# Mrm Functions


MrmCloseHierarchy

MrmFetchBitmapLiteral

MrmFetchColorLiteral

MrmFetchIconLiteral

MrmFetchLiteral

MrmFetchSetValues

MrmFetchWidget

MrmFetchWidgetOverride

MrmInitialize

MrmOpenHierarchy

MrmOpenHierarchyFromBuffer

MrmOpenHierarchyPerDisplay

MrmRegisterClass

MrmRegisterNames

MrmRegisterNamesInHierarchy

# MrmCloseHierarchy (library call)

## NAME

**MrmCloseHierarchy** -Closes a UID hierarchy

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmCloseHierarchy**(
MrmHierarchy *hierarchy_id*);

## DESCRIPTION

The **MrmCloseHierarchy** function closes a UID hierarchy previously opened by **MrmOpenHierarchyPerDisplay**. All files associated with the hierarchy are closed by the Motif Resource Manager (MRM) and all associated memory is returned.

*hierarchy_id*

Specifies the ID of a previously opened UID hierarchy. The *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

## RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmFAILURE**

The function failed.

## RELATED

**MrmOpenHierarchyPerDisplay**(3) .

# MrmFetchBitmapLiteral (library call)

## NAME

**MrmFetchBitmapLiteral** – Fetches a bitmap literal from a hierarchy

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmFetchBitmapLiteral**(
MrmHierarchy *hierarchy_id*,
String *index*,
Screen **screen*,
Display **display*,
Pixmap **pixmap_return*,
Dimension **width*,
Dimension **height*);

## DESCRIPTION

The **MrmFetchBitmapLiteral** function fetches a bitmap literal from an MRM hierarchy, and converts the bitmap literal to an X pixmap of depth 1. The function returns this pixmap and its width and height.

*hierarchy_id*

Specifies the ID of the UID hierarchy that contains the specified icon literal. The value of *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

*index*

Specifies the UIL name of the bitmap literal to fetch.

*screen*

Specifies the screen used for the pixmap. The *screen* argument specifies a pointer to the Xlib structure **Screen** which contains the information about that screen and is linked to the **Display** structure. For more information on the **Display** and **Screen** structures, see the Xlib function **XOpenDisplay** and the associated screen information macros.

*display*

Specifies the display used for the pixmap. The *display* argument specifies the connection to the X server. For more information on the **Display** structure, see the Xlib function **XOpenDisplay**.

*pixmap_return*

Returns the resulting X pixmap value. The function allocates space for this pixmap. The application is responsible for managing the allocated space. The application can recover the allocated space by calling **XmDestroyPixmap**.

*width*

Specifies a pointer to the width of the pixmap.

*height*

Specifies a pointer to the height of the pixmap.

## RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmNOT_FOUND**

The bitmap literal was not found in the hierarchy.

**MrmWRONG_TYPE**

The caller tried to fetch a literal of a type not supported by this function.

**MrmFAILURE**

The function failed.

# RELATED

**MrmFetchIconLiteral(3)** , **MrmFetchLiteral**(3) , and **XOpenDisplay**(3).

# MrmFetchColorLiteral (library call)

## NAME

**MrmFetchColorLiteral** - Fetches a named color literal from a UID file

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmFetchColorLiteral**(
MrmHierarchy *hierarchy_id*,
String *index*,
Display **display*,
Colormap *colormap_id*,
Pixel **pixel*);

## DESCRIPTION

The **MrmFetchColorLiteral** function fetches a named color literal from a UID file, and converts the color literal to a pixel color value.

*hierarchy_id*

Specifies the ID of the UID hierarchy that contains the specified literal. The value of *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

*index*

Specifies the UIL name of the color literal to fetch. You must define this name in UIL as an exported value.

*display*

Specifies the display used for the pixmap. The *display* argument specifies the connection to the X server. For more information on the **Display** structure, see the Xlib function **XOpenDisplay**.

*colormap_id*

Specifies the ID of the color map. If *colormap_id* is NULL, the default color map is used.

*pixel*

Returns the ID of the color literal.

## RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmNOT_FOUND**

The color literal was not found in the UIL file.

**MrmWRONG_TYPE**

The caller tried to fetch a literal of a type not supported by this function.

**MrmFAILURE**

The function failed.

# RELATED

**MrmFetchBitmapLiteral(3)** , **MrmOpenHierarchyPerDisplay**(3) , **MrmFetchIconLiteral**(3) , **MrmFetchLiteral**(3) , and **XOpenDisplay**(3).

# MrmFetchIconLiteral (library call)

## NAME

**MrmFetchIconLiteral** – Fetches an icon literal from a hierarchy

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmFetchIconLiteral**(
MrmHierarchy *hierarchy_id*,
String *index*,
Screen **screen*,
Display **display*,
Pixel *fgpix*,
Pixel *bgpix*,
Pixmap **pixmap*);

## DESCRIPTION

The **MrmFetchIconLiteral** function fetches an icon literal from an MRM hierarchy and converts the icon literal to an X pixmap.

*hierarchy_id*

Specifies the ID of the UID hierarchy that contains the specified icon literal. The *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

*index*

Specifies the UIL name of the icon literal to fetch.

*screen*

Specifies the screen used for the pixmap. The *screen* argument specifies a pointer to the Xlib structure **Screen**, which contains the information about that screen and is linked to the **Display** structure. For more information on the **Display** and **Screen** structures, see the Xlib function **XOpenDisplay** and the associated screen information macros.

*display*

Specifies the display used for the pixmap. The *display* argument specifies the connection to the X server. For more information on the **Display** structure, see the Xlib function **XOpenDisplay**.

*fgpix*

Specifies the foreground color for the pixmap.

*bgpix*

Specifies the background color for the pixmap.

*pixmap*

Returns the resulting X pixmap value. The function allocates space for this pixmap. The application is responsible for managing the allocated space. The application can recover the allocated space by calling **XmDestroyPixmap**.

## RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmNOT_FOUND**

The icon literal was not found in the hierarchy.

**MrmWRONG_TYPE**

The caller tried to fetch a literal of a type not supported by this function.

**MrmFAILURE**

The function failed.

# RELATED

**MrmFetchBitmapLiteral(3)** , **MrmOpenHierarchyPerDisplay**(3) , **MrmFetchLiteral**(3) , **MrmFetchColorLiteral**(3) , and **XOpenDisplay**(3).

# MrmFetchLiteral (library call)

# NAME

**MrmFetchLiteral** –Fetches a literal from a UID file

# SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmFetchLiteral**(
MrmHierarchy *hierarchy_id*,
String *index*,
Display **display*,
XtPointer **value*,
MrmCode **type*);

# DESCRIPTION

The **MrmFetchLiteral** function reads and returns the value and type of a literal (named value) that is stored as a public resource in a single UID file. This function returns a pointer to the value of the literal. For example, an integer is always returned as a pointer to an integer, and a string is always returned as a pointer to a string.

Applications should not use **MrmFetchLiteral** for fetching icon or color literals. If this is attempted, **MrmFetchLiteral** returns an error.

*hierarchy_id*

Specifies the ID of the UID hierarchy that contains the specified literal. The value of *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

*index*

Specifies the UIL name of the literal (pixmap) to fetch. You must define this name in UIL as an exported value.

*display*

Specifies the display used for the pixmap. The *display* argument specifies the connection to the X server. For more information on the **Display** structure, see the Xlib function **XOpenDisplay**.

*value*

Returns the ID of the named literal's value. The function allocates space for the returned value. The application is responsible for managing the allocated space by calling the appropriate deallocation function. For example, if the returned ID symbolizes a pixmap, then the application can recover the allocated space by calling **XmDestroyPixmap**.

*type*

Returns the named literal's data type. Types are defined in the include file `Mrm/MrmPublic.h`.

# RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmNOT_FOUND**

The literal was not found in the UIL file.

**MrmWRONG_TYPE**

The caller tried to fetch a literal of a type not supported by this function.

**MrmFAILURE**

The function failed.

# RELATED

**MrmFetchBitmapLiteral(3)** , **MrmOpenHierarchyPerDisplay**(3) , **MrmFetchIconLiteral**(3) , **MrmFetchColorLiteral**(3) , and **XOpenDisplay**(3).

# MrmFetchSetValues (library call)

## NAME

**MrmFetchSetValues** – Fetches the values to be set from literals stored in UID files

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmFetchSetValues**(
MrmHierarchy *hierarchy_id*,
Widget *widget*,
ArgList *args*,
Cardinal *num_args*);

## DESCRIPTION

The **MrmFetchSetValues** function is similar to **XtSetValues**, except that the values to be set are defined by the UIL named values that are stored in the UID hierarchy. **MrmFetchSetValues** fetches the values to be set from literals stored in UID files.

*hierarchy_id*

Specifies the ID of the UID hierarchy that contains the specified literal. The value of *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

*widget*

Specifies the widget that is modified.

*args*

Specifies an argument list that identifies the widget arguments to be modified as well as the index (UIL name) of the literal that defines the value for that argument. The name part of each argument (*args[n].name*) must begin with the string **XmN** followed by the name that uniquely identifies this attribute tag. For example, **XmNwidth** is the attribute name associated with the core argument *width*. The value part (*args[n].value*) must be a string that gives the index (UIL name) of the literal. You must define all literals in UIL as exported values.

*num_args*

Specifies the number of entries in *args*.

This function sets the values on a widget, evaluating the values as public literal resource references resolvable from a UID hierarchy. Each literal is fetched from the hierarchy, and its value is modified and converted as required. This value is then placed in the argument list and used as the actual value for an **XtSetValues** call. **MrmFetchSetValues** allows a widget to be modified after creation using UID file values the same way creation values are used in **MrmFetchWidget**.

As in **MrmFetchWidget**, each argument whose value can be evaluated from the UID hierarchy is set in the widget. Values that are not found or values in which conversion errors occur are not modified.

Each entry in the argument list identifies an argument to be modified in the widget. The name part identifies the tag, which begins with **XmN**. The value part must be a string whose value is the index of the literal. Thus, the following code would modify the label resource of the widget to have the value of the literal accessed by the index *OK_button_label* in the hierarchy:

```
args[n].name = XmNlabel;
args[n].value = "OK_button_label";
```

## RETURN VALUE

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmPARTIAL_SUCCESS**

At least one literal was successfully fetched.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmFAILURE**

The function failed.

# RELATED

**MrmOpenHierarchyPerDisplay(3)** , **XtSetValues**(3).

# MrmFetchWidget (library call)

# NAME

**MrmFetchWidget** – Fetches and creates an indexed (UIL named) application widget and its children

# SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmFetchWidget**(
MrmHierarchy *hierarchy_id*,
String *index*,
Widget *parent_widget*,
Widget **widget*,
MrmType **class*);

# DESCRIPTION

The **MrmFetchWidget** function fetches and creates an indexed application widget and its children. The indexed application widget is any widget that is named in UIL. In fetch operations, the fetched widget's subtree is also fetched and created. This widget must not appear as the child of a widget within its own subtree. **MrmFetchWidget** does not execute **XtManageChild** for the newly created widget.

All widgets fetched by a call to **MrmFetchWidget** are not managed at the time of their creation callbacks.

*hierarchy_id*

Specifies the ID of the **UID** hierarchy that contains the interface definition. The value of *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

*index*

Specifies the UIL name of the widget to fetch.

*parent_widget*

Specifies the parent widget ID.

*widget*

Returns the widget ID of the created widget.

*class*

This argument must be set to an actual pointer; it cannot be a NULL pointer. **MrmFetchWidget** sets this argument to an implementation dependent value.

An application can fetch any named widget in the **UID** hierarchy using **MrmFetchWidget**. **MrmFetchWidget** can be called at any time to fetch a widget that was not fetched at application startup. **MrmFetchWidget** can be used to defer fetching pop–up widgets until they are first referenced (presumably in a callback), and then used to fetch them once.

**MrmFetchWidget** can also create multiple instances of a widget (and its subtree). In this case, the **UID** definition functions as a template; a widget definition can be fetched any number of times. An application can use this template to make multiple instances of a widget, for example, in a dialog box box or menu.

The index (UIL name) that identifies the widget must be known to the application.

# RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmNOT_FOUND**

The widget was not found in UID hierarchy.

**MrmFAILURE**

The function failed.

# RELATED

**MrmOpenHierarchyPerDisplay(3)** , **MrmFetchWidgetOverride**(3) .

# MrmFetchWidgetOverride (library call)

## NAME

**MrmFetchWidgetOverride** – Fetches any indexed (UIL named) application widget. It overrides the arguments specified for this application widget in UIL

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmFetchWidgetOverride**(
MrmHierarchy *hierarchy_id*,
String *index*,
Widget *parent_widget*,
String *override_name*,
ArgList *override_args*,
Cardinal *override_num_args*,
Widget **widget*,
MrmType **class*);

## DESCRIPTION

The **MrmFetchWidgetOverride** function is the extended version of **MrmFetchWidget**. It is identical to **MrmFetchWidget**, except that it allows the caller to override the widget's name and any arguments that **MrmFetchWidget** would otherwise retrieve from the UID file or one of the defaulting mechanisms. That is, the override argument list is not limited to those arguments in the UID file.

The override arguments apply only to the widget fetched and returned by this function. Its children (subtree) do not receive any override parameters.

*hierarchy_id*

Specifies the ID of the UID hierarchy that contains the interface definition. The value of *hierarchy_id* was returned in a previous call to **MrmOpenHierarchyPerDisplay**.

*index*

Specifies the UIL name of the widget to fetch.

*parent_widget*

Specifies the parent widget ID.

*override_name*

Specifies the name to override the widget name. Use a NULL value if you do not want to override the widget name.

*override_args*

Specifies the override argument list, exactly as given to **XtCreateWidget** (conversion complete and so forth). Use a NULL value if you do not want to override the argument list.

*override_num_args*

Specifies the number of arguments in *override_args*.

*widget*

Returns the widget ID of the created widget.

*class*

Returns the class code identifying MRM's widget class. Literals identifying MRM widget class codes are defined in the include file `Mrm/MrmPublic.h`.

# RETURN VALUE

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmBAD_HIERARCHY**

The hierarchy ID was invalid.

**MrmNOT_FOUND**

The widget was not found in UID hierarchy.

**MrmFAILURE**

The function failed.

# RELATED

**MrmOpenHierarchyPerDisplay(3)** , **MrmFetchWidget**(3) .

# MrmInitialize (library call)

## NAME

**MrmInitialize** –Prepares an application to use MRM widget–fetching facilities

## SYNOPSIS

void **MrmInitialize**(void);

## DESCRIPTION

The **MrmInitialize** function must be called to prepare an application to use MRM widget–fetching facilities. You must call this function prior to fetching a widget. However, it is good programming practice to call **MrmInitialize** prior to performing any MRM operations.

**MrmInitialize** initializes the internal data structures that MRM needs to successfully perform type conversion on arguments and to successfully access widget creation facilities. An application must call **MrmInitialize** before it uses other MRM functions.

# MrmOpenHierarchy (library call)

# NAME

**MrmOpenHierarchy** –Allocates a hierarchy ID and opens all the UID files in the hierarchy

# SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmOpenHierarchy**(
MrmCount *num_files*,
String *file_names_list*[],
MrmOsOpenParamPtr **ancillary_structures_list*,
MrmHierarchy **hierarchy_id*);

# DESCRIPTION

This routine is obsolete and exists for compatibility with previous releases. It is replaced by **MrmOpenHierarchyPerDisplay**.
**MrmOpenHierarchy** is identical to **MrmOpenHierarchyPerDisplay** except that **MrmOpenHierarchy** does not take a *display* argument.

*num_files*

Specifies the number of files in the name list.

*file_names_list*

Specifies an array of character strings that identify the UID files.

*ancillary_structures_list*

A list of operating−system−dependent ancillary structures corresponding to items such as filenames, clobber flags, and so forth. This argument should be NULL for most operations. If you need to reference this structure, see the definition of **MrmOsOpenParamPtr** in the `MrmPublic.h` header file for more information.

*hierarchy_id*

Returns the search hierarchy ID. The search hierarchy ID identifies the list of UID files that MRM searches (in order) when performing subsequent fetch calls.

Each UID file string in *file_names_list* can specify either a full pathname or a filename. If a UID file string has a leading slash (*/*), it specifies a full pathname, and MRM opens the file as specified. Otherwise, the UID file string specifies a filename. In this case, MRM looks for the file along a search path specified by the **UIDPATH** environment variable or by a default search path, which varies depending on whether or not the **XAPPLRESDIR** environment variable is set.

The **UIDPATH** environment variable specifies a search path and naming conventions associated with UID files. It can contain the substitution field *%U*, where the UID file string from the *file_names_list* argument to **MrmOpenHierarchyPerDisplay** is substituted for *%U*. It can also contain the substitution fields accepted by **XtResolvePathname**. The substitution field *%T* is always mapped to *uid*. The entire path is first searched with *%S* mapped to *.uid*. If no file is found, it is searched again with *%S* mapped to NULL.

If no display is set prior to calling this function, the result of this function's call to **XtResolvePathname** is undefined.

For example, the following **UIDPATH** value and **MrmOpenHierarchy** call cause MRM to open two separate UID files:

```
UIDPATH=/uidlib/%L/%U.uid:/uidlib/%U/%L
  static char *uid_files[] = {"/usr/users/me/test.uid", "test2"};
  MrmHierarchy  *Hierarchy_id;
  MrmOpenHierarchy((MrmCount)2,uid_files, NULL, Hierarchy_id)
```

MRM opens the first file, `/usr/users/me/test.uid`, as specified in the *file_names_list* argument to **MrmOpenHierarchy**, because the

UID file string in the *file_names_list* argument specifies a full pathname. MRM looks for the second file, `test2`, first as `/uidlib/%L/test2.uid` and second as `/uidlib/test2/%L`, where the display's language string is substituted for *%L*.

After **MrmOpenHierarchy** opens the UID hierarchy, you should not delete or modify the UID files until you close the UID hierarchy by calling **MrmCloseHierarchy**.

If **UIDPATH** is not set but the environment variable **XAPPLRESDIR** is set, MRM searches the following pathnames:

- `%U%S`
- `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/%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 neither **UIDPATH** nor **XAPPLRESDIR** is set, MRM searches the following pathnames:

- `%U%S`
- `HOME/%L/uid/%N/%U%S`
- `HOME/%l/uid/%N/%U%S`
- `HOME/uid/%N/%U%S`
- `HOME/%L/uid/%U%S`
- `HOME/%l/uid/%U%S`
- `HOME/uid/%U%S`
- `HOME/%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`

These paths are defaults that vendors may change. For example, a vendor may use different directories for `/usr/lib/X11` and `/usr/include/X11`.

The following substitutions are used in these paths:

*%U*

The UID file string, from the *file_names_list* argument.

*%N*

The class name of the application.

*%L*

The display's language string. This string is influenced by **XtSetLanguageProc**. The default string is determined by calling

**setlocale(LC_ALL, NULL)**.

*%l*

The language component of the display's language string.

*%S*

The suffix to the filename. The entire path is first searched with a suffix of `.uid`. If no file is found, it is searched again with a NULL suffix.

# RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmNOT_FOUND**

File not found.

**MrmFAILURE**

The function failed.

# RELATED

**MrmOpenHierarchyPerDisplay(3)** and **MrmCloseHierarchy**(3) .

# MrmOpenHierarchyFromBuffer (library call)

## NAME

**MrmOpenHierarchyFromBuffer** – Allocates a hierarchy ID and opens a buffer containing a memory image of a UID file

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmOpenHierarchyFromBuffer**(
unsigned char *uid_buffer*,
MrmHierarchy **hierarchy_id*);

## DESCRIPTION

**MrmOpenHierarchyFromBuffer** allows you to specify a buffer containing information from UID files that MRM searches in subsequent fetch operations. This function also allocates a hierarchy ID and initializes the optimized search lists in the hierarchy.

*buffer*

Specifies a stream of bytes containing information from UID files

*hierarchy_id*

Returns the search hierarchy ID. The search hierarchy ID identifies the buffer that MRM searches when performing subsequent fetch calls.

## RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmFAILURE**

The function failed.

## RELATED

**MrmCloseHierarchy(3)** and **MrmOpenHierarchyPerDisplay**(3) .

# MrmOpenHierarchyPerDisplay (library call)

## NAME

**MrmOpenHierarchyPerDisplay** – Allocates a hierarchy ID and opens all the UID files in the hierarchy

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmOpenHierarchyPerDisplay**(
Display **display*,
MrmCount *num_files*,
String *file_names_list*[],
MrmOsOpenParamPtr **ancillary_structures_list*,
MrmHierarchy **hierarchy_id*);

## DESCRIPTION

**MrmOpenHierarchyPerDisplay** allows you to specify the list of UID files that MRM searches in subsequent fetch operations. All subsequent fetch operations return the first occurrence of the named item encountered while traversing the UID hierarchy from the first list element (UID file specification) to the last list element. This function also allocates a hierarchy ID and opens all the UID files in the hierarchy. It initializes the optimized search lists in the hierarchy. If **MrmOpenHierarchyPerDisplay** encounters any errors during its execution, any files that were opened are closed.

The application must call **XtAppInitialize** before calling **MrmOpenHierarchyPerDisplay**.

*display*

Specifies the connection to the X server and the value to pass to **XtResolvePathname**. For more information on the **Display** structure, see the Xlib function **XOpenDisplay**.

*num_files*

Specifies the number of files in the name list.

*file_names_list*

Specifies an array of character strings that identify the UID files.

*ancillary_structures_list*

A list of operating−system−dependent ancillary structures corresponding to items such as filenames, clobber flags, and so forth. This argument should be NULL for most operations. If you need to reference this structure, see the definition of **MrmOsOpenParamPtr** in the `MrmPublic.h` header file for more information.

*hierarchy_id*

Returns the search hierarchy ID. The search hierarchy ID identifies the list of UID files that MRM searches (in order) when performing subsequent fetch calls.

Each UID file string in *file_names_list* can specify either a full pathname or a filename. If a UID file string has a leading / (slash), it specifies a full pathname, and MRM opens the file as specified. Otherwise, the UID file string specifies a filename. In this case MRM looks for the file along a search path specified by the **UIDPATH** environment variable or by a default search path, which varies depending on whether or not the **XAPPLRESDIR** environment variable is set.

The **UIDPATH** environment variable specifies a search path and naming conventions associated with UID files. It can contain the substitution field *%U*, where the UID file string from the *file_names_list* argument to **MrmOpenHierarchyPerDisplay** is substituted for *%U*. It can also contain the substitution fields accepted by **XtResolvePathname**. The substitution field *%T* is always mapped to *uid*. The entire path is searched

first with *%S* mapped to `.uid`. If no file is found, it is searched again with *%S* mapped to NULL. For example, the following **UIDPATH** value and **MrmOpenHierarchyPerDisplay** call cause MRM to open two separate UID files:

```
UIDPATH=/uidlib/%L/%U.uid:/uidlib/%U/%L
  static char *uid_files[] = {"/usr/users/me/test.uid", "test2"};
  MrmHierarchy  *Hierarchy_id;
  MrmOpenHierarchyPerDisplay((MrmCount)2,uid_files, NULL, Hierarchy_id)
```

MRM opens the first file, `/usr/users/me/test.uid`, as specified in the *file_names_list* argument to **MrmOpenHierarchyPerDisplay**, because the UID file string in the *file_names_list* argument specifies a full pathname. MRM looks for the second file, `test2`, first as `/uidlib/%L/test2.uid` and second as `/uidlib/test2/%L`, where the display's language string is substituted for *%L*.

After **MrmOpenHierarchyPerDisplay** opens the UID hierarchy, you should not delete or modify the UID files until you close the UID hierarchy by calling **MrmCloseHierarchy**.

If **UIDPATH** is not set, but the environment variable **XAPPLRESDIR** is set, MRM searches the following pathnames:

- `%U%S`
- `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/%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 neither **UIDPATH** nor **XAPPLRESDIR** is set, MRM searches the following pathnames:

- `%U%S`
- `HOME/%L/uid/%N/%U%S`
- `HOME/%l/uid/%N/%U%S`
- `HOME/uid/%N/%U%S`
- `HOME/%L/uid/%U%S`
- `HOME/%l/uid/%U%S`
- `HOME/uid/%U%S`
- `HOME/%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`

These paths are defaults that vendors may change. For example, a vendor may use different directories for `/usr/lib/X11` and `/usr/include/X11`.

The following substitutions are used in these paths:

*%U*

The UID file string, from the *file_names_list* argument.

*%N*

The class name of the application.

*%L*

The display's language string. This string is influenced by **XtSetLanguageProc**. The default string is determined by calling **setlocale(LC_ALL, NULL)**.

*%l*

The language component of the display's language string.

*%S*

The suffix to the filename. The entire path is first searched with a suffix of `.uid`. If no file is found, it is searched again with a NULL suffix.

# RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmNOT_FOUND**

File not found.

**MrmFAILURE**

The function failed.

# RELATED

**MrmCloseHierarchy(3)** .

# MrmRegisterClass (library call)

## NAME

**MrmRegisterClass** − Saves the information needed for MRM to access the widget creation function for user−defined widgets

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmRegisterClass**(
MrmType *class_code*,
String *class_name*,
String *create_name*,
Widget (**create_proc*) (),
WidgetClass *class_record*);

## DESCRIPTION

The **MrmRegisterClass** function allows MRM to access user−defined widget classes. This function registers the necessary information for MRM to create widgets of this class. You must call **MrmRegisterClass** prior to fetching any user−defined class widget.

**MrmRegisterClass** saves the information needed to access the widget creation function and to do type conversion of argument lists by using the information in MRM databases.

*class_code*

This argument is ignored; it is present for compatibility with previous releases.

*class_name*

This argument is ignored; it is present for compatibility with previous releases.

*create_name*

Specifies the case−sensitive name of the low−level widget creation function for the class. An example from the Motif Toolkit is **XmCreateLabel**. Arguments are *parent_widget*, *name*, *override_arglist*, and *override_argcount*.

For user−defined widgets, *create_name* is the creation procedure in the UIL that defines this widget.

*create_proc*

Specifies the address of the creation function that you named in *create_name*.

*class_record*

Specifies a pointer to the class record.

## RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmFAILURE**

The function failed.

# MrmRegisterNames (library call)

# NAME

**MrmRegisterNames** –Registers the values associated with the names referenced in UIL (for example, UIL callback function names or UIL identifier names)

# SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmRegisterNames**(
MrmRegisterArglist *register_list*,
MrmCount *register_count*);

# DESCRIPTION

The **MrmRegisterNames** function registers a vector of names and associated values for access in MRM. The values can be callback functions, pointers to user−defined data, or any other values. The information provided is used to resolve symbolic references occurring in UID files to their run−time values. For callbacks, this information provides the procedure address required by the Motif Toolkit. For names used as identifiers in UIL, this information provides any run−time mapping the application needs.

This function is similar to **MrmRegisterNamesInHierarchy**, except that the scope of the names registered by **MrmRegisterNamesInHierarchy** is limited to the hierarchy specified in the call to that function, whereas the names registered by **MrmRegisterNames** have global scope. When MRM looks up a name, it first tries to find the name among those registered for the given hierarchy. If that lookup fails, it tries to find the name among those registered globally.

*register_list*

Specifies a list of name/value pairs for the names to be registered. Each name is a case−sensitive, NULL−terminated ASCII string. Each value is a 32−bit quantity, interpreted as a procedure address if the name is a callback function, and uninterpreted otherwise.

*register_count*

Specifies the number of entries in *register_list*.

The names in the list are case−sensitive. The list can be either ordered or unordered.

Callback functions registered through **MrmRegisterNames** can be either regular or creation callbacks. Regular callbacks have declarations determined by Motif Toolkit and user requirements. Creation callbacks have the same format as any other callback:

void **CallBackProc**(
Widget **widget_id*,
Opaque *tag*,
XmAnyCallbackStruct **callback_data*);

*widget_id*

Specifies the widget ID associated with the widget performing the callback (as in any callback function).

*tag*

Specifies the tag value (as in any callback function).

*callback_data*

Specifies a widget−specific data structure. This data structure has a minimum of two members: event and reason. The reason member is always set to **MrmCR_CREATE**.

Note that the widget name and parent are available from the widget record accessible through *widget_id*.

# RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmFAILURE**

The function failed.

# MrmRegisterNamesInHierarchy (library call)

## NAME

**MrmRegisterNamesInHierarchy** -Registers the values associated with the names referenced in UIL within a single hierarchy (for example, UIL callback function names or UIL identifier names)

## SYNOPSIS

#include <Mrm/MrmPublic.h>

Cardinal **MrmRegisterNamesInHierarchy**(
MrmHierarchy *hierarchy_id*,
MrmRegisterArglist *register_list*,
MrmCount *register_count*);

## DESCRIPTION

The **MrmRegisterNamesInHierarchy** function registers a vector of names and associated values for access in MRM. The values can be callback functions, pointers to user−defined data, or any other values. The information provided is used to resolve symbolic references occurring in UID files to their run−time values. For callbacks, this information provides the procedure address required by the Motif Toolkit. For names used as identifiers in UIL, this information provides any run−time mapping the application needs.

This function is similar to **MrmRegisterNames**, except that the scope of the names registered by **MrmRegisterNamesInHierarchy** is limited to the hierarchy specified by *hierarchy_id*, whereas the names registered by **MrmRegisterNames** have global scope. When MRM looks up a name, it first tries to find the name among those registered for the given hierarchy. If that lookup fails, it tries to find the name among those registered globally.

*hierarchy_id*

Specifies the hierarchy with which the names are to be associated.

*register_list*

Specifies a list of name/value pairs for the names to be registered. Each name is a case−sensitive, NULL−terminated ASCII string. Each value is a 32−bit quantity, interpreted as a procedure address if the name is a callback function, and uninterpreted otherwise.

*register_count*

Specifies the number of entries in *register_list*.

The names in the list are case−sensitive. The list can be either ordered or unordered.

Callback functions registered through **MrmRegisterNamesInHierarchy** can be either regular or creation callbacks. Regular callbacks have declarations determined by Motif Toolkit and user requirements. Creation callbacks have the same format as any other callback:

void **CallBackProc**(
Widget **widget_id*,
Opaque *tag*,
XmAnyCallbackStruct **callback_data*);

*widget_id*

Specifies the widget ID associated with the widget performing the callback (as in any callback function).

*tag*

Specifies the tag value (as in any callback function).

*callback_data*

Specifies a widget−specific data structure. This data structure has a minimum of two members: event and reason. The reason member is always set to **MrmCR_CREATE**.

Note that the widget name and parent are available from the widget record accessible through *widget_id.*

# RETURN

This function returns one of the following status return constants:

**MrmSUCCESS**

The function executed successfully.

**MrmFAILURE**

The function failed.