# Analysing Plugin Code

From Notepad++ Wiki

Jump to: [navigation](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#mw-navigation), [search](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#p-search)

Understanding the source of an actual plugin

People of very different backgrounds may want to build a plugin to suit their own purposs, and perhaps share it later. Some are experienced C or C++ coders, for whom [Plugin Development](http://npp-wiki.tuxfamily.org/index.php?title=Plugin_Development) is simply a technical page with (nearly?) everything they need. For other segments of the audience, describing interfacces isn't as easy to grasp, and the example of an actual, useful plugin may be more helpful. This page will attempt to show how XBrackets Lite source was designed by dv\_\_.

While a few basics on Windows programming will be explained, this is not a Windows C++ programming tutorial. Nor a C++ tutorial. The Links page on the Notepad++ website features a link to the C++ In Action tutorial, which may be a useful reference in the sequel.

The archive of XBracketsLite source files unpacks into 15 header files, 9 code files, 1 resource file for the option dialog box and 3 other files we won't be concerned with. Any version of Visual Studio C++ Express will compile those, perhaps with more warnings using VS2008, as unsafe string functions are being reported. A short and last word on the remaining 3 files:

* readme.txt is meant for the end user only;
* XBrackets\_VC8.vcproj is the Visual Studio option file for the project. While inspecting it may help by disclosing various compiler options, it is generated by the IDE and should be edited in its framework only;
* XBrackets\_VC8.sln is the solution file. Again, this is generated by the IDE, should not be edited manually and holds very little usable information.

## Contents

 [[hide](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code)]

* [1 Brief reminder on the Windows programming model](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#Brief_reminder_on_the_Windows_programming_model)
* [2 Header files overview](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#Header_files_overview)
* [3 Header files in detail](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#Header_files_in_detail)
  + [3.1 base.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#base.h)
  + [3.2 NppMessager.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NppMessager.h)
  + [3.3 NppPlugin.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NppPlugin.h)
  + [3.4 NppPluginMenu.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NppPluginMenu.h)
  + [3.5 NppPluginMsgSender.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NppPluginMsgSender.h)
  + [3.6 SciMessager.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#SciMessager.h)
  + [3.7 resource.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#resource.h)
  + [3.8 SettingsDlg.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#SettingsDlg.h)
  + [3.9 XBracketsOptions.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#XBracketsOptions.h)
  + [3.10 XBracketsMenu.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#XBracketsMenu.h)
  + [3.11 XBrackets.h](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#XBrackets.h)
* [4 The code proper](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#The_code_proper)
  + [4.1 NppMessager.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NppMessager.cpp)
  + [4.2 SciMessager.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#SciMessager.cpp)
  + [4.3 NppPluginMenu.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NppPluginMenu.cpp)
  + [4.4 SetingDlg.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#SetingDlg.cpp)
  + [4.5 NpppPluginMenu.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NpppPluginMenu.cpp)
  + [4.6 XBracketsOptions.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#XBracketsOptions.cpp)
  + [4.7 Main.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#Main.cpp)
  + [4.8 NppPlugin.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#NppPlugin.cpp)
  + [4.9 XBrackets.cpp](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#XBrackets.cpp)
    - [4.9.1 nppCallWndProc()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#nppCallWndProc.28.29)
    - [4.9.2 nppNewWndProc()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#nppNewWndProc.28.29)
    - [4.9.3 CXBrackets(), ~CXBrackets()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#CXBrackets.28.29.2C_.7ECXBrackets.28.29)
    - [4.9.4 npppGetFuncsArray(), nppGetName()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#npppGetFuncsArray.28.29.2C_nppGetName.28.29)
    - [4.9.5 beNotified()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#beNotified.28.29)
    - [4.9.6 OnNppSetInfo()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#OnNppSetInfo.28.29)
    - [4.9.7 OnNppBufferActivated(), OnNppFileOpened(), OnNppFileSaved()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#OnNppBufferActivated.28.29.2C_OnNppFileOpened.28.29.2C_OnNppFileSaved.28.29)
    - [4.9.8 OnNppReady()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#OnNppReady.28.29)
    - [4.9.9 OnNppShutdown()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#OnNppShutdown.28.29)
    - [4.9.10 OnNppMacro()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#OnNppMacro.28.29)
    - [4.9.11 OnSciCharAdded()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#OnSciCharAdded.28.29)
    - [4.9.12 ReadOptions()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#ReadOptions.28.29)
    - [4.9.13 SaveOptions(), UpdateFileType()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#SaveOptions.28.29.2C_UpdateFileType.28.29)
    - [4.9.14 getEscapedPrefixPos(), isEscapePrefix()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#getEscapedPrefixPos.28.29.2C_isEscapePrefix.28.29)
    - [4.9.15 AutoBracketsFunc()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#AutoBracketsFunc.28.29)
    - [4.9.16 getFileType()](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code#getFileType.28.29)

### Brief reminder on the Windows programming model

Under Windows, the basic objects are ***windows***, which may or may not paint stuff on the monitor. They exchange ***messages***, and receive messages from the operating system. They react to those messages they wish to process, polling their own ***message queue*** and handling messages in their ***message loop***.

A window which is to perform a very specific and standard group of related tasks is called a ***control***. Windows defines quite a few of them - scroll bars, buttons, dropdown lists and many more -. A special kind of window is being used to marshal a group of controls, and is called a ***dialog box***. Windows differ by their declared behaviour, which is summarized by their ***class***. The operating system will not send a message retrieving the thumb position to a button, because a scroll bar and a button belong to different classes, and buttons have no thumb. Hopefully, neither will your application do that - the effects could be interesting.

Everything under Windows has a ***handle***, an unique unsigned long identifier. They identify windows, ***resources*** and other more exotic objects. While programming, you deal with Windows objects almost exclusively using handles - we'll mention the most prominent exception later.

Finally, while messages basically can be sent by any window to any window, a large class of information is one way only. This is conveyed by special sorts of messages called ***notifications***. Typically, a ***child control*** will send a notification to its ***parent window*** so as to report that it was clicked, dragged, resized etc. For historical reasons, two messages convey notifications, WM\_COMMAND and WM\_NOTIFY. An application with a graphical interface cannot dispense with monitoring these messages.

### Header files overview

You will invariably wind up with between 3 and 5 categories of header files:

* System header files. The inclusion of Windows.h, for instance, is pretty much ubiquitous.
* Notepad++ header and Scintilla header files. Typically contain structures for plugins, Notepad++ messages and notifications, and the like.
* Custom header files, because you may plan to reuse part of your code later, and they solve forward reference issues.
* Optionally, third party header files, specially if you use the PluginInterface library.
* Optionally, structure header files to organize multiple custom or third party header files.

XBracketsLite has them all:

* System header files:
  + None is included in the sources, as expected, but base.h does include some.
* Notepad++ header and Scintilla header files:
  + menuCmdID.h: the list of all menu identifiers in C++. A likely presence.
  + Notepad\_plus\_msgs.h: is it really useful to explain what the contents are? You've got three guesses. Hint: it is very likely that your plugin will exchange messages with Notepad++, and listen to some of its notifications.
  + Scintilla.h: basic structure definitions related to the Scintilla control, a class around which Notepad++ is built.
* Custom header files. Usually, a custom header file is paired with a code file:
  + NppMessager.h
  + NppPlugin.h
  + NppPluginMenu.h
  + NppPluginMsgSender.h
  + SciMessager.h
  + resource.h
  + SettingsDlg.h
  + XBrackets.h
  + XBracketsMenu.h
  + XBracketsOptions.h
* Third party header files:
  + PluginInterface.h: a bundle of helper functions.
* Structure header files:
  + base.h

Note that, even though there are many files, most of them are very small. Th custom code, i.e. not borrowed to Notepad++ source or third party libraries, totals at most 2,000 lines of code - a small amount.

We will not describe in detail the borrowed code, only when it matters for proper understanding custom code.

### Header files in detail

All header files are guarded against multiple inclusion through this very common idiom:

#ifndef <filename>\_H

#define <filename>\_H

// actual contents

#endif

This way, the file can be symbolically included as many times as needed, to tell the compiler that the symbols it defines are in scope, but actually parsed only once. The compiler would complain about multiple definitions without this trick.

#### base.h

This simply includes the needed system header files. Since the program is very small, there is not much need to distinguish whether a file needs both Windows.h and TCHAR.h, only one or non of them. The former has all the basic Windows structure and message definitions. TCHAR is a macro that defines what a character is, since there are non Unicode older) and Unicode (newer) Windows builds. Also, since the plugin may have to deal with either ANSI or Unicode versions of Notepad++, this TCHAR macro is quite useful to avoid most of the otherwise required #ifdef's. The files are between angle brackets as they are in predefined, compiler dependent directories.

#### NppMessager.h

This defines a class the name of which suggests it serves to communicate with Notepad++. It is meant to be derived from, as the virtual destructor and protected members indicate. For the most part, it defines pure queries (they are constant). The NppData structure is defined in PluginInterface.h; at load time, such a structure is passed to the plugin, telling it about the handles of the main window and the two visible Scintilla controls in Notepad++.

#### NppPlugin.h

This defines the CNppPlugin class, which defines, among other things, the indispensable elements of a plugin interface, without which the plugin loader will abort loading the plugin. They are listed early in [Plugin Development](http://npp-wiki.tuxfamily.org/index.php?title=Plugin_Development). Again, it is meant to be derived from.

Another advantage of defining all public methods virtual is that it make their address available through the virtual pointer table, which helps when programming using the COM Windows component model. While not zero, the performance price to pay is usually unnoticeable on modern computers.

#### NppPluginMenu.h

This defines a class which is basically a CNppMessager with an extra method. It has been decided to aggregate rather than to derive. For such simple classes it hardly matters.

#### NppPluginMsgSender.h

This defines yet another simple class the obvious purpose of which is to communicate with other plugins. The Notepad++ message set indeed has the specially crafted NPPM\_MSGTOPLUGIN message. Checking the [Messages And Notifications](http://npp-wiki.tuxfamily.org/index.php?title=Messages_And_Notifications) page will make the class design fairly obvious.

#### SciMessager.h

This class handles all communication with Scintilla. It defines many methods, which boil down to sending Scintilla all the messages the plugin needs at some point.

#### resource.h

Contains control identifiers that are described in XBrackets.rc. .rc files are the simplest way to build dialog boxes, and the plugin uses one for its configuration. There will be a specific section devoted to the topic.

#### SettingsDlg.h

Defines the prototype for a dialog box hook callback.

#### XBracketsOptions.h

Defines the CXBracketsOptions class, which summarizes all options the configuration dialog is a graphical interface to.

#### XBracketsMenu.h

The CXBracketsMenu class defined therein derives from CNppPluginMenu. Look at the definition while clicking on the Plugins menu in Notepad++ and highlighting the XBrackets Lite entry. Mostly self-explanatory.

#### XBrackets.h

Last but not least, it defines the XBrackets class, deriving from CNppPlugin. The comments inside don't leave much to talk about.

Please note three important points:

1. The above seems to be overly complex for such a small program. But this is the price to pay for having reusable code - the classes with virtual features. At least in C++.
2. XBrackets Lite was born out of XBrackets, a plugin for a different editor. This may have influenced the design in possibly unnatural-looking ways.
3. Recall that you are inspecting the result of countless hours of coding and debugging. Your future plugin may eventually look like this, but it will not start like this, and will

undergo many rounds of remodeling and possibly unnatural additions - because of the way C++ does things, or the way Windows does things; there is a heavy legacy burden there -. A screencast on the actual genesis of a plugin would be nice - any taker?

### The code proper

It may look like header files are meant to hold declarations and class blueprints, while code files implement the features in the same named header files. Basically, this is right. However, you can stick code in class definitions, which header files are meant to hold for reuse purposes. Such code is by default inlined, which means method calls are not translated into routine calls at the assembly level. For performance- bound applications or very old computers, this is a valuable feature. Since the practice scatters code further than what plain logic would suggests, it lowers code readability. Everything has its own price. You have seen such implicitly inlined routines already.

#### NppMessager.cpp

Implements the various commands and queries targeting Scintilla. Apart perhaps from getCurrebtScintillaWnd(), they are completely straightforward. At this point, you may wish to read the [scintilla.iface](http://scintilla.cvs.sourceforge.net/viewvc/scintilla/scintilla/include/scintilla.iface?view=markup) file for a quick reference. It is concise without being terse, not a small achievement. You can of course consult the documentation page at [http://www.scintilla.org](http://www.scintilla.org/) , as well as [Messages And Notifications](http://npp-wiki.tuxfamily.org/index.php?title=Messages_And_Notifications) for the Notepad++ side.

#### SciMessager.cpp

Same kind, even more straightforward perhaps.

#### NppPluginMenu.cpp

Hardly anything to explain there, if you know how to code constructors and destructors. Could have been folded into the corresponding .h just as well.

#### SetingDlg.cpp

An excellent illustration of how tedious GUI programming can be. Most of the cde here is about reading the state of components (checkboxes, radio button groups, edit fields, ...) and store them into the corresponding members of the XBrackets class.

If you skip the SettigsDlgProc() function, which will require some more attention, you will find the rest of the file pretty repetitive. Having a reference on Windows controls at hand will help. This is not the place to summarize it. An important point to bear in mind is that, while controls are usually accessed through their handle, this is not true in a dialog box. Instead, controls have ***identifiers***, and Windows has a special set of routines to talk to identifiers, which may make code more expressive.

SettingsDlgProc() is the equivalent of a message loop for the options dialog box. Windows calls it whenever any of its child controls needs attention. Its prototype is defined in Windows.h.

The function reacts to three different messages. For anything else, it returns 0, telling the caller that it didn't handle the message; it is up to the caller to decide what to do.

* WM\_INITDIALOG: this message is sent to a dialog box procedure once all child controls have been created ad the box is visible. A good place to initialize child controls.
* WM\_NOTIFY: the modern way to send notifications, but no control on the dialog uses it (yet).
* WM\_COMMAND. A pivotal Windows message, as it carries notifications for all staple

controls those which already existed in the 16-bit era - as well as telling a window that a menu it owns was clicked. The low word of the wParam argument has the identifier of the sender, so we can switch on the sender. Recall that, when a window send a message to another, the latter cannot know who sent the message unless, like here, the arguments are so designed.

#### NpppPluginMenu.cpp

This initializes the FuncItems structure to be passed to the plugin loader, and defines what to do when an entry is clicked. Since the compiler considers a manifest string to define the contents pointed by a const char\*, you need the \_T macro provided in TCHAR.h to convert them into something the compiler will like.

The AllowAutocomplete() function is interesting because it shows that a plugin can do everything with the main Notepad++ window. Handle with care though.

#### XBracketsOptions.cpp

Another example of the repetitive coding the handling of persistent options requires. Parsing text, converting C values into and from text. Keep the file as a reference to copy and paste from when you face this requirement.

#### Main.cpp

This contains the exported stuff - remember that a plugin is a dll - which the plugin loader expects:

* the dll entry point. Initialize on load, clean up on free. The actual initialization tasks

are performed in a different file, to be inspected later.

* setInfo(): the loader passes a structure with the handle of the main window and the two views. Usual action for the routine is to store the data into the appropriate class members.
* getName() returns the plugin name.
* beNotified(): handler for Notepad++ notifications. It is hard to think of a case where this is not needed.
* messageProc(): the message loop procedure of the plugin main window, if any.
* getFuncArray(): returns the FuncItems structure the loader will use to initialize the plugin submenu.
* isUnicode(): this must be defined if and only if the plugin can talk with an Unicode Notepad++, and then must return true. As of v6.6.6, plugin must choose to be ANSI or Unicode, and cannot allow the difference dynamically.

#### NppPlugin.cpp

A helper file for Main.cpp. It contains obvious constructor and destructor for the CNppPlugin class, as well as what to do when the dll is attached or detached. Nothing special on detach.

The main job of the attach procedure is to determine where the plugin and executable are, possibly - like here - to figure out the path of auxiliary files.

#### XBrackets.cpp

The throbbing heart of the plugin, we'll deal with it in more detail.

##### nppCallWndProc()

A unified wrapper which allows to paper over a minute difference: CallWindowPrc(), which you use to send a message to a window message handler, exists in Ansi and Wide flavour.

##### nppNewWndProc()

This performs a subclassing of the main Notepad++ window. The reason for it is to be aware of all macro events, as the plugin considers it has to do something about it. When done, the routine forwards the message to its initial recipient. This is a very common Windows programming technique.

##### CXBrackets(), ~CXBrackets()

Pretty obvious, specially the destructor.

##### npppGetFuncsArray(), nppGetName()

Hardly more to say, they return what they are asked to.

##### beNotified()

This processes Notepad++ notifications. While a plugin may have an empty message processor, its notification processor will always have something to do. This very often takes the form of a switch on the notification code. Here, it is split into Notepad++ and Scintilla notifications - the comments in the lower part should read "Scintilla" instead of "Notepad++". Processing for each relevant notification is dispatched to a helper function.

##### OnNppSetInfo()

Not only this retrieves the handle information, but also asks whether it is talking to a Unicode window, and sets a flag accordingly.

##### OnNppBufferActivated(), OnNppFileOpened(), OnNppFileSaved()

The file type may change, and it matters, so we check it afresh.

##### OnNppReady()

Plugins shouldn't do much before they get this signal that Notepad++ is up and running, apart from internal initialization chores. Here, we subclass the main Notepad++ handle. This involves the WinAPI SetWindowLongPtr(), and it exists in two flavours.

##### OnNppShutdown()

Undo the subclassing, and don't forget to save options.

##### OnNppMacro()

We don't want the autocomplete feature to kick in in the middle of a macro recording: it would happen both on recording and replaying, resulting in a duplicate closing bracket on replay. This is why we trap this event and turn internal states on or off accordingly. All the subclassing stuff is for this function to work, since Notepad++ does not notify plugins that a macro recording or replay starts or stops. Sure, there is a WM\_ISCURRENTMACRORECORDED message, but polling Scintilla on every keyboard event would be inefficient.

##### OnSciCharAdded()

SCN\_CHARADDED is a Scintilla notification forwarded by Notepad++ and informing that a character was just typed. This has to be listened to in order to autocomplete a bracket. There is o course a fair amount of code, as the options in XBrackets are rather fine grained. The comments in the code make it reasonably easy to understand.

##### ReadOptions()

Prepares the ini option file and passes it to the CXBracketsOptions object.

##### SaveOptions(), UpdateFileType()

Get info and set internal state.

##### getEscapedPrefixPos(), isEscapePrefix()

Autocompletion of quotes should not trigger on an escaped quote, so we have to check whether the currently typed quote is escaped.

##### AutoBracketsFunc()

This handles actual insertion of the closing bracket, which should take place only in normal cases. While this is interesting code, it deals more with autocompletion than with plugin creation, so we won't cover it in detail.

##### getFileType()

This is about checking what the current file type is, so as to adjust autocompletion status.

Retrieved from "<http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code&oldid=3654>"

[Categories](http://npp-wiki.tuxfamily.org/index.php?title=Special:Categories):

* [Short Title(All)](http://npp-wiki.tuxfamily.org/index.php?title=Category:Short_Title(All))
* [Plugins](http://npp-wiki.tuxfamily.org/index.php?title=Category:Plugins)
* [Usage(Programming)](http://npp-wiki.tuxfamily.org/index.php?title=Category:Usage(Programming))
* [Programming](http://npp-wiki.tuxfamily.org/index.php?title=Category:Programming)
* [NP++ Dev](http://npp-wiki.tuxfamily.org/index.php?title=Category:NP%2B%2B_Dev)

## Navigation menu

### Personal tools

* [Create account](http://npp-wiki.tuxfamily.org/index.php?title=Special:UserLogin&returnto=Analysing+Plugin+Code&type=signup)
* [Log in](http://npp-wiki.tuxfamily.org/index.php?title=Special:UserLogin&returnto=Analysing+Plugin+Code)

### Namespaces

* [Page](file:///I:\Projects\VC\ContentsViewerPlugin\Release\Notepad++\Analysing%20Plugin%20Code_files\Analysing%20Plugin%20Code.htm)
* [Discussion](http://npp-wiki.tuxfamily.org/index.php?title=Talk:Analysing_Plugin_Code&action=edit&redlink=1)

### Variants

### Views

* [Read](file:///I:\Projects\VC\ContentsViewerPlugin\Release\Notepad++\Analysing%20Plugin%20Code_files\Analysing%20Plugin%20Code.htm)
* [View source](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code&action=edit)
* [View history](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code&action=history)

### Actions

### Search

窗体顶端



窗体底端

### NpWiki++

* [Main Page](http://npp-wiki.tuxfamily.org/index.php?title=Main_Page)
* [Help writing articles](http://npp-wiki.tuxfamily.org/index.php?title=Writing_Articles)
* [Tools and Helpers](http://npp-wiki.tuxfamily.org/index.php?title=Category:NpWiki%2B%2B)
* [Recent changes](http://npp-wiki.tuxfamily.org/index.php?title=Special:RecentChanges)

### [NP++ ARTICLES](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code)

* [by Title](http://npp-wiki.tuxfamily.org/index.php?title=Category:Short_Title(All))
* [by Topic](http://npp-wiki.tuxfamily.org/index.php?title=Category:Keywords)
* [by Usage](http://npp-wiki.tuxfamily.org/index.php?title=Notepad%2B%2B_Articles_Index_-_Class_Usage)
* [by Area](http://npp-wiki.tuxfamily.org/index.php?title=Area_Index_For_Notepad%2B%2B_Articles)

### [NP++ RESOURCES](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code)

* [Guides](http://npp-wiki.tuxfamily.org/index.php?title=NP%2B%2B_Guides)
* [Plugins](http://npp-wiki.tuxfamily.org/index.php?title=Plugin_Resources)
* [Project Support](http://npp-wiki.tuxfamily.org/index.php?title=Project_Resources)
* [User community](http://npp-wiki.tuxfamily.org/index.php?title=Category:User_Community)

### [NP++ LINKS](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code)

* [Gateway (Home)](http://notepad-plus-plus.org/)
* [News](http://notepad-plus-plus.org/news/)
* [Latest releases](http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=Releases#Version_5_series)
* [Forums](http://sourceforge.net/projects/notepad-plus/forums)
* [Downloads](http://notepad-plus-plus.org/download)
* [IdeaTorrent](http://sourceforge.net/apps/ideatorrent/notepad-plus/ideatorrent/)

### [INVISIBLE](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code)

### [Tools](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code)

* [What links here](http://npp-wiki.tuxfamily.org/index.php?title=Special:WhatLinksHere/Analysing_Plugin_Code)
* [Related changes](http://npp-wiki.tuxfamily.org/index.php?title=Special:RecentChangesLinked/Analysing_Plugin_Code)
* [Special pages](http://npp-wiki.tuxfamily.org/index.php?title=Special:SpecialPages)
* [Printable version](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code&printable=yes)
* [Permanent link](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code&oldid=3654)
* [Page information](http://npp-wiki.tuxfamily.org/index.php?title=Analysing_Plugin_Code&action=info)
* This page was last modified on 18 December 2009, at 23:21.
* This page has been accessed 12,695 times.
* Content is available under [GNU Free Documentation License 1.3 or later](http://www.gnu.org/copyleft/fdl.html) unless otherwise noted.
* [Privacy policy](http://npp-wiki.tuxfamily.org/index.php?title=NppWiki:Privacy_policy)
* [About Notepad++ Wiki](http://npp-wiki.tuxfamily.org/index.php?title=NppWiki:About)
* [Disclaimers](http://npp-wiki.tuxfamily.org/index.php?title=NppWiki:General_disclaimer)