopsis

#include <Mrm/MrmPublic.h>

```
Cardinal MrmRegisterNamesInHierarchy (  MrmHierarchy        hierarchy,
                                        MrmRegisterArgList  name_list,
                                        MrmCount            count)
```

Inputs

hierarchy	Specifies an Mrm hierarchy obtained from a previous call to MrmOpenHierarchy() or MrmOpenHierarchyPerDisplay().
name_list	Specifies an array of name/value pairs to be registered with Mrm.
count	Specifies the number elements in name_list.

Returns

MrmSUCCESS	On success.
MrmFAILURE	On failure.

Description

`MrmRegisterNamesInHierarchy`()[1] registers an array of name/value pairs that are used as identifiers and procedures in a UIL source module. Names registered with this routine are accessible only within the specified *hierarchy*. By contrast, names registered with `MrmRegisterNames`() are accessible from any open hierarchy.

If `MrmRegisterNamesInHierarchy`() is called with a name that has been registered previously in the same hierarchy, the old value associated with the name is replaced by the new value. There is no way to unregister a previously registered name while the hierarchy is open. However, closing the hierarchy automatically unregisters all names.

Usage

The MrmRegisterArg structure consists of a name and an associated value. The case of name is significant. name must be in all uppercase characters if name is used in a UIL module compiled with case-insensitive names, because this setting causes the UIL compiler to store procedure and identifier name references in all uppercase characters.

1.Erroneously given as MrmRegisterNames() in 1st edition.

The *name_list* array can contain names that represent both callback procedures and identifier values. A procedure value in *name_list* should be a pointer to a function of type XtCallbackProc. An identifier value is any application-defined value that is exactly sizeof (XtPointer). Mrm makes no distinction between procedures and identifiers, although an application may organize them in two separate arrays for clarity. A distinction is made in a UIL source module, where any name used must be declared as either a procedure or an identifier.

Procedures and identifiers must be registered with `MrmRegisterNames`() or `MrmRegisterNamesInHierarchy`() before an application attempts to create a widget which references them. Mrm converts a procedure or identifier reference to a value by first searching hierarchy-local names registered with `MrmRegisterNamesInHierarchy`(). If the value is not found, the search continues with global names registered with `MrmRegisterNames`().

Example

The following code fragment illustrates the use of `MrmRegisterNamesInHierarchy`():

```
/* Open a hierarchy and register it’s file name list. */

Cardinal register_and_open (Display display, MrmCount count, String *files)
{
    Cardinal            status;
    int                 *count = (int *) malloc ((unsigned) sizeof (int));
    MrmRegisterArg      names[2];
    if (count == NULL)
        return (MrmFAILURE);

    names[0].name = “file_list”;
    names[0].value = (XtPointer) file_list;
    names[1].name = “file_count”;
    names[1].value = (XtPointer) file_count;

    status = MrmOpenHierarchyPerDisplay (display, count, files, NULL,

              &hierarchy);

    if (status != MrmSUCCESS)
        return (status);

    status = MrmRegisterNamesInHierarchy (*hierarchy, names, XtNumber
    (names));

    return (status);
}
```

Structures

MrmRegisterArgList is defined as follows:

```
typedef struct {
    String       name;        /* case-sensitive name */
    XtPointer    value;       /* procedure/value to associate with name */
} MrmRegisterArg, *MrmRegisterArglist;
```

See Also

`MrmRegisterNames`(3), `identifier`(5), `procedure`(5).