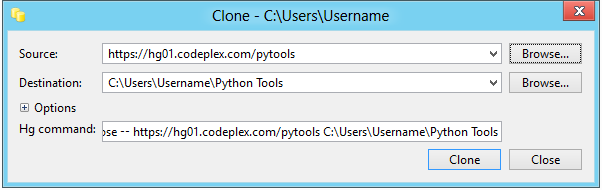
To build PTVS on your own PC, for example to pick up the latest bug fixes, make modifications, or contribute back to the project, please use the following instructions.

# Cloning

After installing the prerequisites, you will require a local clone of our Mercurial repository. The source URL is <https://hg01.codeplex.com/pytools>. If you’re behind a corporate firewall you may need to enter your proxy settings. These can be found in TortoiseHg’s Global Settings dialog.



If you do not want to use Mercurial, you can download the latest commit as a ZIP file from the [Source Code](http://pytools.codeplex.com/SourceControl/list/changesets) page.

If you intend to contribute back to PTVS, you will need to [Create a Fork](http://pytools.codeplex.com/SourceControl/network/create/fork) and use the source URL provided there.

# Prerequisites

The following list of software is required in order to build PTVS from source. At present, all of these are required.

|  |  |
| --- | --- |
| **Software** | **Download** |
| Microsoft Visual Studio 2010 Premium or Ultimate | Free trials: <http://www.microsoft.com/visualstudio/en-us/try> |
| VS 2010 Service Pack 1 | <http://www.microsoft.com/en-us/download/details.aspx?id=23691> |
| Visual Studio 2010 SP1 SDK | <http://www.microsoft.com/en-us/download/details.aspx?id=21835> |
| TortoiseHg | <http://tortoisehg.bitbucket.org/download/index.html> (Mercurial client) |
| Microsoft .NET Framework 3.5 SP1 | <http://www.microsoft.com/en-us/download/details.aspx?id=22> |
| Microsoft HPC Pack 2008 R2 Client Utilities | <http://www.microsoft.com/en-us/download/details.aspx?id=17017> |
| Microsoft HPC SDK | <http://www.microsoft.com/en-us/download/details.aspx?id=12218> |
| Microsoft Kinect SDK | <http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx> |
| Windows Azure SDK for Python | <https://www.windowsazure.com/en-us/develop/python/> |

Visual Studio 2010 Premium or Ultimate are required because our profiling feature depends on some header files that are only included with those editions.

Also, be aware that the Windows Azure SDK for Python installer will install the latest release of PTVS. A system-wide PTVS installation may interfere with the build process, so you should remove it from the list of packages to install (or uninstall it later).

Some other prerequisites are included with the source code.

Our automated tests use a separately installed tool for controlling Visual Studio. These can be installed by running **Setup\TCTestHostAdapters.msi**.

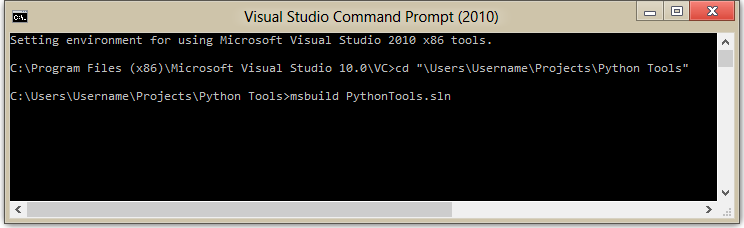
Finally you'll need to disable strong name verification for the Python Tools binaries. This can be done by merging the **Setup\EnableSkipVerification.reg** file (or **Setup\EnableSkipVerificationX86.reg**, depending on your own system).  This will install registry keys which disable strong name verification for the assemblies built by PTVS.  If you also want to use the installer you may need to stop and restart the Windows Installer service (from an elevated command prompt, type **net stop "Windows Installer"** and then **net start "Windows Installer"**).

# Building with MSBuild

PTVS can be built from the Visual Studio Command Prompt by typing

**msbuild PythonTools.sln**

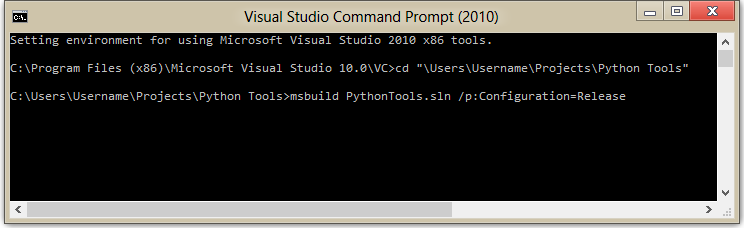
in your cloned directory. This will install PTVS into the VS Experimental hive, but does not create an installer or affect your main VS installation. Building should complete with zero warnings or errors.



To build using the Release configuration, which produces optimized code, type:

**msbuild PythonTools.sln /p:Configuration=Release**

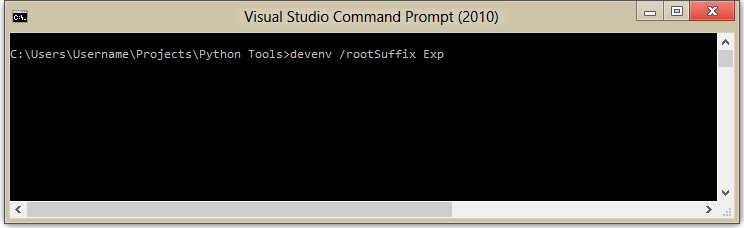
in the cloned directory. As above, this does not create an **.msi** file, but it will install PTVS into the VS Experimental hive.



The VS Experimental hive can be started from the Start menu or by typing

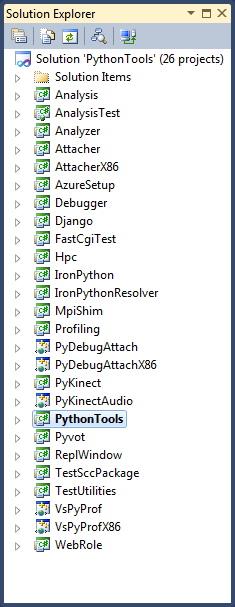
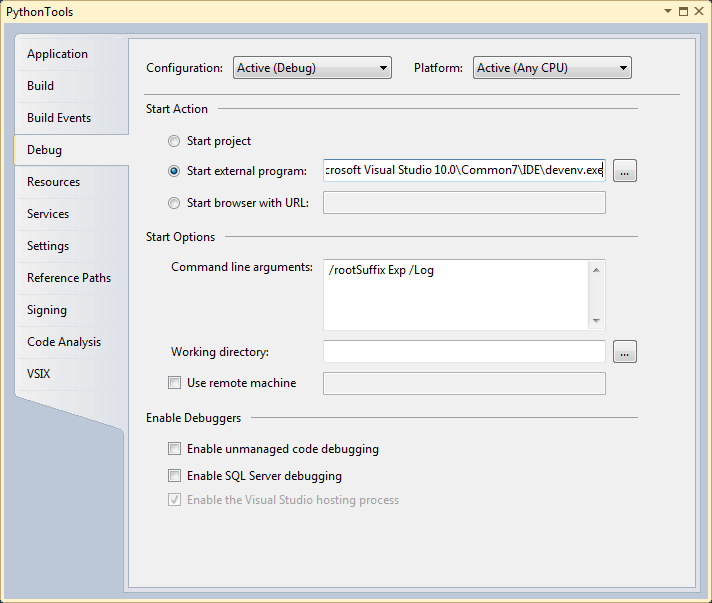
**devenv /rootSuffix Exp**

at the Visual Studio Command Prompt.



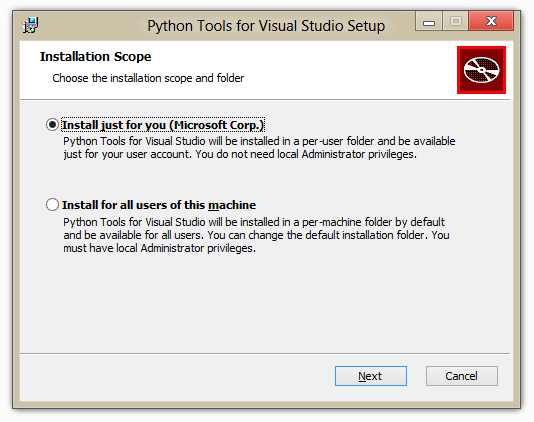
# Building with Visual Studio

**PythonTools.sln** can be opened and built in Visual Studio 2010 using the **Build Solution** command. To debug, ensure that **PythonTools** is selected as the startup project and use F5 to run PTVS inside of the VS Experimental hive. If an error appears rather than a new instance of VS, ensure the **Project|Debug** settings are correct.

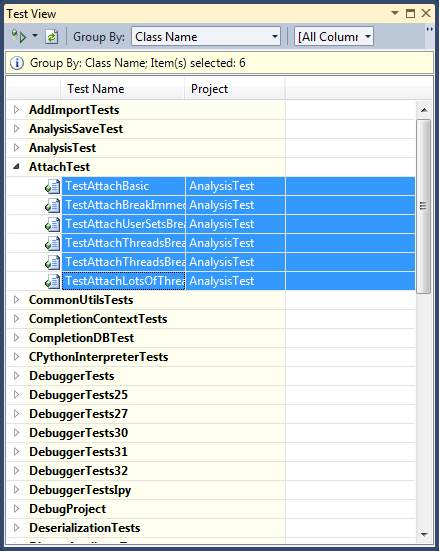
Building in Visual Studio may produce a number of warnings related to potentially incompatible assemblies and missing references. As long as all projects build successfully, these warnings are benign and can be ignored.

If you already have PTVS installed for all users then you will get an error while compiling. The solution is to uninstall PTVS, or to reinstall it for the current user only. To install for the current user, when installing choose **Advanced** and then select **Install just for you**.



# Running Automated Tests

PTVS contains a large number of automated tests, including a mix of UI-based tests (which will start another VS, take control of your mouse, etc.) and non-UI tests. Some of the UI-based tests may fail intermittently or interfere with each other, while some tests may make incorrect assumptions about your system configuration (such as the availability of certain versions of Python). It is best to select the tests you need from the **Test View** window and supervise them as they run.



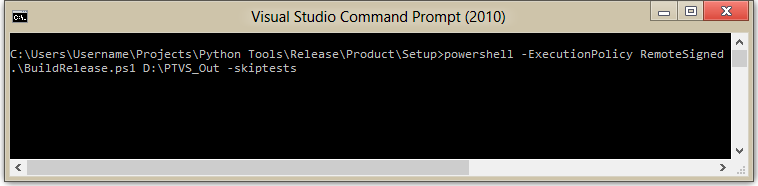
# Building for VS 2012

Details to come.

# Building the Installer

The simplest way to build the installer is to run the release script **Release\Product\Setup\BuildRelease.ps1** with PowerShell. This script updates the version to the current date/time, builds all flavors (Release, Debug and Dev11 Release/Debug releases), and archives the source code, binaries, symbols, and installers into an output directory.

**powershell -ExecutionPolicy RemoteSigned .\BuildRelease.ps1 D:\PTVS\_Out -skiptests**



The output directory will contain Debug, Release and Sources folders, and a Dev11 folder if Visual Studio 2012 is installed. The Debug and Release directories will contain the installersfor Visual Studio 2010. This is the same script which we use to builds releases of PTVS, so you'll get an installer which works just like the released versions. However, unless you sign the binaries with your own code signing certificate, the installer will only work if the **EnableSkipVerification.reg** (or **EnableSkipVerificationX86.reg**) file has been merged into the registry on the destination computer.

A quicker way to build an installer for a single configuration is to use MSBuild in the **Release\Python\Setup** directory. You'll need to specify dirs.proj as the source, as well as the configuration and version of VS targeted. For example:

**msbuild dirs.proj /p:WixVersion=1.1.0.0 /p:VSTarget=10.0 /p:Configuration=Release**

creates a Release build for Visual Studio 2010 including the installer. The resulting .msi file is in the main binaries directory: **\Binaries\Release**.

