

# MLPluginIfc

1.0

Generated by Doxygen 1.13.2



---

<b>1 MLPluginIfc</b>	<b>1</b>
<b>2 Class Index</b>	<b>3</b>
2.1 Class List . . . . .	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 MLPlugin Class Reference . . . . .	5
3.1.1 Detailed Description . . . . .	6
3.1.2 Constructor & Destructor Documentation . . . . .	7
3.1.2.1 MLPlugin() . . . . .	7
3.1.3 Member Function Documentation . . . . .	7
3.1.3.1 createWindow() . . . . .	7
3.1.3.2 getPluginDescription() . . . . .	7
3.1.3.3 getPluginName() . . . . .	7
3.1.4 Member Data Documentation . . . . .	7
3.1.4.1 af . . . . .	7
<b>Index</b>	<b>9</b>



# Chapter 1

## MLPluginIfc

**MLPluginIfc** library provides interfaces to create plugins for [MyLibrary](#).

To start add cmake package MLPluginIfc to your project.

```
find_package(MLPluginIfc)
if (MLPluginIfc_FOUND)
    target_include_directories(myproject
        PRIVATE MLPluginIfc::mlpluginifc
        PRIVATE MLBookProc::mlbookproc
    )
    target_link_libraries(myproject
        PRIVATE MLPluginIfc::mlpluginifc
        PRIVATE MLBookProc::mlbookproc
    )
endif()
```

See [MLPlugin](#) for details and code example.



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">MLPlugin</a>	
The <a href="#">MLPlugin</a> class . . . . .	<a href="#">5</a>





## Chapter 3

# Class Documentation

### 3.1 MLPlugin Class Reference

The [MLPlugin](#) class.

```
#include <MLPlugin.h>
```

#### Public Member Functions

- [MLPlugin](#) (void \*af\_ptr)  
*MLPlugin constructor.*
- virtual void [createWindow](#) (Gtk::Window \*parent\_window)  
*Virtual function.*
- Glib::ustring [getPluginName](#) ()  
*Returns plugin name if set.*
- Glib::ustring [getPluginDescription](#) ()  
*Returns plugin description.*

#### Protected Attributes

- std::shared\_ptr< AuxFunc > [af](#)  
*Pointer to AuxFunc object.*
- Glib::ustring **plugin\_name**  
*Plugin name.*
- Glib::ustring **plugin\_description**  
*Plugin description.*

### 3.1.1 Detailed Description

The [MLPlugin](#) class.

[MLPlugin](#) is base class for plugins creation. To create plugin inherit your plugin base class from [MLPlugin](#) and override [createWindow\(\)](#) method. Also set [plugin\\_name](#) and [plugin\\_description](#) if needed. Your plugin base header file must include C function create (see example).

```
#ifndef EXAMPLEPLUGIN_H
#define EXAMPLEPLUGIN_H

#include <MLPlugin.h>

class ExamplePlugin : public MLPlugin
{
public:
    ExamplePlugin(void *af_ptr);

    void
    createWindow(Gtk::Window *parent_window) override;
};

extern "C"
{
#ifdef __linux
    MLPlugin *
    create(void *af_ptr)
    {
        return new ExamplePlugin(af_ptr);
    }
#endif
#ifdef _WIN32
    __declspec(dllexport) MLPlugin *
    create(void *af_ptr)
    {
        return new ExamplePlugin(af_ptr);
    }
#endif
}
#endif // EXAMPLEPLUGIN_H

#include <ExamplePlugin.h>
#include <gtkmm-4.0/gtkmm/grid.h>
#include <gtkmm-4.0/gtkmm/label.h>

ExamplePlugin::ExamplePlugin(void *af_ptr) : MLPlugin(af_ptr)
{
    plugin_name = "Example plugin";
    plugin_description = "Small example plugin";
}

void
ExamplePlugin::createWindow(Gtk::Window *parent_window)
{
    Gtk::Window *window = new Gtk::Window();
    window->set_application(parent_window->get_application());
    window->set_title(plugin_name);
    window->set_name("MLwindow");
    window->set_transient_for(*parent_window);
    window->set_modal(true);

    Gtk::Grid *grid = Gtk::make_managed<Gtk::Grid>();
    grid->set_halign(Gtk::Align::FILL);
    grid->set_valign(Gtk::Align::FILL);
    window->set_child(*grid);

    Gtk::Label *lab = Gtk::make_managed<Gtk::Label>();
    lab->set_margin(5);
    lab->set_halign(Gtk::Align::START);
    lab->set_name("windowLabel");
    lab->set_text("Example plugin");
    grid->attach(*lab, 0, 0, 1, 1);

    window->signal_close_request().connect (
        [window] {
            std::unique_ptr<Gtk::Window> win(window);
            win->set_visible(false);
            return true;
        },
        false);

    window->present();
}
```

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 MLPlugin()

```
MLPlugin::MLPlugin (
    void * af_ptr)
```

[MLPlugin](#) constructor.

##### Parameters

<i>af_ptr</i>	pointer to std::shared_ptr<AuxFunc> object.
---------------	---

### 3.1.3 Member Function Documentation

#### 3.1.3.1 createWindow()

```
virtual void MLPlugin::createWindow (
    Gtk::Window * parent_window) [virtual]
```

Virtual function.

You should override this method to create your plugin window. New window should be transient for parent\_window. It is also strongly recommended to make new window modal (see example in class description).

##### Parameters

<i>parent_window</i>	pointer to parent window object
----------------------	---------------------------------

#### 3.1.3.2 getPluginDescription()

```
Glib::ustring MLPlugin::getPluginDescription ()
```

Returns plugin description.

##### Returns

[Glib](#) string containing plugin name

#### 3.1.3.3 getPluginName()

```
Glib::ustring MLPlugin::getPluginName ()
```

Returns plugin name if set.

##### Returns

[Glib](#) string containing plugin name

### 3.1.4 Member Data Documentation

#### 3.1.4.1 af

```
std::shared_ptr<AuxFunc> MLPlugin::af [protected]
```

Pointer to AuxFunc object.

AuxFunc class contains various useful methods (see **MLBookProc** documentation).



# Index

af

MLPlugin, [7](#)

createWindow

MLPlugin, [7](#)

getPluginDescription

MLPlugin, [7](#)

getPluginName

MLPlugin, [7](#)

MLPlugin, [5](#)

af, [7](#)

createWindow, [7](#)

getPluginDescription, [7](#)

getPluginName, [7](#)

MLPlugin, [7](#)

MLPluginIfc, [1](#)