# Appendix 1 – Installation Guide (Local-Machine Modules in Python)

**Note**: This section is based on the Windows operating system.

**Step 1 – Download and Install Slicer Client**

1. Download the Slicer4 pre-compiled binaries from <http://download.slicer.org/>.
   1. Stable: A safe and stable version. However, it does not support extensions/modules that were created or updated after it was compiled
   2. Nightly: A new version is released every night. Provides access to all new extensions and modules but is not 100% stable.

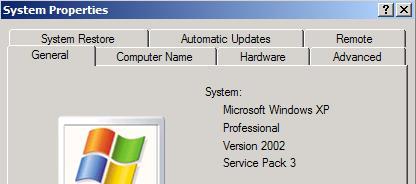
**Note:** Windows users will have to choose between 32 bit and 64 bit. This choice will depend on the local machine’s version of windows. To find this version, use the following steps:

* Access the **Start** menu
* Right click on **My Computer** and select **Properties**
* Look under the **General** tab to find the version of Windows.
* Use the table below to determine which version of Slicer to download

**Note 2:** The 32-bit version of Slicer does **not** support Slicer extensions. However, modules can be written using the 32-bit version and can also be converted into extensions without problems.

|  |  |  |
| --- | --- | --- |
| **Operating System** | **System** | **Version to Download** |
| Windows XP | If ‘x64 Edition’ is not listed under **System** | 32-bit |
| If ‘x64 Edition’ is listed under **System** | 64-bit |
| Windows Vista / 7 | If ’32-bit Operating System’ is listed next to **System Type** | 32-bit |
| If ’64-bit Operating System’ is listed next to **System Type** | 64-bit |

The image below shows an example of a 32-bit Windows XP platform.



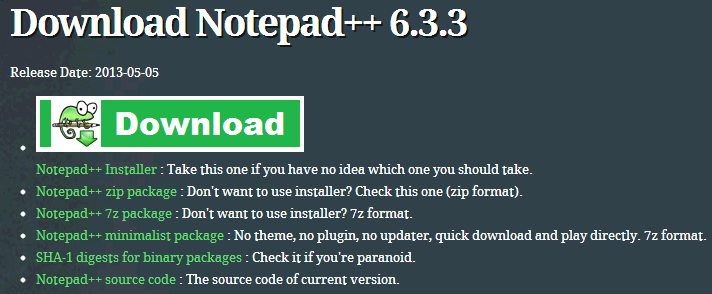
**Step 2 – Install Slicer**

1. Run the **Slicer executable file.**
2. Proceed with normal installation. None of the default settings have to be changed.

**Step 3 – Download Source Code Editor (Optional)**

It is recommended that python developers have a source code editor to help in the code development process. The editor detailed in this guide is Notepad++.

1. Go to the Notepad++’s download page at <http://notepad-plus-plus.org/download>.
2. Download **Notepad++ Installer.**



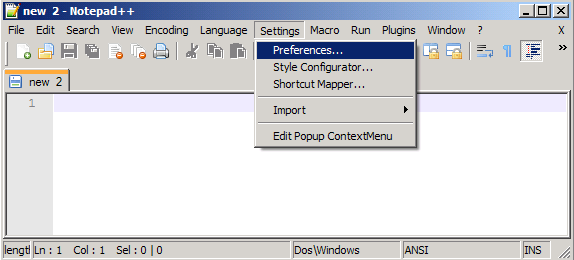
**Step 4 – Install Notepad++**

1. Run the Notepad++ installer **(npp.6.x.x.Installer.exe).**
2. Use the **Default** settings for all installation prompts.

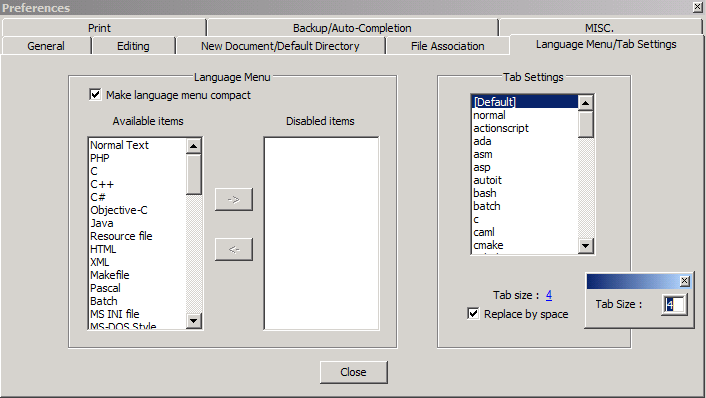
**Step 5 – Configure Notepad++**

This step helps configure Notepad++ to meet the Slicer development team’s request for all indentations in python code to be done via 4 spaces.

1. Start **Notepad++.**
2. Click on **Settings** at the menu bar and select **Preferences...**

****

1. Select the **Language Menu/Tab Settings** tab.
2. On the right side of the menu, check the **Replace by space** box.
3. Click on the number next the **Tab size**. Set the value in the pop-up that appears to **4**.



**4**

**3**

**2**

**1**

**-- End of Appendix 1 --**

# Appendix 2 – Installation Guide (Extensions and Local-Machine Modules in C++)

**Note**: This section is based on the Windows operating system.

|  |
| --- |
| Required Files |
| Slicer Source Code |
| CMake |
| Git |
| Subversion |
| Microsoft Visual Studio |
| Qt Libraries (v4.7.4) |

All of the following files have to be manually downloaded and installed except for the Slicer source code that will be obtained through Git.

**Step 1 – Install Microsoft Visual Studio**

It is recommended that users make use of **Visual Studio 9 2008** for coding and building Slicer modules. This is the version supported by Slicer and will ensure that there are no errors when compiling the Slicer source code, modules and extensions.

**Visual Studio 2010** is another platform that developers can use. However, note that problems may be encountered when using this development environment to build the Slicer source code.

**Step 2 – Download Service Pack 1 for Visual Studio 2008**

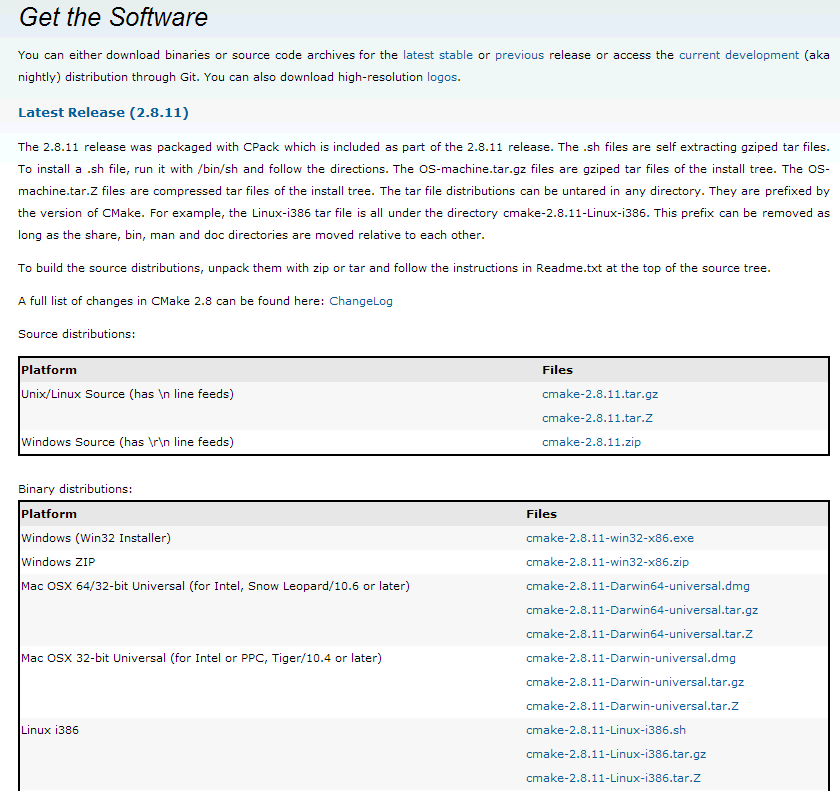
**Note:** This step is only for developers who are using Visual Studio 9 2008.

1. Download Service Pack 1 from <http://www.microsoft.com/en-us/download/details.aspx?id=10986>
2. Run the **VS90sp1-KB945140-ENU.exe** executable.
3. **Accept the license terms** and use the **default** settings.
4. Installer will automatically download and install the service park.

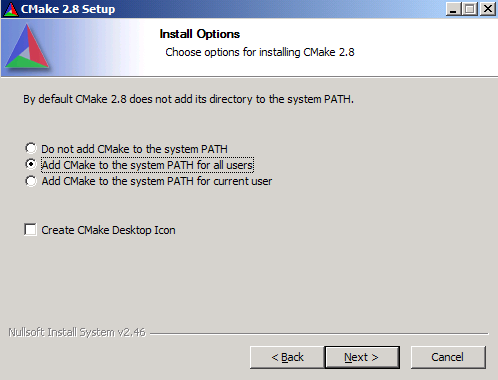
**Step 3 – Download and Install CMake**

1. Download CMake from <http://www.cmake.org/cmake/resources/software.html>.

**Note:** Download the **Binary distribution** of CMake and not the Source distributions.



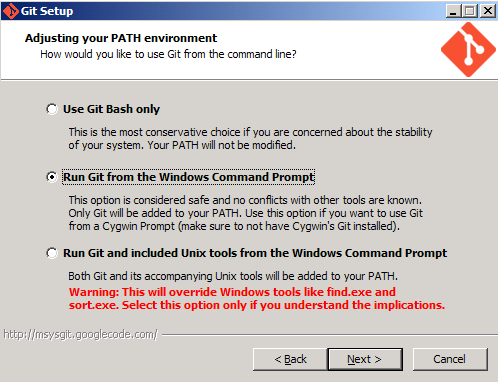
1. Run the CMake installer (**cmake-x.x.x-win32-x86.exe**).
2. Click on **Next** and then **I Agree.**
3. When asked about system PATH, change the default “Do not add CMake to the system PATH” to **“Add CMake to the system PATH for all users”**.



1. Finish the installation by clicking on **Next > Install > Finish**.

**Step 4 – Download and Install Git**

1. Download the latest stable release of git from <http://git-scm.com/downloads>.
2. Run the git installer (**Git-1.8.1.2-preview20130201.exe**).
3. Follow the prompts and click **Next > Next > Next**. No settings need to be changed.
4. When prompted about “Adjusting your PATH environment”, change the default “Use Git Bash only” to **“Run Git from the Windows Command Prompt”** and click **Next**.



1. Leave the button on the default **“Checkout Windows-style, commit Unix-style line endings”** and click **Next** to install.

**Step 5 – Download and Install GitHub for Windows (Optional)**

**Note: This step is recommended for users creating extensions. C++ module developers can skip this step.**

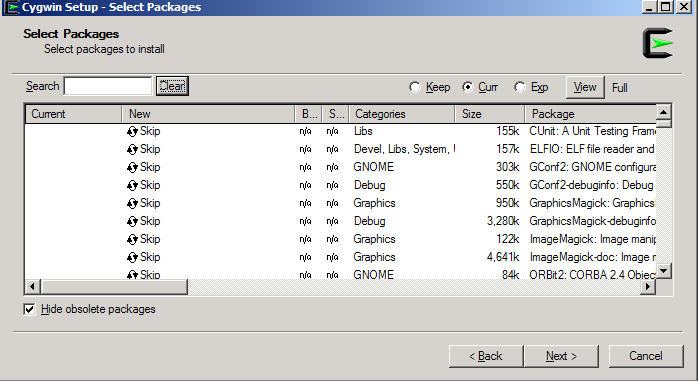
1. Create a GitHub account at <http://github.com>.
2. Download the GitHub for Windows installer from <http://windows.github.com/>.
3. Run **GitHubSetup.exe**.
4. Click on **Install**. The installer will automatically download and install GitHub for Windows.
5. When the installation is complete, **login** to the GitHub for Windows client using the GitHub account made in step (1).

**Step 6 – Download SVN (Cygwin SVN)**

1. Download the Cygwin installer file from <http://cygwin.com/setup.exe>.
2. Run **setup.exe** and click **Next**.
3. For download source, use the default of **‘Install from Internet’** and click **Next**.
4. The defaults for Installation Directory and Local Package Directory do not have to be changed unless the user wants to specify directories. Click on **Next > Next**.
5. Under Select Your Internet Connection, use the default **‘Direct Connection’** and click **Next**.
6. Select **any** download mirror from the list.

**Tip:** Select a mirror that is in the same region for the best download speeds. For example, users in Australia should try to find a mirror that ends with .au.

1. When the initial download is complete, users will be presented with a Select Packages screen.
2. Click on **View** at the top right once. This will change View to **Full**.

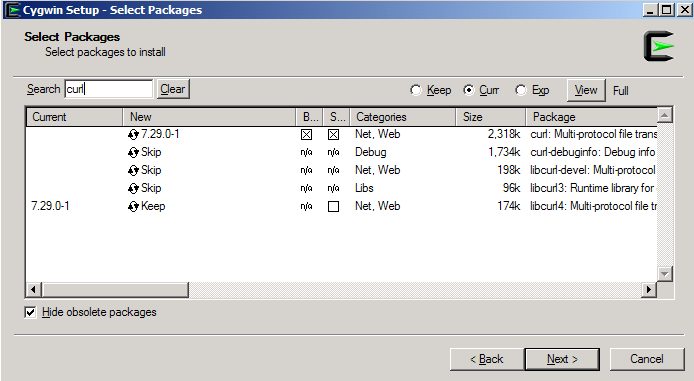


1. In addition to the default packages that are to be installed, also download the following additional packages:

* curl
* cvs
* git
* subversion
* tcltk
* unzip
* zip

1. Use the search bar at the top left to search for each of the above packages. The package names are on the right and will be labeled ‘Skip’ on the left. For each of these packages, **click on ‘Skip’ once** so that a version number appears instead.

**Note:** If the search returns multiple instances of a file, select only the one that matches the packages listed above exactly. For example, searching for ‘curl’ will return multiple items. Users are to install only the ‘curl’ package and not ‘curl-debuginfo’. An example of this is shown in the figure below.



1. When all the packages have been selected, click on **Next** to start the installation.
2. When the installation is complete, **uncheck ‘Create icon on desktop’**.
3. Click on **Finish** to end.

**Step 7 – Download SVN (SlikSVN)**

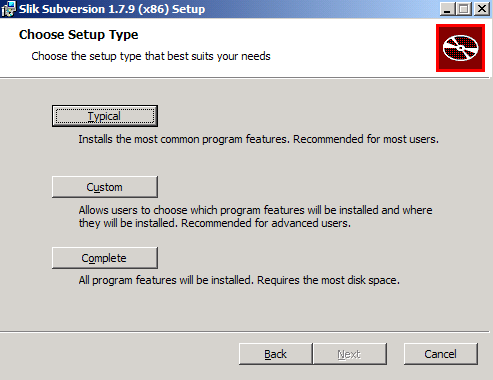
1. Download SlikSVN from <http://www.sliksvn.com/en/download>.

**Note:** Windows users will have to choose between 32 bit and 64 bit. This choice will depend on the local machine’s version of windows.

* Access the **Start** menu
* Right click on **My Computer** and select **Properties**
* Look under the **General** tab to find Windows version.
* Use the table below to determine which version of SlikSVN to download

|  |  |  |
| --- | --- | --- |
| **Operating System** | **System** | **Version to Download** |
| Windows XP | If ‘x64 Edition’ is not listed under **System** | 32-bit |
| If ‘x64 Edition’ is listed under **System** | 64-bit |
| Windows Vista/7 | If ’32-bit Operating System’ is listed next to **System Type** | 32-bit |
| If ’64-bit Operating System’ is listed next to **System Type** | 64-bit |

1. Run the installer file (**Slik-Subversion-1.x.x-win32.msi)**.
2. Click on **Next**.
3. **Accept the license agreement** and click on **Next**.
4. Change the install directory if desired (but not necessary) and click **Next**.
5. Click on **Typical** and then click **Install**.



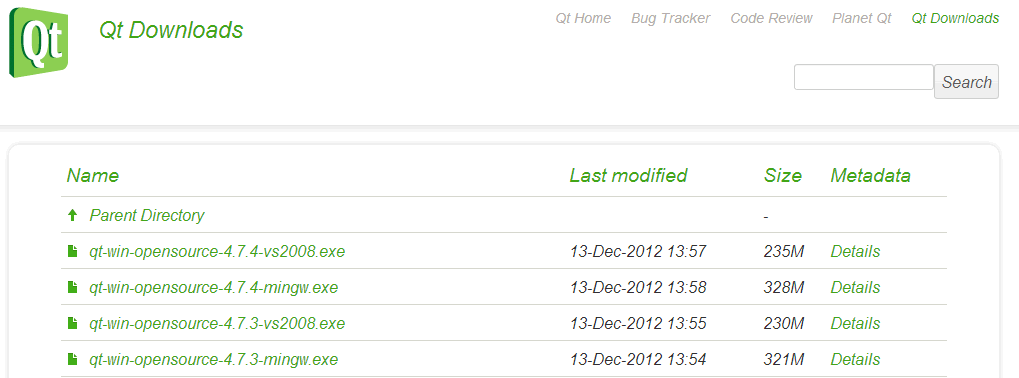
1. Click on **Finish** to end the installation.

**Step 8 – Download and Install Qt Libraries**

**Note:** The latest version of the Qt libraries that is compatible with Slicer is v4.7.4.

1. Download Qt 4.7.4 from <http://download.qt-project.org/archive/qt/4.7/>.

**Note:** Download **qt-win-opensource-4.7.4-vs2008.exe**. Ensure that the version is 4.7.4 and that the file name has ‘vs2008’ in it (to ensure the file is compatible with Visual Studio 2008 and Visual Studio 2010)



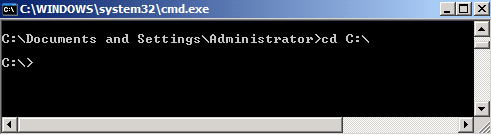
1. Run the installer (**qt-win-opensource-4.7.4-vs2008.exe**).
2. Click on **Next > Next > Accept terms of License Agreement > Next**.
3. When asked to choose components, leave **File Associations** checked and click **Next**.
4. Change the install directory if desired and click **Next** to start the install.

**Step 9 – Obtain Slicer Source Code**

1. Open up the Windows command prompt via **Start > Run**. Type in **cmd** and click **OK.**
2. Navigate to the directory in which the Slicer source code is to be saved. This is done bytyping the following into command prompt

**cd Directory**

An example of navigating to the directory **C:\** is shown in the figure below



1. Download the Slicer source code by typing the following into command prompt

**git clone git://github.com/Slicer/Slicer.git**

If this is typed into the command prompt shown above (at directory C:\), the source code will be stored in C:\Slicer\.

**Note:** Leave command prompt alone until the download is complete. While the download is running, an underscore ( \_ ) will be flashing at the bottom of the screen. When the download is complete, a new line with the current directory (for example **C:\>**) will appear at the bottom.

1. Setup Slicer and link it to the version control system. This is done through the following commands.

**Note:** Each line is separate. Press Enter at the end of each line.

**cd Slicer**

**git svn init http://svn.slicer.org/Slicer4/trunk**

**git update-ref refs/remotes/git-svn refs/remotes/origin/master**

**git checkout master**

**git svn rebase**

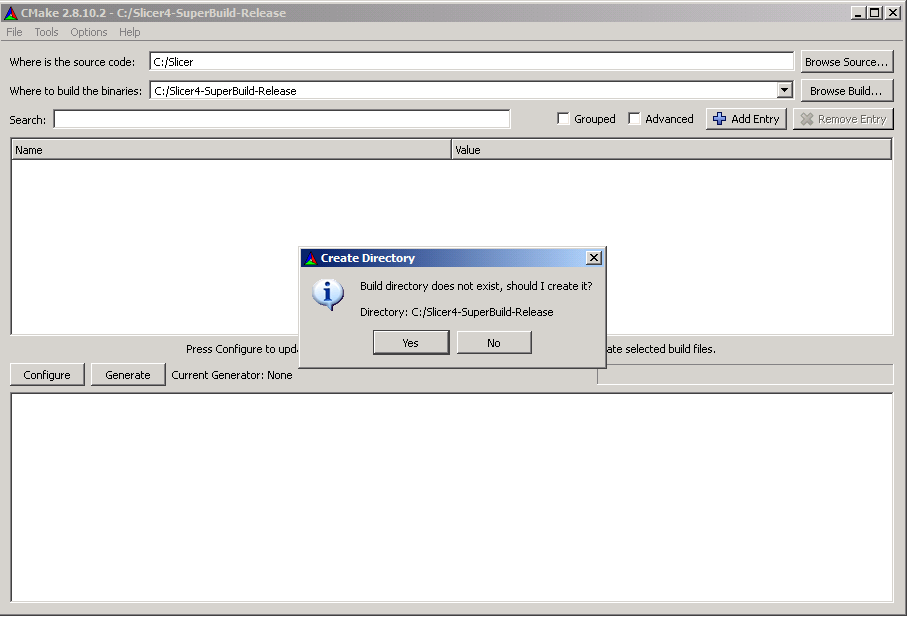
**Step 10 – Build Slicer**

1. Run CMake-gui via **Start > All Programs > CMake 2.8 > CMake (cmake-gui)**.
2. Under “**Where is the source code**” put the directory where the Slicer source code is saved.

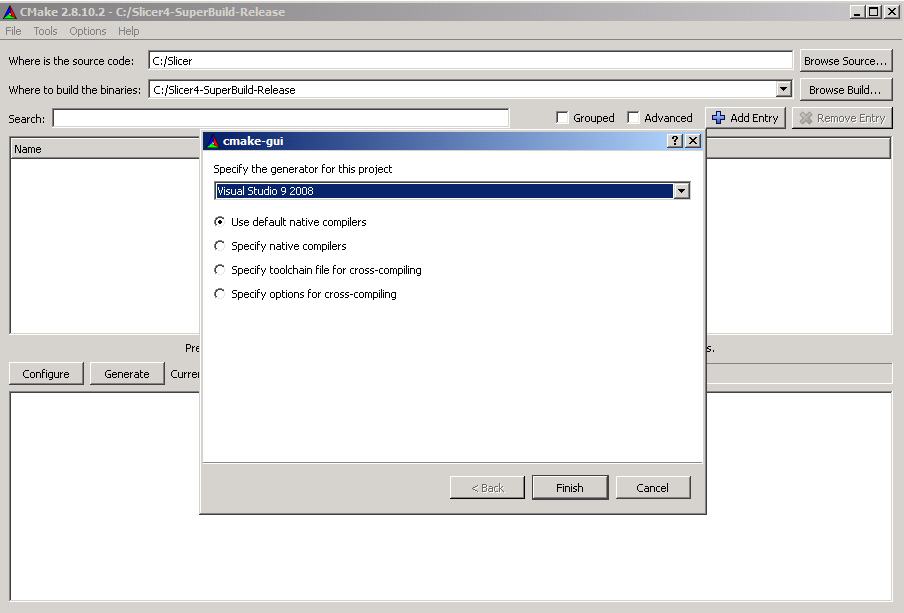
**Note:**  Git will **always** put the course code inside a **/Slicer** folder. So if git cloned the source code in **C:/Code**, the Slicer code will be in **C:/Code/Slicer.**

1. Under “**Where to build the binaries**”, create a directory in which to save the Slicer files. This can be named anything, but this example will use C:/Slicer4-SuperBuild-Release as the binary storage location.

**Tip:** It is recommended that the binaries are saved to a new folder. This will prevent any file mix-ups in the future. After the next step, CMake will prompt the user to confirm whether or not it should make the new directory. Click **Yes**.



1. Click on **Configure**.
2. Select the generator that is installed (Visual Studio 9 2008 if VS2008 is installed).
3. Use the default **Use default native compilers**
4. Click on **Finish**.



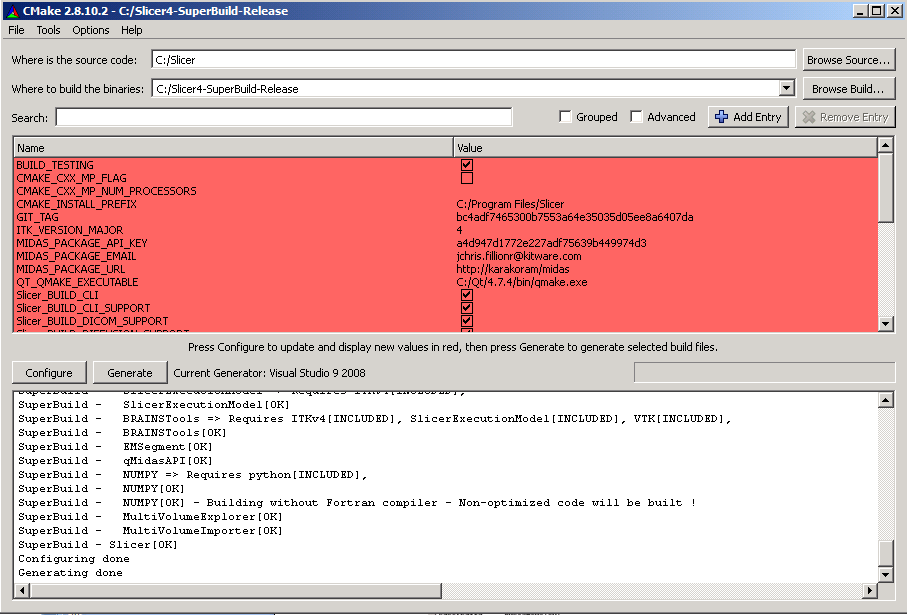
**3**

**2**

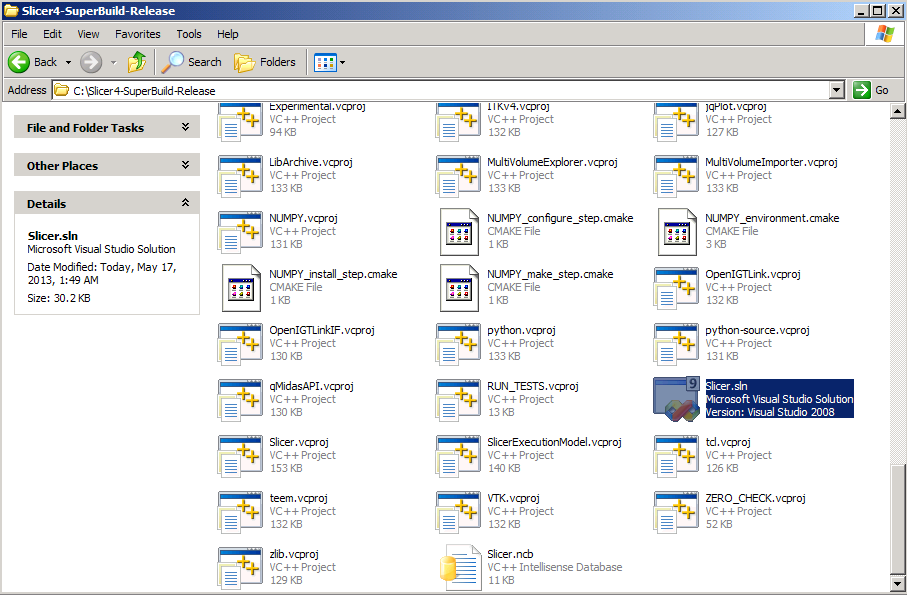
**1**

1. When CMake says that “Configuring done”, click on **Generate**.

**Note:** There will be unchecked boxes in the top menu. Do **not** change these.

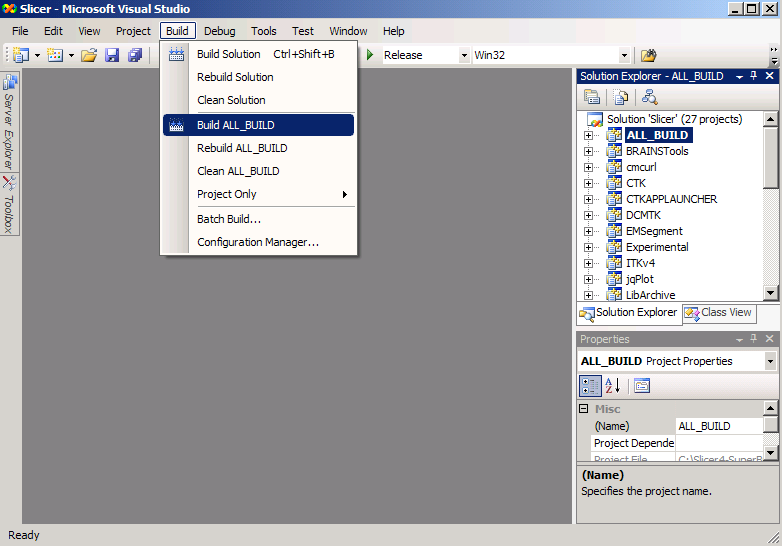


1. When CMake says that Generating done, close CMake.
2. Go to the folder where the Slicer binaries were built in CMake (Slicer4-SuperBuild-Release based on example above). Run **Slicer.sln**.



1. In Solution Explorer on the right, click on **ALL\_BUILD**.
2. Click on **Build** at the top menu then **Build ALL\_BUILD**.

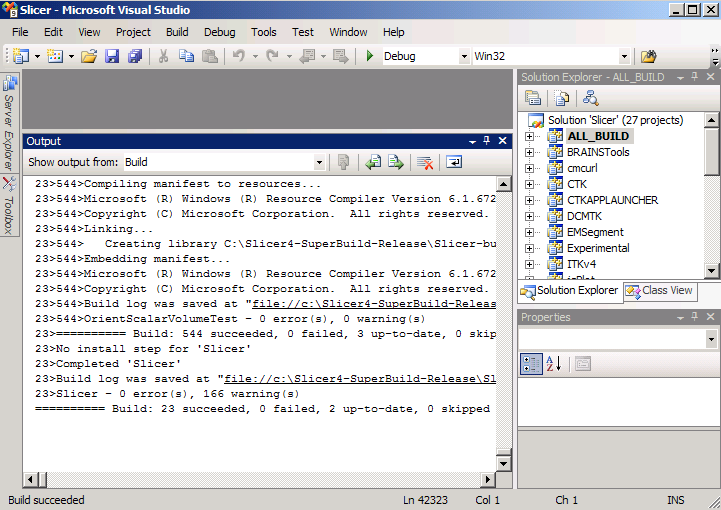
**2**



**3**

**1**

1. Wait till Visual Studio says **Build Succeeded** at the bottom left. Note that this can be a time-consuming process.

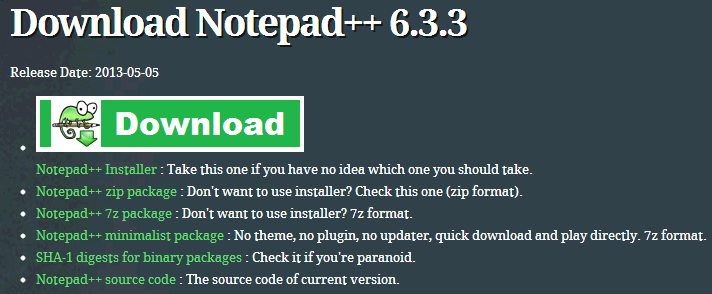


**Note**: To access the Slicer executable, go back to the folder where CMake saved the binaries. Go into the folder **Slicer-build** and run **Slicer.exe.**

**Step 12 – Download & Install Source Code Editor (Optional)**

It is recommended that python developers have a source code editor to help in the code development process. The editor detailed in this guide is Notepad++. C++ developers can skip the rest of **Appendix 2**.

1. Go to the Notepad++’s download page at <http://notepad-plus-plus.org/download>.
2. Download **Notepad++ Installer.**



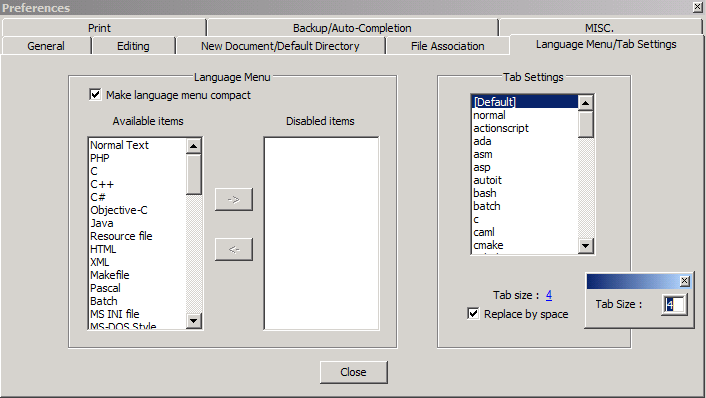
**Step 14 – Install Notepad++**

1. Run the Notepad++ installer **(npp.6.x.x.Installer.exe).**
2. Use **Default** settings for all installation prompts.

**Step 15 – Configure Notepad++**

This step helps configure Notepad++ to meet Slicer’s request for all indentations in python code to be done via 4 spaces.

1. Start **Notepad++.**
2. Click on **Settings** at the menu bar and select **Preferences...**
3. Select the **Language Menu/Tab Settings** tab.
4. On the right side of the menu, check the **Replace by space** box.
5. Click on the number next the **Tab size**. Set the value in the pop-up that appears to **4**.



**4**

**3**

**2**

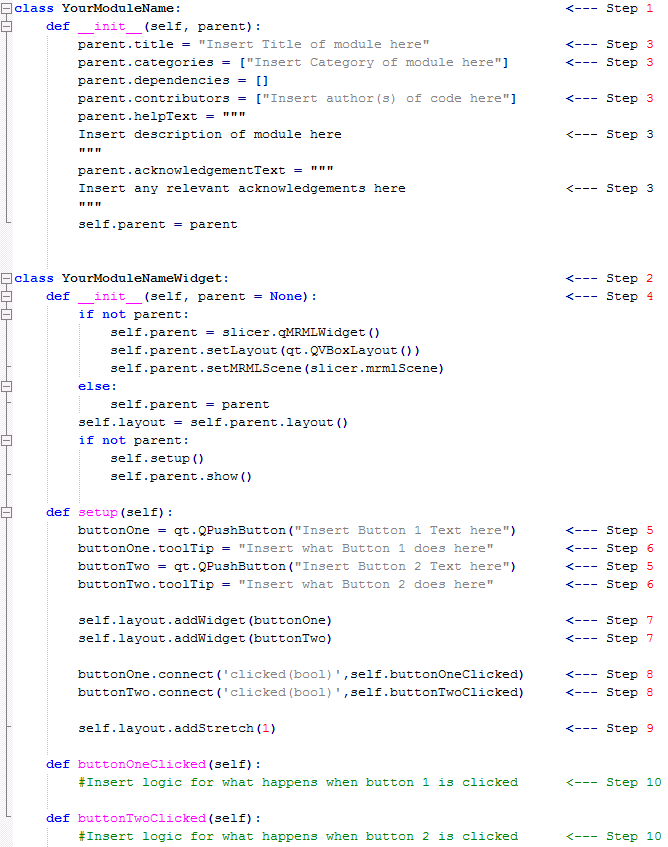
**1**

**-- End of Appendix 2 --**

# Appendix 3 – Slicer Interface Code – Python Template

The Slicer Interface code consists of a Parent class and a Widget class as shown below. The following pages include step-by-step information on how to use this template.

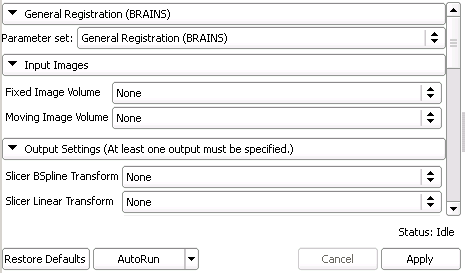
Note that these go **at the front** of the module.



1. Replace **YourModuleName** with the name of the module. **Important Note:** The script must be saved under the same name as YourModuleName. For example, a module name of Template must be saved under Template.py.
2. Replace **YourModuleNameWidget** with “Widget” appended to the name of the module. This must not include any spaces. For example, a module name of Template must have this field as TemplateWidget.
3. Replace all marked fields in the **YourModuleName class**.

* parent.title = Title of module that users will see in the module list
* parent.category = Category that this module will fit into in the module list
  + See module list for examples (Utilities, Informatics etc.)
* parent.contributors = List of people who contributed to this module
* parent.helpText = Description of module that users will see
* parent.acknowledgementText = List of acknowledgements (i.e. Sponsors)

1. Do **not** touch the code in **\_\_init\_\_** of **YourModuleNameWidget**
2. Under **def setup(self)** of YourModuleNameWidget, determine number of required buttons and add/remove as necessary. The template only includes generic push buttons (qt.QPushButton). These push buttons are shown in the figure below.



Buttons are stored as variables using the following format.

**variableName = qt.QPushButton(“Text to display on button”)**

The apply button above will be: **applyButton = qt.QPushButton(“Apply”)**

Users can add other buttons as required to the Widget code. The best way to do so is to find an existing module that includes the required Widget part and access the source code for the module at <https://github.com/Slicer/Slicer/tree/master/Modules>. The code for the required Widget part can then be copied.

1. The **toolTips** for each button is optional. These tooltips show the entered text if users mouse-over the button. This can be used to tell users what each button does.
2. Once all the buttons are defined, add each one to the “layout” through **self.layout.addWidget(VariableNameOfButton)**.

Thus, to add a button that is stored under the variable A: **self.layout.addWidget(A)**

1. Tell Slicer what class to execute when a button is clicked. The format for this is **buttonName.connect(‘clicked(bool)’,self.classToRunWhenClicked)**. Modify buttonName and classToRunWhenClicked as necessary to point to the right class.

For example, the template has **buttonOne.connect(...self.buttonOneClicked)**. This runs the buttonOneClicked class when buttonOne is clicked by the user.

1. Leave the **addStretch** function alone. This will help to space the buttons out.
2. Add the code that should be executed when a button is clicked.

**Note: Developers can point to another class within these functions. This is the best way to integrate this template to already written codes.**

Assume that the module is already written under the class ModuleCode. If this code is to be run when buttonOne is clicked, the following code can be used

def buttonOneClicked(self):

ModuleCode() 🡨 Note that the ()s are necessary even though it is a class

**-- End of Appendix 3 --**

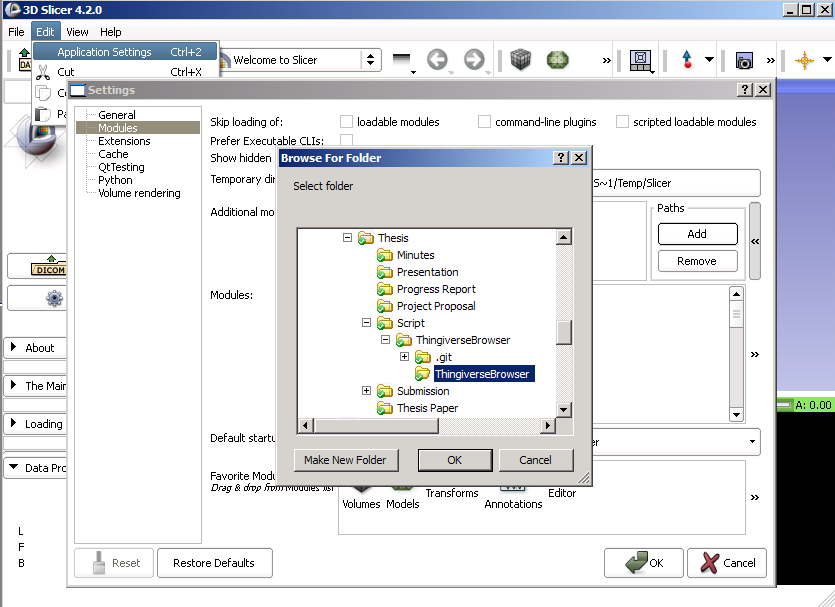
# Appendix 4 – Integrating Module with Slicer

**Step 1 – Save module into folder**

1. Save module(s) into a folder. Do not have any other folders within this folder.

**Step 2 – Integrate folder with Slicer**

1. Start the Slicer application.
2. In the menu bar at the top of Slicer, click **Edit > Application Settings.**
3. Inside Applications Settings, click on the **Modules** tab in the menu on the left.
4. Click on the **double arrows** to the right of **Additional module paths**. This will open a new tab called **Paths**.
5. In the new Paths menu, click on **Add**. This will open a Browse for Folder menu.
6. In the Browse menu, navigate to and select the **folder** where the module is saved and click **OK**



**3**

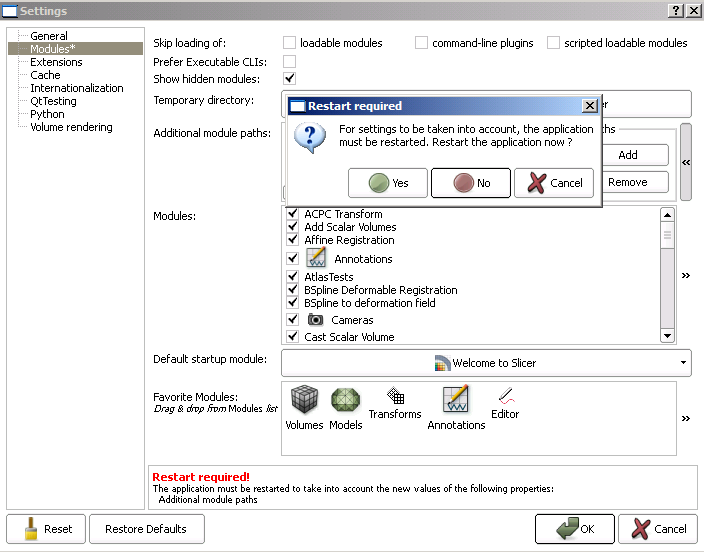
**5**

**4**

**2**

**1**

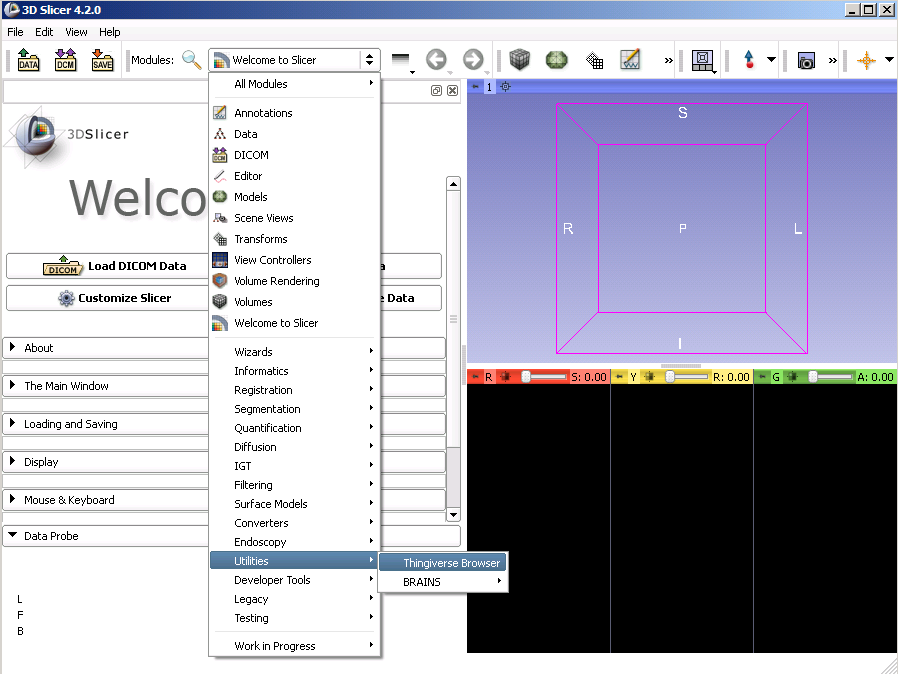
1. Click on **OK** at the bottom right of Settings.
2. Slicer will then prompt the user to restart. Click on **Yes**.



**1**

**2**

1. Once Slicer has restarted, the module will be available in the **Modules** menu.



**-- End of Appendix 4 --**

# Appendix 5 – Extension Documentation

**Step 1 – Upload Icon to SlicerWiki**

1. Create an icon to represent the extension. This icon must be:

* 128x128 pixels
* Saved in .PNG format

1. Go to Slicer Wiki (<http://slicer.org/slicerWiki/index.php/>) and sign in.
2. In the left menu, under toolbox, click on **Upload file.**
3. Use **Choose File** next to Source filename to select the icon.
4. Destination filename will be automatically filled. Input a brief summary for the icon being uploaded.
5. Click on **Upload File.**



**3**

**2**

**1**

**Step 2 – Create the Module Page on the Slicer Wiki**

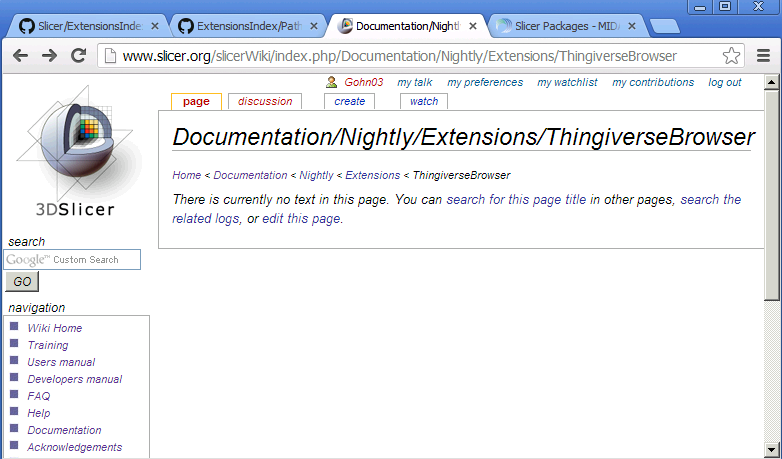
**Note**: Steps 2, 3 and 4 are based off the information provided by the Slicer team at <http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Developers/Tutorials/DocumentExtension>

1. Create a Wiki page for the module by going to this URL.

[http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Modules/**name**](http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Modules/name)

Replace the bolded **name** in the URL with the name of the module being documented.

1. Click on **edit this page**



**1**

1. Open another browser window and find the template for Wiki pages of modules at:

<http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Modules/YOURMODULENAME>

**Note:** Do not change YOURMODULENAME in the URL above. Use the URL as-is.

1. Click on **edit** at the top of the page. Copy all the text into the Wiki page being created. Modify fields as appropriate and **Save Page**.

**Step 3 – Create the Extension Page on the Wiki**

1. Create another Wiki page for the extension at

[http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Extensions/**name**](http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Extensions/name)

**Note**: Link is similar to the one above. Modules has simply been changed to Extensions.

1. Click on **edit** and type the following.

#REDIRECT [[Documentation/Nightly/Modules/Name]]

Where Name is the name of the module. For example, a module called Thingiverse will have

#REDIRECT [[Documentation/Nightly/Modules/Thingiverse]]

Note that there is a space in between #REDIRECT and [[. This will redirect users from the extension page to the module page that was created earlier.

**Step 4 – Add Extension Entry to Extension List**

1. Go to Slicer’s list of extensions at

<http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Extensions#Extension_Categories>

1. Add extension Wiki page to either Category 1, Category 2 or Category 3.

* Category 1: Maintained extensions with Slicer license
* Category 2: Normal, open-source extension
* Category 3: All other extensions (Work in progress, closed course etc)

The average extension will fall under Category 2. To input the extension Wiki page, copy the format from other entries and replace the extension name as appropriate.

**Note:** The \* at the start of each line represents indentation. Indentation is used to show modules contained within modules. If unsure, use one \*.

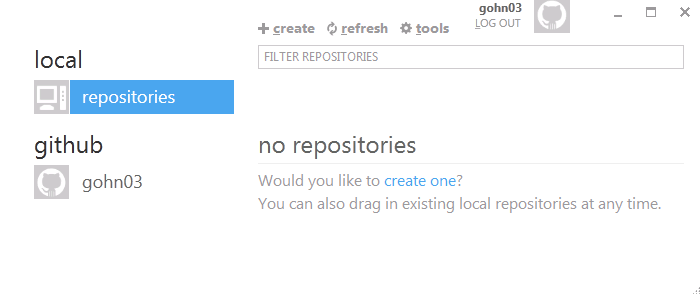
**-- End of Appendix 5 --**

# Appendix 6 – Extension Setup

**Note**: Ensure that all steps in **Appendix 2** have been completed

**Step 1 – Set-up GitHub**

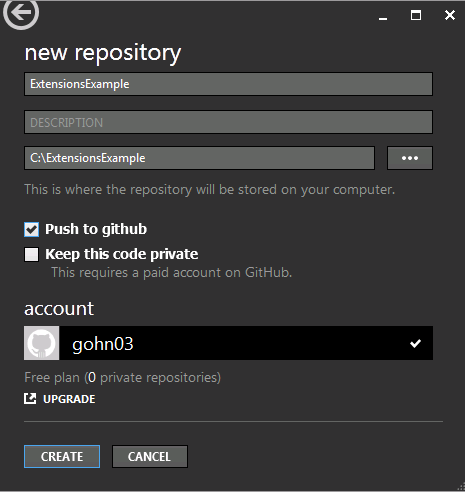
1. Run the GitHub for Windows client. **Start > Programs > GitHub, Inc > GitHub.**
2. In the menu on the left, click on **repositories.**
3. Click on **create**.



**2**

**1**

1. Under **NAME**, input the name of the Extension. Change the directory if desired.
2. Ensure that **Push to github** is checked and click **CREATE** to create a repository folder.



**1**

**3**

**2**

**Step 2 – Set-up Extension Folders**

1. Outside of GitHub, navigate to the repository folder that was just created. For this example, assume that this folder is called ExtensionName.

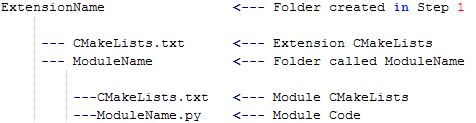
**Note**: If unsure of how to find the newly created repository folder, follow the steps below:

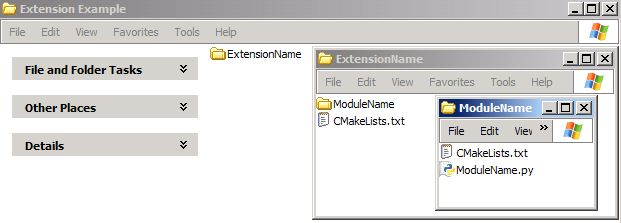
* Open GitHub for Windows.
* Click on **tools** at the top of the client.
* In the dropdown menu, click on **options.**
* The GitHub default directory is shown on the right under **default storage directory.**
* The created repository folder will be inside the default storage directory.

1. In the ExtensionName folder, create another folder with the name of the module. For this example, assume that this folder is called ModuleName.

**Note:** It is possible for the folders **ExtensionName** and **ModuleName** to have the same name.

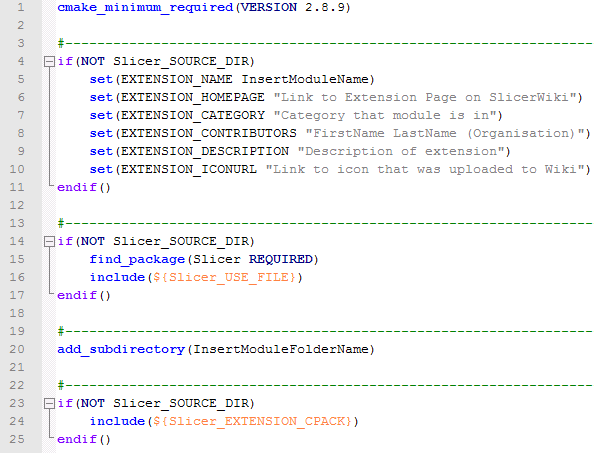
1. Also in the ExtensionName folder, create a **CMakeLists.txt** file. Leave this file empty for now.
2. Go into the **ModuleName** folder and create another empty **CMakeLists.txt** file.
3. Move the module code (ModuleName.py) into the **ModuleName** folder.



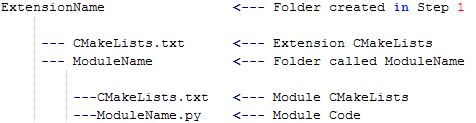


**-- End of Appendix 6 --**

# Appendix 7 – Extension CMakeLists Template



1. Open up CMakeLists.txt in the extension folder



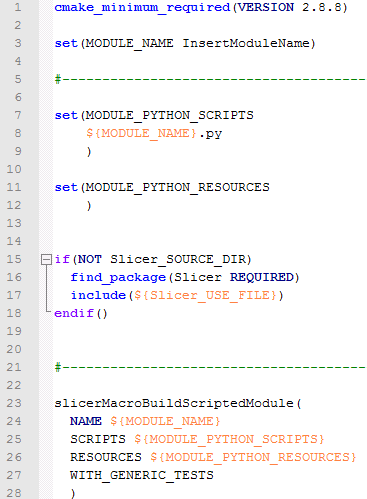
1. Copy and paste the template into the CMakeLists.txt file. Note that only lines 4-10 and 20 will have to be modified. Everything else should not be touched.
2. **EXTENSION\_NAME (Line 5):** **Replace InsertModuleName** with the name of the module (do **not** include the extension .py). Note that this field does **not** use quotation marks unlike the others.
3. **EXTENSION\_HOMEPAGE (Line 6):** Input the full link to the Extension Wiki page that was created in Appendix 5 **within the quotation marks**.
4. **EXTENSION\_CATEGORY (Line 7):** Input the module category **within the quotation marks**. If unsure, check the module script. This should have been included as part of the module code under **parent.categories = [“ ”]**.
5. **EXTENSION\_CONTRIBUTOR (Line 8):** Input name of all contributors to code **within the quotation marks.**
6. **EXTENSION\_DESCRIPTION (Line 9)**: Input description of extension **within the quotation marks**.
7. **EXTENSION\_ICONURL (Line 10)**: Input direct link to icon that is used to represent the extension **within the quotation marks**.

To find the icon URL on the SlicerWiki, use the following steps:

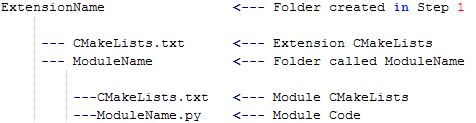
1. Go to the Slicer Wikipage and log-in
2. Click on **my contributions** at the top right
3. Find **File: IconName.PNG** and click on it
4. **Click on the displayed image** of the icon
5. Copy the URL
6. **InsertModuleFolderName (Line 20):** Replace **InsertModuleFolderName** with the name of the folder that is contained inside the extensions folder. Using the image above as an example, this would be the folder **ModuleName**.

**-- End of Appendix 7 --**

# Appendix 8 – Module CMakeLists Template



1. Open up CMakeLists.txt in the module folder

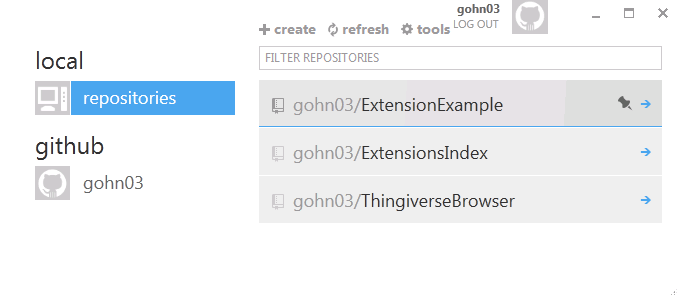


1. Copy and paste the template into the CMakeLists.txt files listed above. Only Line 3 will have to be modified.
2. **MODULE\_NAME (Line 3):** Replace **InsertModuleName** with the name of the module (do **not** include the .py extension)

**--End of Appendix 8--**

# Appendix 9 – Committing back into GitHub

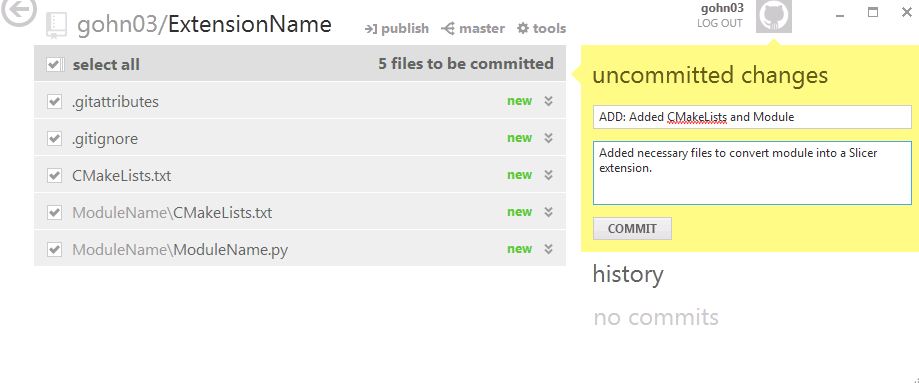
1. Open up GitHub for Windows. **Start > Programs > GitHub, Inc > GitHub**.
2. On the left, find and click **repository** under Local.
3. Locate the folder that contains the extension and click on the **blue arrow** next to it.



**2**

**1**

1. Fill in **COMMIT MESSAGE**. Try to detail what is being changed.
2. Fill in **EXTENDED DESCRIPTION**. This is optional but recommended.
3. Click on **COMMIT.**
4. Click on **publish** at the top of GitHub.



**4**

**3**

**2**

**1**

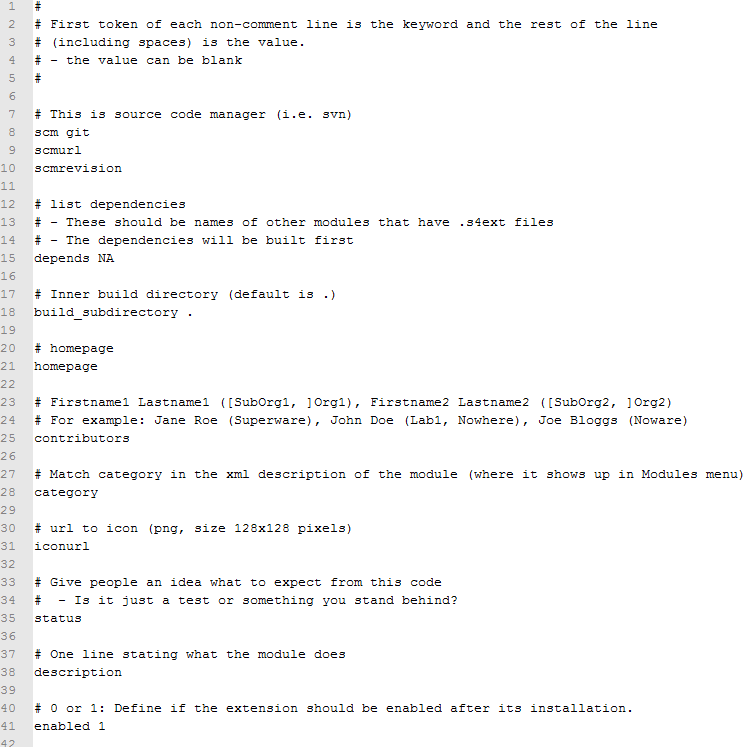
**-- End of Appendix 9 --**

# Appendix 10 – .s4ext Template

**Note:** If manually typing the s4ext file, all comment lines (preceded with #) can be skipped.

**Important Note**: The s4ext file makes use of keywords at the start of each line (i.e. scmurl for line 9 in the example below). To assign values to these keywords, simply add a space after each keyword and type the necessary text.

For example, in line 8, the keyword is scm. The “value” assigned to this is **git**. The rest of this appendix goes through each line of the s4ext template that has to be modified and provides a guide on how to modify the file.

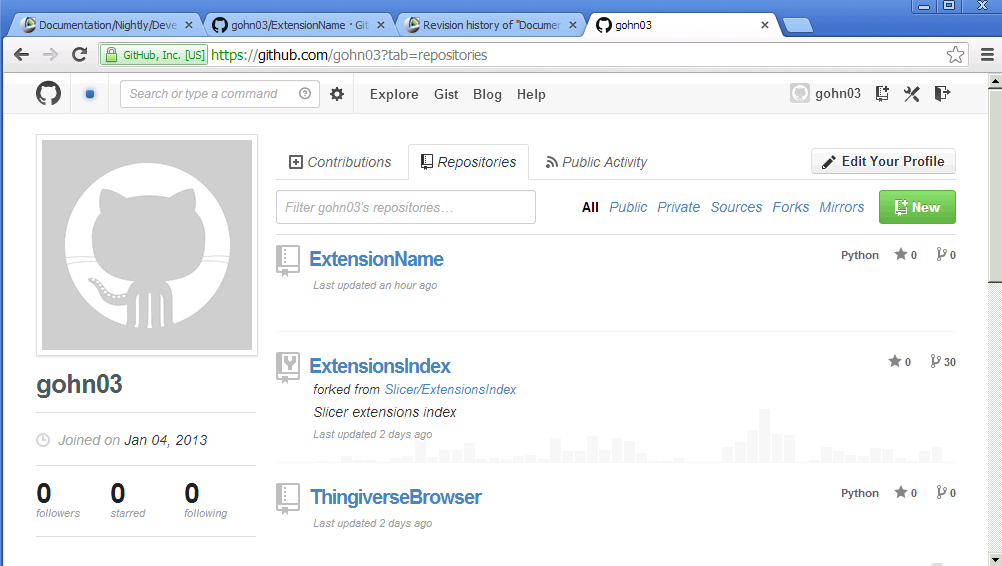


1. Open up a text file. Save it as **ModuleName.s4ext**. (Note that .s4ext is the extension. Do not save it as ModuleName.s4ext.txt)

**Note**: This s4ext file can be saved anywhere. It does not have to be in the same folder as the extension. It is recommended that this file is saved in an easy to access directory.

1. **scm (Line 8)**: Input the version control system used. The template has this as **git** as that is the version control system used for this guide. Change only if svn is used.
2. **scmurl (Line 9)**: Input the github address of extension

* Go to <http://github.com>.
* Login and click on the **username** at the top right.
* Click on the **Repositories** tab.
* Click on the repository that contains the extension.

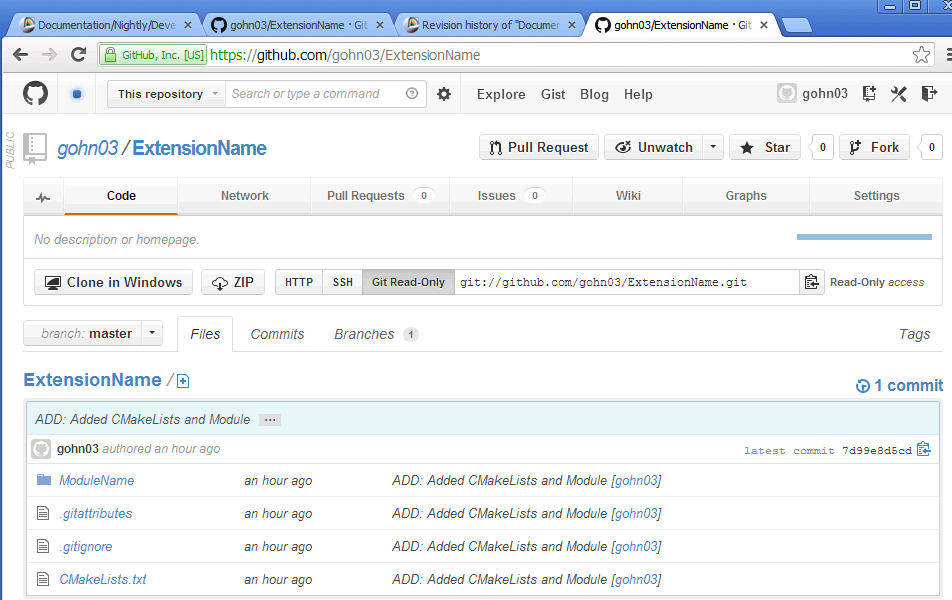


**3**

**2**

**1**

* Click on **Git Read-Only.**
* Click on the **Copy to Clipboard button.**
* Paste the copied text behind scmurl in the s4ext file.



**2**

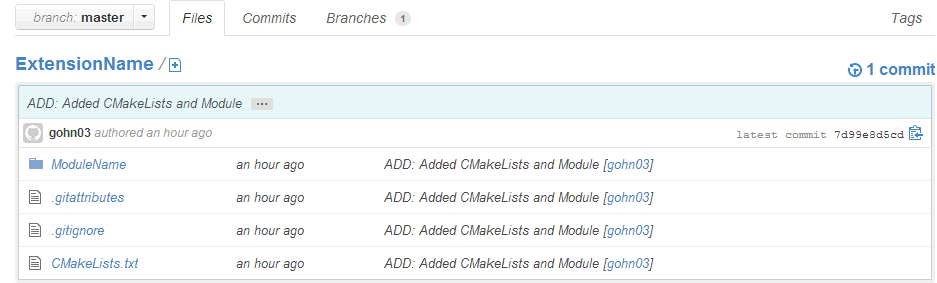
**1**

Thus, for the example in the figure above, Line 9 of the s4ext file will read

scmurl git://github.com/gohn03/ExtensionName

1. **scmrevision (Line 10):** Input the latest commit code

* Go to the extension repository on Github (same page from Step 2).
* Click on **Copy to Clipboard** next to latest commit.
* Paste this behind scmrevision in the s4ext file (Note the actual revision code is longer than what is shown).



**1**

For this example, Line 10 of the s4ext file will read

scmrevision 7d99e8d5cdb79a074f54b32126b0085c699308fb

1. **depends (Line 15):** By default, this is **NA** and the template already has this filled in. Only change this value if the extension requires other modules/extensions to run.
2. **build\_subdirectory (Line 18):** By default, this is **.** and the template has this already filled in. Only change if necessary. In most instances, **.** is sufficient.
3. **homepage (Line 21):** Input the link to the extension page on the SlicerWiki.

homepage http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Extensions/...

1. **contributors (Line 25)**: Input all contributors to the code in the format of

**First Last (Organisation), First Last (Organisation)**

A code written by Nigel Goh from UWA will have the following

Contributors Nigel Goh (UWA)

1. **category (Line 28)**: Input the module’s category. This should have been listed in the Slicer integration portion of the module script under **parent.categories = [“ ”]**.

A module that of category “Utilities” would have the following line in the s4ext file

category Utilities

1. **iconurl (Line 31)**: Input the URL to the 128x128 icon that represents the extension.
   * Go to the Slicer Wikipage and log-in.
   * Click on **my contributions** at the top right.
   * Find **File: IconName.PNG** and click on it.
   * **Click on the displayed image** of the icon.
   * Copy the URL and paste behind iconurl.

iconurl http://www.slicer.org/slicerWiki/images/...

1. **status (Line 35)**: Input the status of the extension. Examples include
   * Alpha
   * Beta
   * Work in Progress
   * Final

status Beta

1. **description (Line 38)**: Input a one line description of the extension.

description This is an example of the description line for the s4ext file

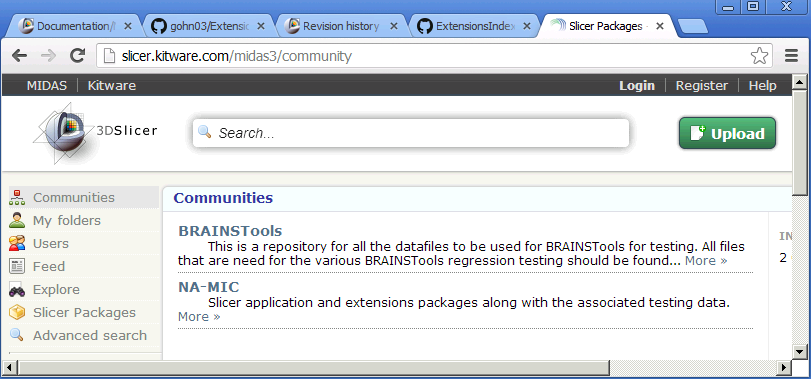
1. **enabled (Line 41):** The template has this set at the default value of **1**. This line defines whether or not the extension should be enabled once it has been installed via the Extensions Manager. 1 for enabled, 0 for disabled. Only change to 0 if the extension should be manually enabled after installation.
2. **Empty line (Line 42):** Do not end the code on line 41. Slicer requires one empty line at the end of the s4ext file to execute the file. Thus, ensure line 42 exists and that it is an empty newline.
3. **Save and close** the s4ext file.

**-- End of Appendix 10 --**

# Appendix 11 – Building, Testing and Uploading Extension

**Step 1 – Obtain an API Key**

1. Go to <http://slicer.kitware.com/midas3/>.
2. Click on **Register** at the top right and create an account.
3. In the toolbar at the left side of the screen, click **Communities.**
4. Under communities, click on **NA-MIC.**
5. Click on **Join the Community** (Not shown in visual representation below).

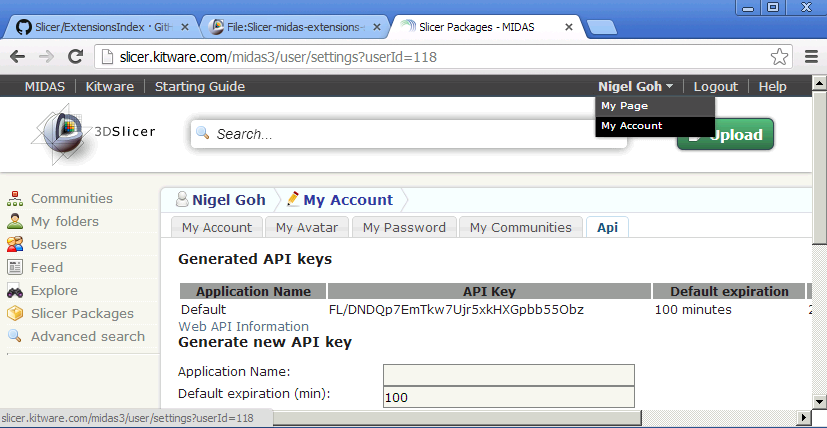


**3**

**2**

**1**

1. Click on **Username** at the top right.
2. Click on **My Account** from the dropdown menu.
3. Click on the **API tab**.
4. Take note of the **Default API**. Note that all special symbols, such as /, are part of the API key.



**2**

**1**

**4**

**3**

**Step 2 – Build and Test Slicer Extension**

1. Open windows command prompt. **Start > Run > cmd > OK**

For the next few steps, assume the following:

**C:\Slicer** = Directory of the Slicer source code cloned from Git

**C:\Slicer4-SuperBuild-Release** = Directory where Slicer was built

**C:\Extension** = Folder in which the extension’s s4ext file is kept

If you are unsure as to where the Slicer source code was cloned or where Slicer was built, look at Steps 9 and 10 of Appendix 2 for a refresher.

1. In command prompt, navigate to the parent directory of where the s4ext file is kept (i.e. C:\Extension has a parent directory of C:\)

cd C:/

1. Create a new folder called **Extension-build**. Replace Extension with the name of folder where the s4ext file is kept. This is done using **mkdir** (Make directory).
2. Navigate to the new **Extension-build** folder using cd.

mkdir Extension-build

cd Extension-build

1. Run the following code. Note that ^ is used in windows to carry the code over into a new line. Press **Enter** when complete. This will build Slicer in the current directory (The newly made Extension–build directory).

cmake -DSlicer\_DIR:PATH=**C:\Slicer4-SuperBuild-Release**\Slicer-build ^

-DSlicer\_EXTENSION\_DESCRIPTION\_DIR:PATH=**C:\Extension** ^

-DCMAKE\_BUILD\_TYPE:STRING=Release ^

-DSlicer\_UPLOAD\_EXTENSIONS:BOOL=ON ^

-DMIDAS\_PACKAGE\_URL:STRING=http://slicer.kitware.com/midas3 ^

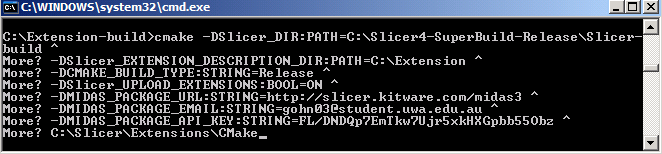
-DMIDAS\_PACKAGE\_EMAIL:STRING=**your@email.here** ^

-DMIDAS\_PACKAGE\_API\_KEY:STRING=**YourAPIKey** ^

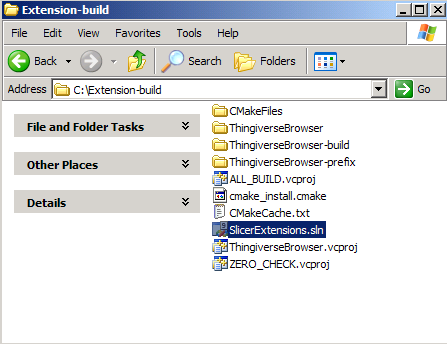
-**C:\Slicer**\Extensions\CMake

Replace the underlined and bolded text with the relevant directories. Note that for some lines, there are directories that are not completely bolded and underlined. This means that only part of the directory should be changed.

For example, in the final line of the code: **C:\Slicer**\Extensions\CMake is used where **C:\Slicer** is the location of the Slicer source code. If the Slicer source code was saved at **C:\My Documents\Slicer Code**, then the final line would read **C:\My Documents\Slicer Code\Extensions\CMake**

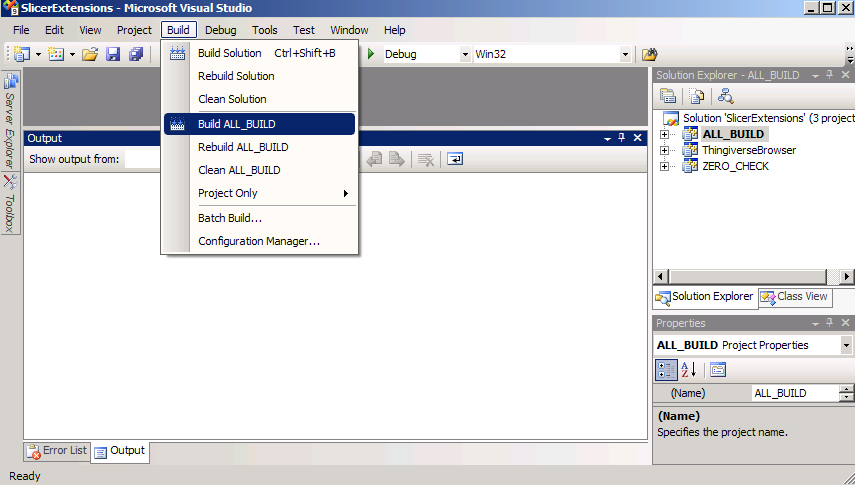


1. When this process is complete, command prompt will say **“Build files have been written to: C:\Extension-build”.**
2. Go to the build directory (C:\Extension-build for this example) and run **SlicerExtensions.sln.**



1. This will open Visual Studio. On the right side in Solution Explorer, click on **ALL\_BUILD**
2. In the menu bar on top, click on **Build**. Select **Build ALL\_BUILD.**

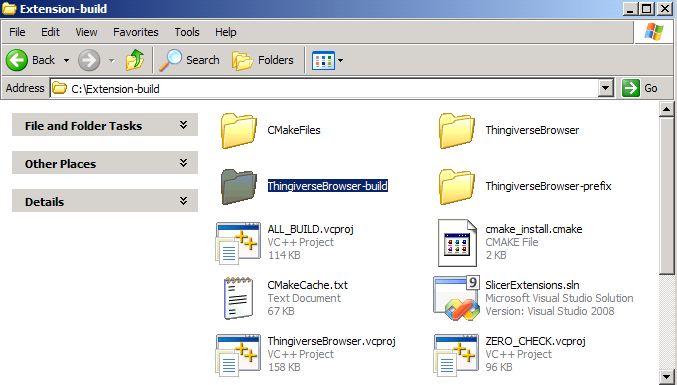
**2**



**3**

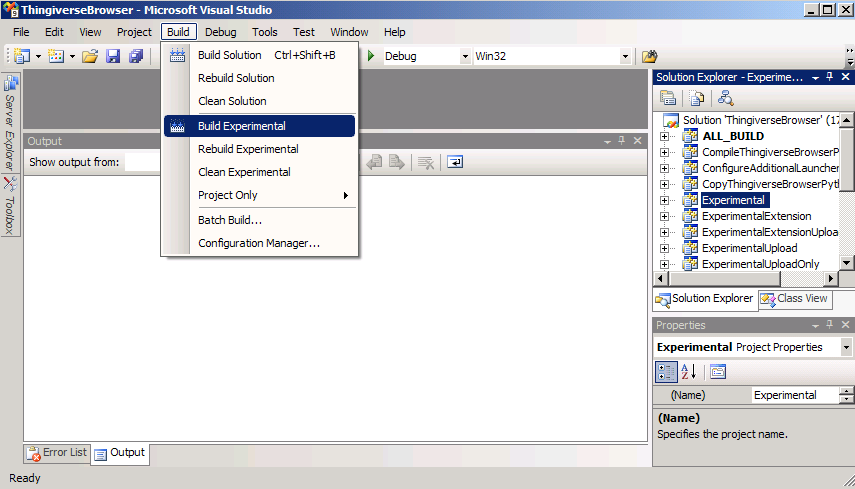
**1**

1. This will download the files from the scmurl listed in the s4ext. Wait until Visual Studio finishes the build process. This will be signified by **Build succeeded** being displayed at the bottom left of Visual Studio.
2. Return to the Extension-build folder. Go into the <**ModuleName>**-**build** folder. Where <ModuleName> is the name of the module being built.



1. Access **<ModuleName>.sln**.
2. In Solution Explorer on the right, click on **Experimental**.
3. In the menu bar on top, click on **Build** then click on **Build Experimental**.

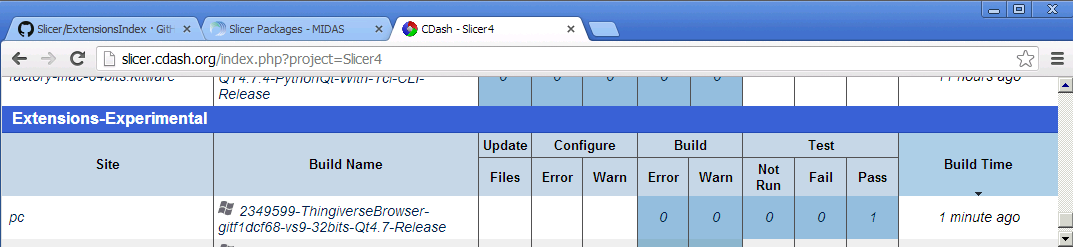
**2**



**1**

**3**

1. Wait until Visual Studio says **Build Succeeded** at the bottom left of the screen.
2. Go to <http://slicer.cdash.org/index.php?project=Slicer4#Extensions-Experimental>
3. Ensure that the build process passed all the relevant tests. There should be 0 under the Fail column. If there are multiple experimental extensions, the module can be identified via the **Build Name** column.

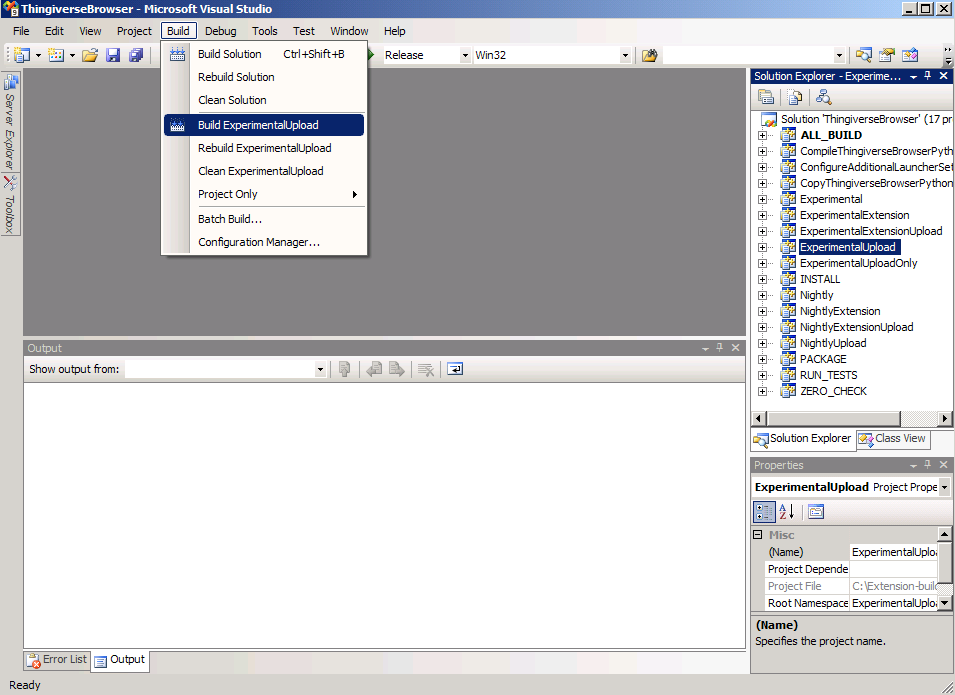


**Step 3 – Upload Extension**

1. Return to **<ModuleName>.sln**.
2. Select **ExperimentalUpload** from Solution Explorer.
3. Click on **Build** at the menu bar and select **Build ExperimentalUpload**.

.

**2**



**1**

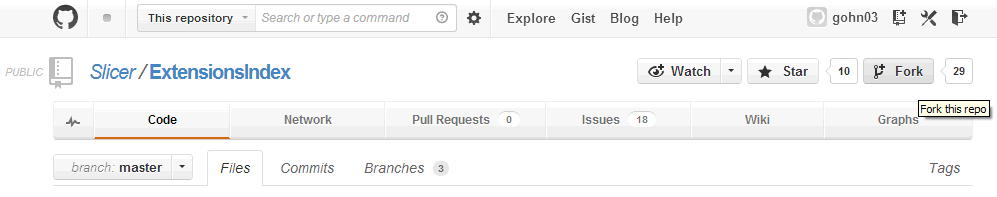
**3**

**--End of Appendix 11--**

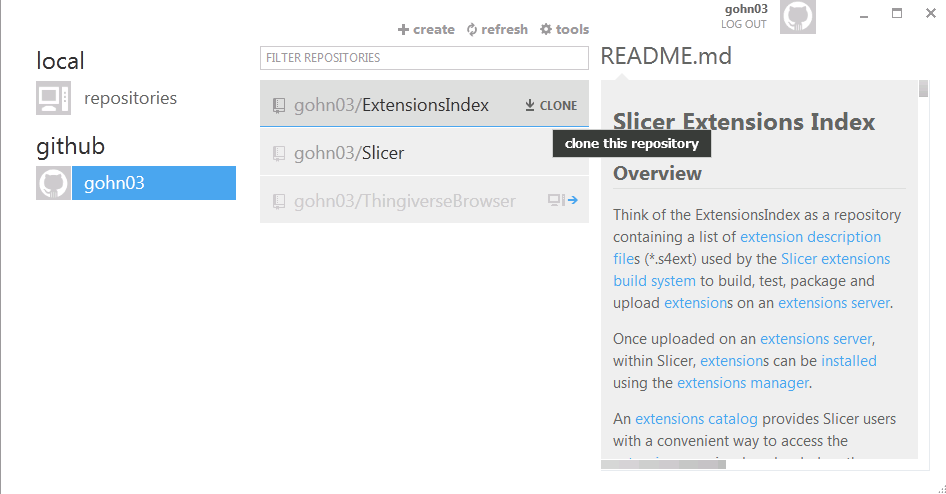
# Appendix 12 – Upload Extension Description to ExtensionIndex

**Step 1 – Fork and Clone the ExtensionsIndex Repository**

1. Go to <https://github.com/Slicer/ExtensionsIndex>.
2. Click on **Fork** at the top right of the screen.



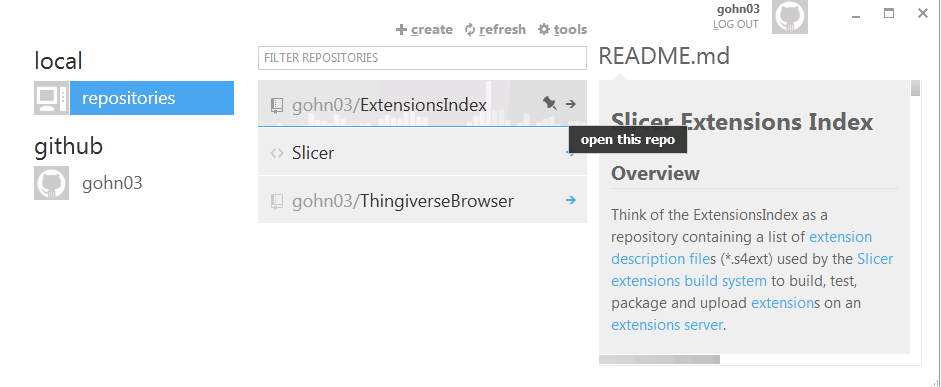
1. Open up GitHub for Windows. **Start > Programs > GitHub, Inc > GitHub.**
2. On the left side of the client, click on your **username** under **github**.
3. Select the ExtensionsIndex repository and click on **CLONE**.



**2**

**1**

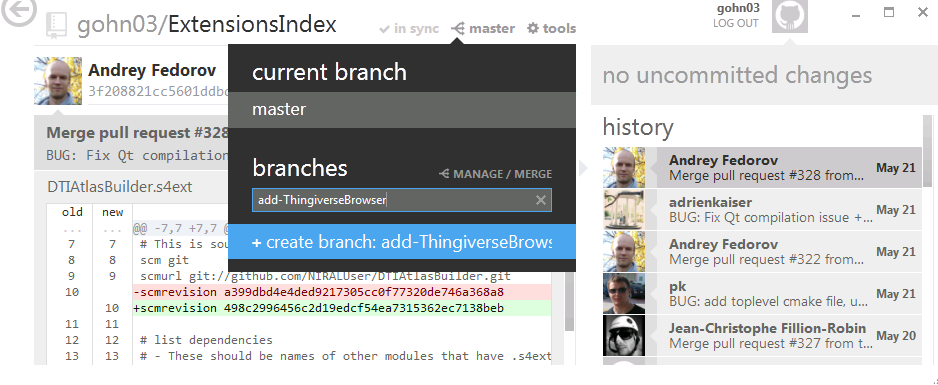
1. On the left side of the screen, click on **repositories** under local.
2. Find the **ExtensionsIndex** repository and click on the arrow to **open the repo**.



**2**

**1**

1. Click on **master** at the top of the screen.
2. A pop-up menu will appear. Under **branches**, type “add-ExtensionName”.
3. Click on **create branch: add-ExtensionName.**

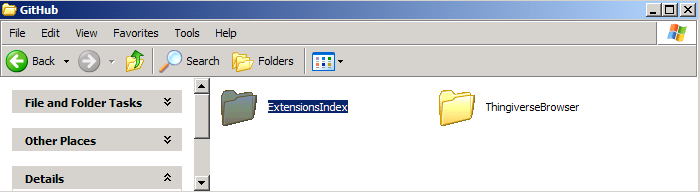


**Step 2 – Integrate extension’s s4ext file with ExtensionsIndex**

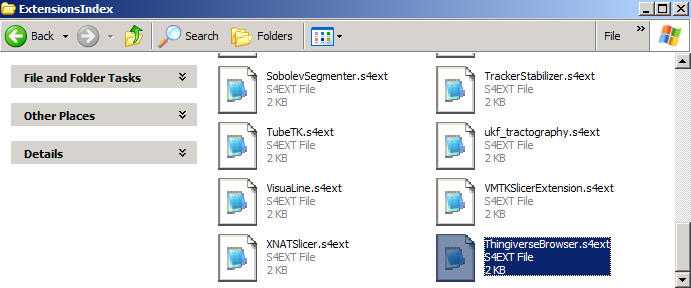
1. Go to the folder where ExtensionsIndex was cloned to. This will be in GitHub’s default folder. If unsure of how to find this folder, follow the steps below:

* Open GitHub for Windows
* Click on **tools** at the top of the client
* In the dropdown menu, click on **options**
* The GitHub default directory is shown on the right under **default storage directory**

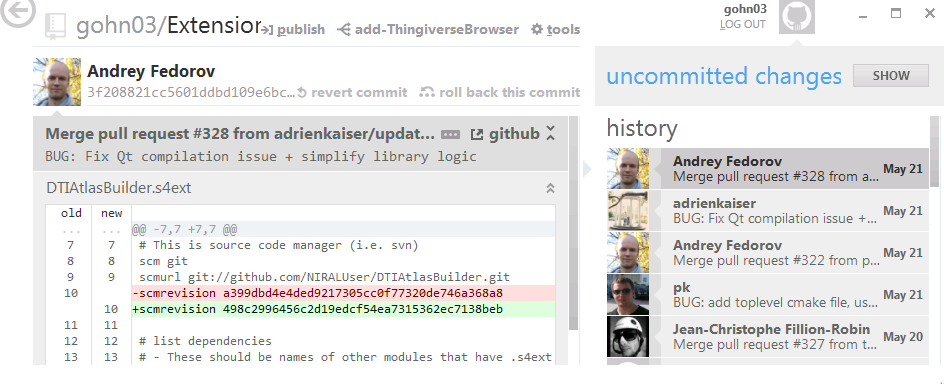
1. Open the **ExtensionsIndex** folder.



1. Copy and paste the extension’s s4ext file into the folder.



1. Return to the GitHub for Windows client.
2. Click on **repositories** under **local** and open the **ExtensionsIndex** repo.
3. GitHub will say that changes have been made. Click on **SHOW**.



**1**

1. Under **COMMIT MESSAGE**, add in **“Add ExtensionName extension”**. This format is requested by the Slicer development team.

Replace ExtensionName with the name of the extension but do not use spaces. Capitalise in between words if necessary.

1. Under **EXTENDED DESCRIPTION**, write a brief description of the extension.
2. Click on **COMMIT**.
3. Once GitHub alerts the user to unsynced commits, click on **publish**.



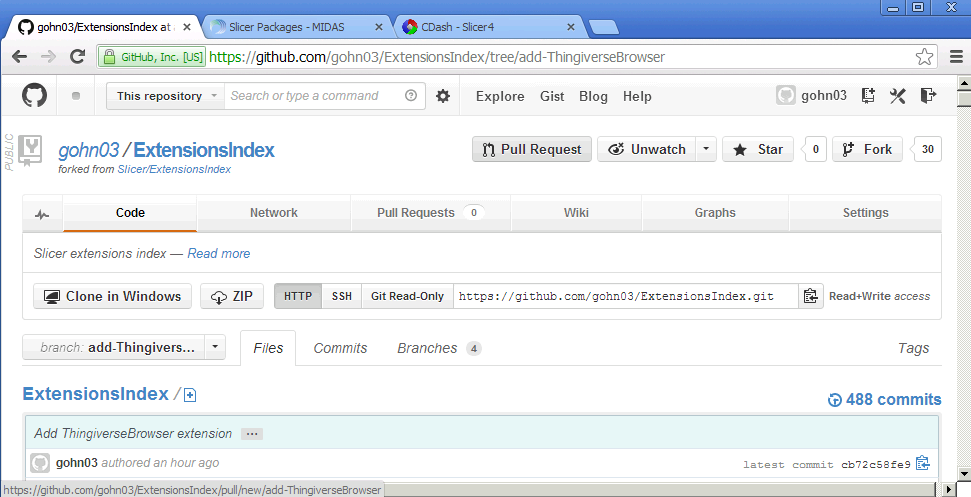
**4**

**1**

**3**

**2**

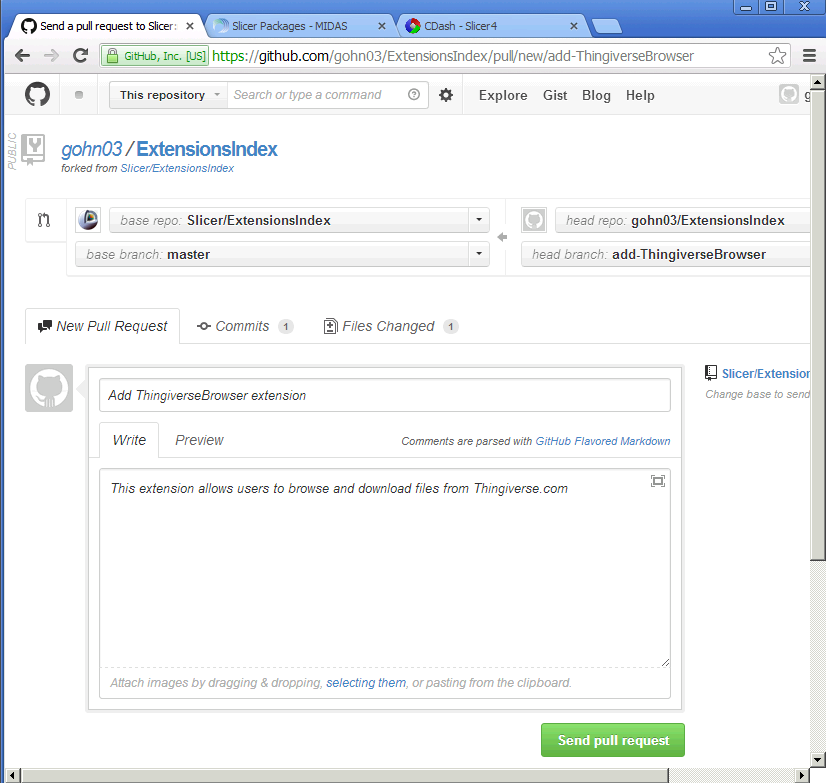
1. When publish has been replaced by **In Sync**, the process is complete.
2. Return to the ExtensionsIndex site (<https://github.com/Slicer/ExtensionsIndex>).
3. Click on **branch** and select the newly added **add-ExtensionName** branch.
4. Click on **Pull Request.**



**2**

**1**

1. In the new screen, check that the **Commit Message** and **Extended Description** are correct. Pay close attention to **Commit Messag**e. It must follow the format **Add ModuleName Extension**.
2. Click on **Send Pull Request.**



**1**

**-- End of Appendix 12 --**

# Appendix 13 – Python Module Development Overview

