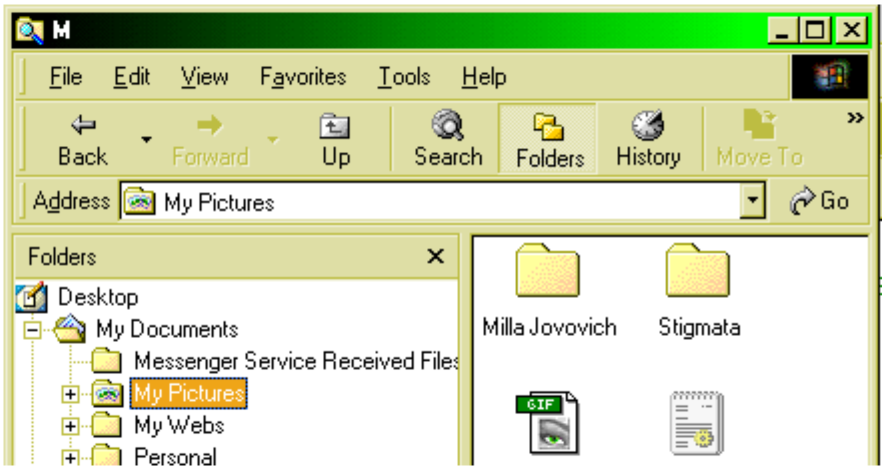
Articles / Programming Languages / C++

# How to Subclass Unicode Window from ANSI Application

**Mumtaz Zaheer**
27 Aug 2001

This article describe what steps are necessary to subclass UNICODE window from ANSI application.


M
File Edit View Favorites Tools Help
Back Forward Up Search Folders History Move To
Address My Pictures Go
Folders
Desktop
My Documents
Messenger Service Received Files
My Pictures
My Webs
Personal
Milla Jovovich
Stigmata
GIF

## Introduction

I am thankfull to Jim Barry and Brian Ross. Both helped me in writing this article. Actually the solution provide by me is the Solution provided by Jim Barry.

Jeffrey Richter in his book *Windows 95 A Developer's Guide* says:

*Microsoft Windows supplies a number of ready-to-use window classes, including listboxes, comboboxes, and scrollbars. These controls are intended to be general and feature-laden enough for* *use in any application. Sometimes, however, you may wish that these controls had slightly different behavior. One solution to this problem is to design your own control from scratch. This task is usually significant; however, it would be nice if the source code to the Windows built-in controls was freely available. Unfortunately, Microsoft does not distribute these sources. So we must take a different approach to this problem. Window subclassing and superclassing come to the rescue here and save us from reinventing the wheel. Subclassing and superclassing can also be used on your own window classes, including application windows and custom controls that you create. However, using subclassing and superclassing on your own code is usually not necessary because you have easy access to your own source code and you can easily modify your own source code to do exactly what you desire.*

We achieve subclassing by replacing the address of that window's `WndProc`. For that we must be a part of the process that created the window. There are numerous method to get in to any process address space, like the `WriteProcessMemory()` API, \HKLM\Software\Microsoft\Windows \AppInit_Dll (for NT only) or Hooks.

At this time most of the API functions have ANSI and UNICODE versions. For example

```
#ifdef UNICODE
#define GetWindowTextLength  GetWindowTextLengthW
#else
#define GetWindowTextLength  GetWindowTextLengthA
#endif
```

So if our application is ANSI `GetWindowTextLenghtA` will be used otherwise `GetwindowTextLengthW` will serve us.

Once we get into address space of our desire process. We use `SetWindowLong` to replace the original WndProc, that will return us original `WndProc` address in response. Latter we can use that address to pass messages to original WndProc with the help of `CallWindowProc()`.

Kyle Marsh in his article *Safe subclassing in Win32* says:

*The value returned from* `SetWindowLong` *and* `GetClassLong` *may not be a pointer to the previous window procedure at all. Win32 may return a pointer to a data structure that it can use to call the actual window procedure. This occurs in Windows NT™ when an application subclasses a Unicode™ window with a non-Unicode window procedure, or a non-Unicode window with a Unicode window procedure. In this case, the operating system must perform a translation between Unicode and ANSI for the messages the window receives.*

So `CallWindowProc()` is supposed to help us with ANSI/UNICODE. ***But it never does.***

From here begins my story.

For some reason I came to the decision that I should subclass the Windows Explorer window. I did that with the help of `SetWindowLong()` and `CallWindowProc()`. I use same code over Win9x and NT/2k. On NT/2K I faced a strange problem of having the wrong window title. After subclassing that subclassed explorer window display its title is single letter i.e. "Exploring System (C:)" is "E". I checked the `WM_SETTEXT` message it receives correct text in lParam.

I start searching for help as I could not figure it out by myself. I posted questions to Microsoft news groups. A guy name Jim Barry helped me by saying:

*This is a Unicode issue. There seems to be a bug in Explorer that prevents mixed ANSI/Unicode subclassing from working properly. Having spent many long hours solving this problem, I can tell you there is only one solution. When subclassing the Explorer window, first check if the window is Unicode by calling `IsWindowUnicode`. If it is Unicode, then install a Unicode wndproc using `SetWindowLongW`. If it is ANSI, then install an ANSI wndproc using `SetWindowLongA`. You need to have your ANSI wndproc forward to Explorer's wndproc using `CallWindowProcA`, whereas your Unicode wndproc should use `CallWindowProcW`. It's a right pain having to provide two wndprocs, but there you go.*

So I change my code to

```
if(IsWindowUnicode(hwndToExplorer))
{
    pfnWndProc = (WNDPROC)SetWindowLongW(hwndToExplorer, GWL_WNDPROC,
                (LPARAM)(WNDPROC)Explorer_SubClassWndProc);
}
else
{
    pfnWndProc = (WNDPROC)SetWindowLongA(hwndToExplorer, GWL_WNDPROC,
                (LPARAM)(WNDPROC)Explorer_SubClassWndProc);
}
```

and

```
if(IsWindowUnicode(hwnd))
{
    lResult = CallWindowProcW((WNDPROC) (DWORD)pfnWndProc, hwnd,
                        uMsg, wParam, lParam);
}
else
{
    lResult = CallWindowProcA((WNDPROC) (DWORD)pfnWndProc, hwnd,
                        uMsg, wParam, lParam);
}
```

Hence I got the what I want to.

*Errors and Omissions are expected/accepted* :)

## License

Written By

**Mumtaz Zaheer**

Web Developer

Pakistan

Mumtaz Zaheer is working as Senior System Analyst with Information Architects, Pakistan (http://www.info-architects.com/).

## Comments and Discussions

**10 messages** have been posted for this article Visit **https://www.codeproject.com/Articles/984/How-to-Subclass-Unicode-Window-from-ANSI-Applicati** to post and view comments on this article, or click **here** to get a print view with messages.