RegistryClass MFC Library Manual

Simplified Registry Access with MFC Data Types

Introduction

This manual was made to help developers learn how to link-to and use the RegistryClass MFC Library. This library was developed to help developers access the registry in an easier and object-oriented manner.

The RegistryClass library only exposes a portion of the Windows Registry Functionality, but exposes more than what is required for standard Windows Applications. It was developed for Windows 9x and Windows NT based versions of Windows in mind, so it is ready for use in multi-user aware applications. The library was also create to support Windows CE for embedded systems. This includes Windows Mobile 2003 and SmartPhone 2003.

This manual is laid out in section and provides both a step-by-step approach to using the RegistryClass library and an API reference for more advanced information about using the RegistryClass library.

Before you Begin

Before you begin using the RegistryClass library, you must make sure you have the required system specification, components, and application installed on your development system.

System Requirements

Using RegistryClass in your own projects requires the same system specifications as the development environment that you are developing your application in. To compile the RegistryClass library from source you will need the following:

- Microsoft Visual C++ version 6.0 with the MFC libraries installed.
- Microsoft Windows 95 or NT4 or better.
- Meet the minimal system specifications for Visual C++ and Operating System you are using.

The recommended setup is as follows:

- Microsoft Windows 2000 with Service Pack 4 or better.
- 128MB of RAM.
- Microsoft Visual C++ 6.0 with Service Pack 6 installed.
- Extra 10MB or hard disc space.
- Microsoft eMbedded Visual C++ with Service Pack 4 for Windows CE, Pocket PC, and SmartPhone development..

Downloading the Latest Copy

Before you begin developing any applications with RegistryClass, you should ensure that you have the latest copy. This will help make sure that you applications are not going to be affected by any bugs or security issues found in older versions of RegistryClass. RegistryClass is designed to be backwards compatible with older versions, so if your application isn't updated or linked against the latest version, it will still work properly with newer version provided that your application doesn't use any known bugs to produce any desired affect.

To download the latest copy of the RegistryClass library, visit http://sourceforge.net/projects/regclass/. You should ALWAYS use binary releases to ensure proper version control of the library on target systems.

Getting Started

To develop application to use the RegistryClass library, you must first download the latest binary distribution RegistryClass.

Step 1: Extracting the Library and Headers

Once you have downloaded the binary release, you should create a directory where you will store RegistryClass and the projects for any applications you will be developing using RegistryClass. Using any available ZIP utility, unzip the binary release to that directory. You MUST use your ZIP utilities "Extract to {Achrive Name as a Folder}" feature to ensure that the files are extracted to the "RegistryClass" directory.

Figure 1 - How to extract the RegistryClass Source Project

Step 2: Including RegistryClass in Your Project

To use the RegistryClass library, you must first make some changes to your Release and Debug settings for you project. First, you will need to open the settings dialog box for your project. Right-click on the your Projects' item in the Workspace tree, then select Settings from the menu.

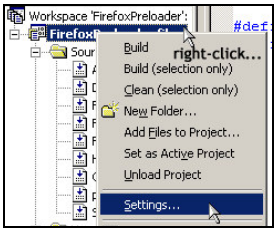

Figure 2 - Project Settings

Next, make sure that your project's settings are linked with the MFC library using the "Shared DLL," there are issues using the "Static Library."

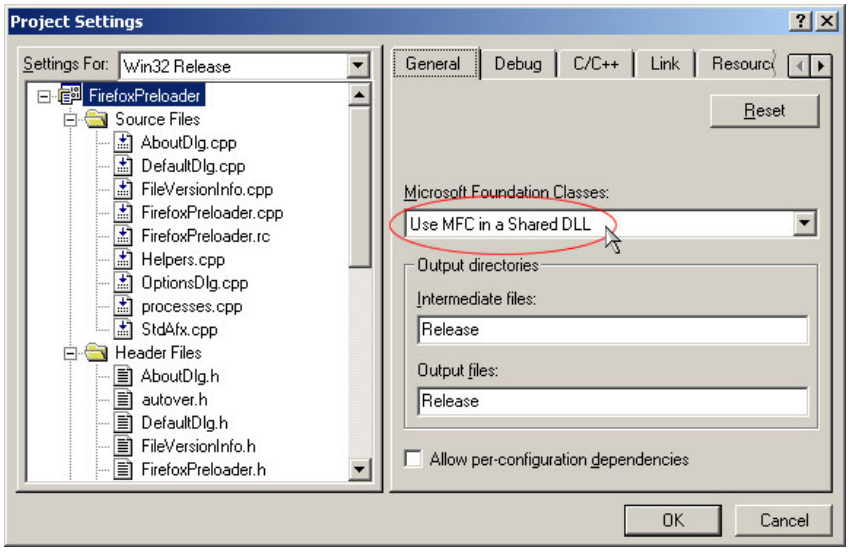

Figure 3 - Use the MFC Shared DLL

Then, you need to add the RegistryClass's output directory to your project's "Additional include directories:" under the "C/C++" tab in the "Preprocessor" category. You may need to use the full path to these directories.

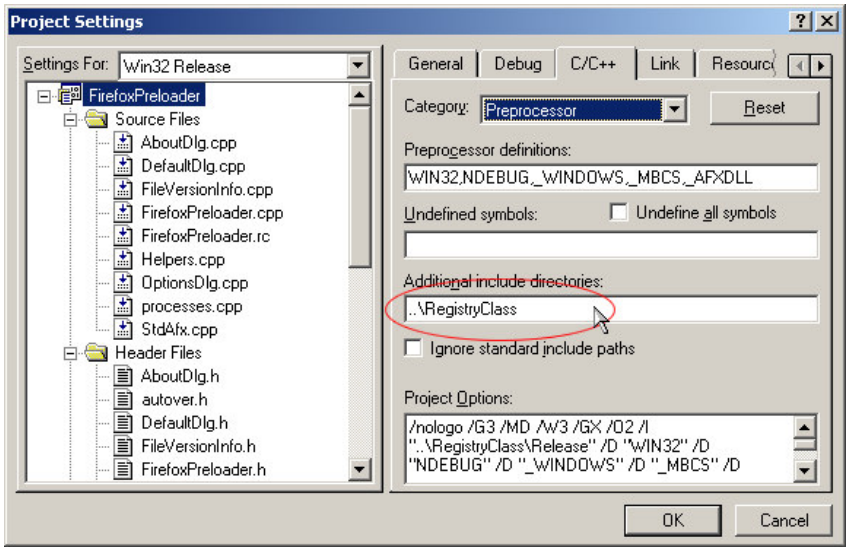

Figure 4 - Add the include directory

Next, you need to add the RegistryClass's output directory to your project's "Additional library path:" under the "Link" tab in the "Input" category. You may need to use the full path to these directories.

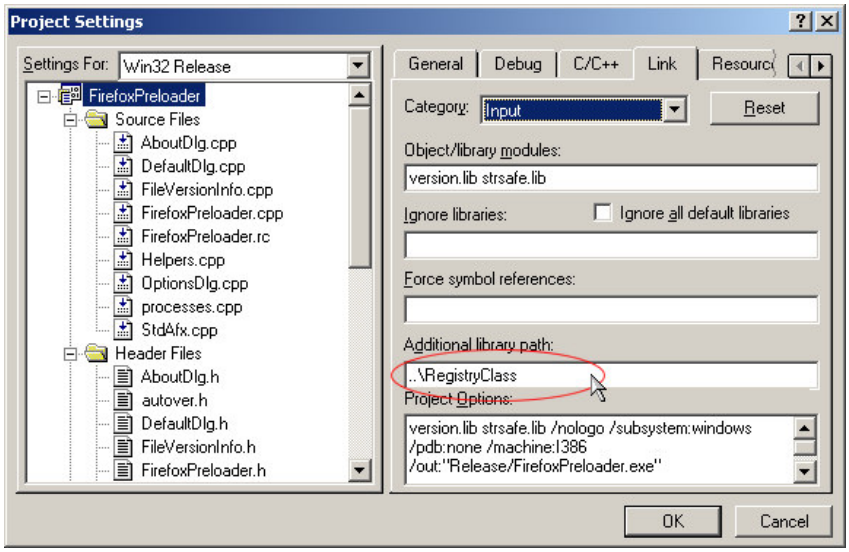

Figure 5 - Add the library path

Now, you will need to add a step to your "Post-build steps," under the tab of that name. This will copy the RegistryClass DLL to the output directory of your project. If you are compiling your project as an ANSI Windows Applications, you will use "regclass.dll;" but if you are compiling your project as a UNICODE Windows Application, you will use "regclassu.dll." With the "Post-build steps" tab selected, you will need to double-click on the empty dotted rectangle to add a new step. When the cursor appears type in, without the '', '`copy` {enter path to RegistryClass here}`\regclass.dll`

`$(OutDir)'` for ANSI projects and '`copy {enter path to RegistryClass here}\regclassu.dll $(OutDir)`' for UNICODE projects.

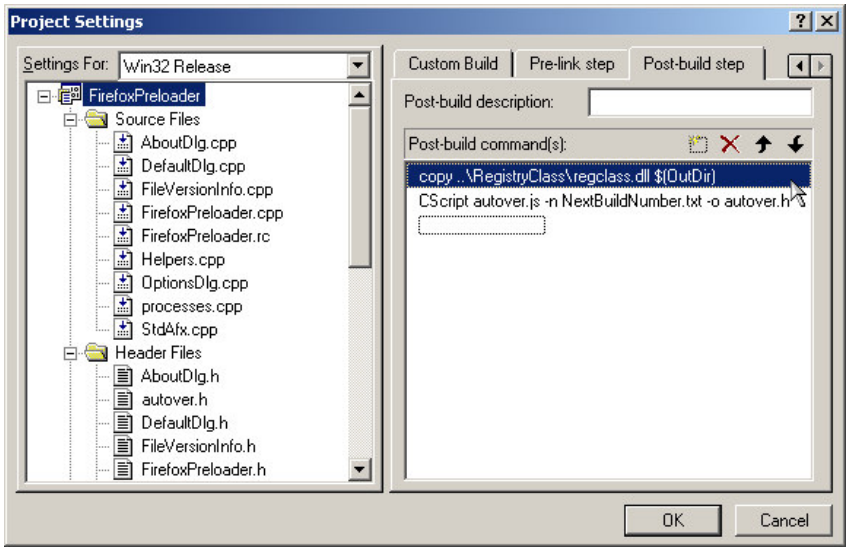

Figure 6 - Add the post-build step to copy the RegistryClass DLL

Finally, you need to do this for all the settings of your project. To view and change the other settings, use the dropdown in the Project Settings dialgo.

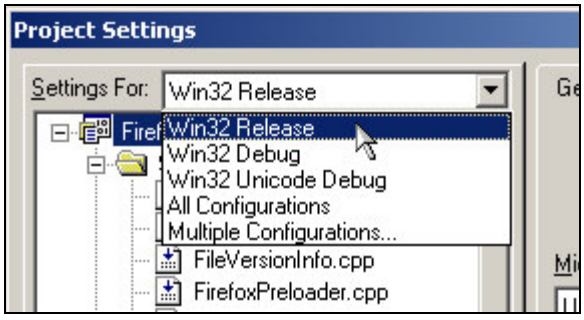

Figure 7 - Repeat for other project settings

Step 3: Adding RegistryClass to your Source Code

Once you have included RegistryClass into your project settings, its now time to add it to your source code. This is a simple process. If you are using the "stdafx.h" header, then you will need to add the line following line somewhere in that file:

```
#include <RegistryClass.h>
```

If you are not using the "stdafx.h" header, then you will need to add the aforementioned line to any files that will use the RegistryClass functions.

If you are not creating an MFC application, but wish to use RegistryClass, you will need to add the following line to either the "stdafx.h" header or any files using RegsitryClass:

```
#include <afxwin.h>
```

If you are able to compile your application without using any RegistryClass function after adding this lines, then you are ready to go. If you get any errors related to any "afx"

please refer to your Visual C++ Documents on how to correct these errors. A rule of thumb is that in any file that uses “afxwin.h” you shouldn’t use “windows.h.”

Using RegistryClass in Your Projects

Now that all the needed settings and includes have been taken care of, it's time to make use of the RegsitryClass library. To begin, you need to create a variable with for the CRegistry class. To do this, place in any function or file that you want to access the registry:

```
CRegistry cReg;
```

cReg is just an example variable name, but this is the class where all access to the Windows Registry begins. Once you have this you can now begin working with the Windows Registry.

Opening Keys

Opening a Registry Key is simple, simple use the path to the key to open it. The Hive Key that the key you want to open is a selected as a member function of the CRegistry class. To open a key, use a line like the following:

```
CRegistryKey *myKey;
myKey = cReg.KeyLocalMachine()->OpenKey(_T("SOFTWARE\\MyKey"));
```

Once you done with the key, you need to delete the opened instance with the following line:

```
delete myKey;
```

Reading Values

To read a simple value like Strings and Dwords, you need only use the GetStringValue and GetIntegerValue member functions of an opened key. Examples:

```
DWORD dwInteger;
CString szString;
myKey->GetIntegerValue(_T("My Integer Value"), &dwInteger);
myKey->GetStringValue(_T("My String Value"), &szString);
```

Writing Values

To write values to the registry, you need only use the SetStringValue and SetIntegerValue member functions of an opened key. Examples:

```
myKey->SetIntegerValue(_T("My Integer Value"), 1);
myKey->SetStringValue(_T("My String Value"), _T("My String"));
```

Creating a Key

To create a Registry Key, you simply use the CreateKey function much like you do the OpenKey function. This will create a new key if it doesn't exist, or open the key if it does.

```
CRegistryKey *myNewKey;
myNewKey = cReg.LocalMachine()->CreateKey(_T("SOFTWARE\\MyNewKey"));
```

Distributing RegistryClass

When distributing RegistryClass with your application, you must first abide by the GNU Lesser/Library General Public License.

RegistryClass should always be installed as a Shared Library in the Windows' System directory. Your installer should insure that no other application is also using the RegistryClass library before removing it, and overwrite only older versions.