Proteomics Research: Set of open source software libraries and applications to facilitate and accelerate the development of proteomics analysis tools.

ProteoWizard Tools and Libraries

Developer Manual

Mallick Lab
Department of Radiology
Stanford University – School of Medicine

Contents

ProteoWizard

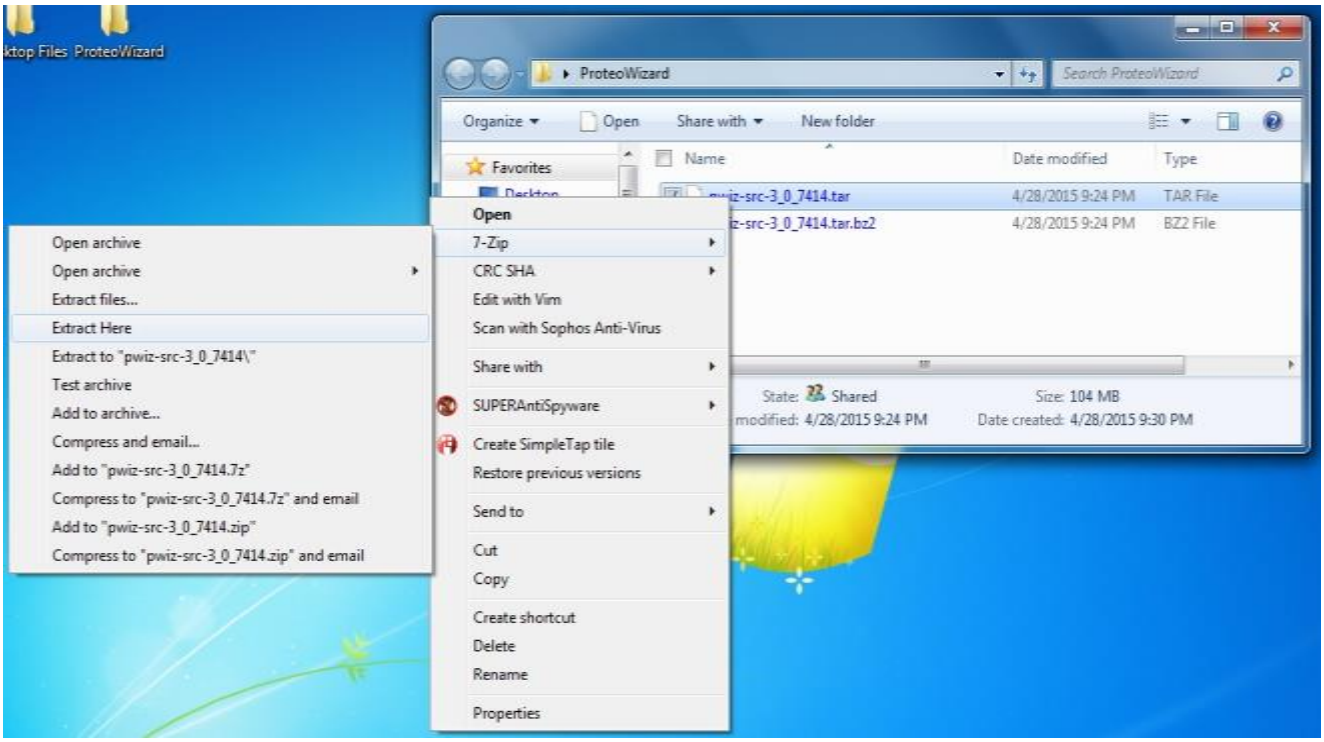
Code Download/Checkout and

Quickbuild.bat run (MS Windows)

- Before looking for a build containing the most recent c++ code, please make sure your computer system as the appropriate software to run build files or do development
- Make sure you have installed Microsoft Visual Studio 2013 (MSVS2013)
    - Use the Microsoft site, or your place of study or work, to acquire a copy
- Make sure you have installed Microsoft .NET Framework **3.5** SP1 and **4.0**
    - Both version are required for vendor DLL support
    - MS .NET Framework 3.5 SP1 can be found at: http://www.microsoft.com/en-us/download/details.aspx?id=22
    - MS .NET Framework 4 can be found at: http://www.microsoft.com/en-us/download/details.aspx?id=17851
- ProteoWizard Checkout (user level: novice)
    - Go to http://proteowizard.sourceforge.net/downloads.shtml to download the existing build under the heading For Developers: (**Source, bjam build** (includes vendor reader support)).
    - **Recommended**: you can get the build from http://sourceforge.net/p/proteowizard/code/HEAD/tree/trunk/pwiz/ where you can use an SVN Client like **TortoiseSVN**, to do an SVN Checkout or Update. Go to http://tortoisesvn.net/downloads.html . Download the version that matches your system such as 32-bit or 64-bit)
        - If checking out via SVN (**TortoiseSVN**), you can create a SourceForge account at: http://sourceforge.net/. It's not required, but you will need an account if you want to do development where you plan on checking code back in.
        - You can subscribe to proteowizard-developer@lists.sourceforge to get added to the ProteoWizard Developer list to stay informed on development updates or questions.
        - To check on possible existing issues or questions, you can also subscribe to the Support list at proteowizard-support@lists.sourceforge
        - Continuing the steps to check out code…
        - Create a 'proteowizard' somewhere (i.e. C:\project\)
        - Right on the 'proteowizard' folder and click **SVN Checkout** form the menu

- Enter https://svn.code.sf.net/p/proteowizard/code/trunk/pwiz under **URL Repository**, and click ok.
- This should take a few minutes, depending on system and internet connection to pull all the code. It is recommended to do this at least 1 time daily, where developing new code or looking at existing code.

- You will see a BZ2 file name in the folder you chose to save to with a name similar to "**pwiz-src-…_..._.....tar.bz2**"
- To extract the files for use from the download, you will need to download and install a file extraction/compression application. Go to http://www.7-zip.org/ and download 7-Zip. (you can use other tools you are comfortable with for this also)
- Once 7-Tip is installed, go to the folder with the **BZ2** file and right-click on the BZ2 file. (Windows will show it under file Type.)
- Select "Extract **Here**" and click Ok and it will create a **TAR** file similar to "**pwiz-src-…_..._.....tar"**
- Right click on the new TAR file and again select "Extract **Here"** and click Ok.

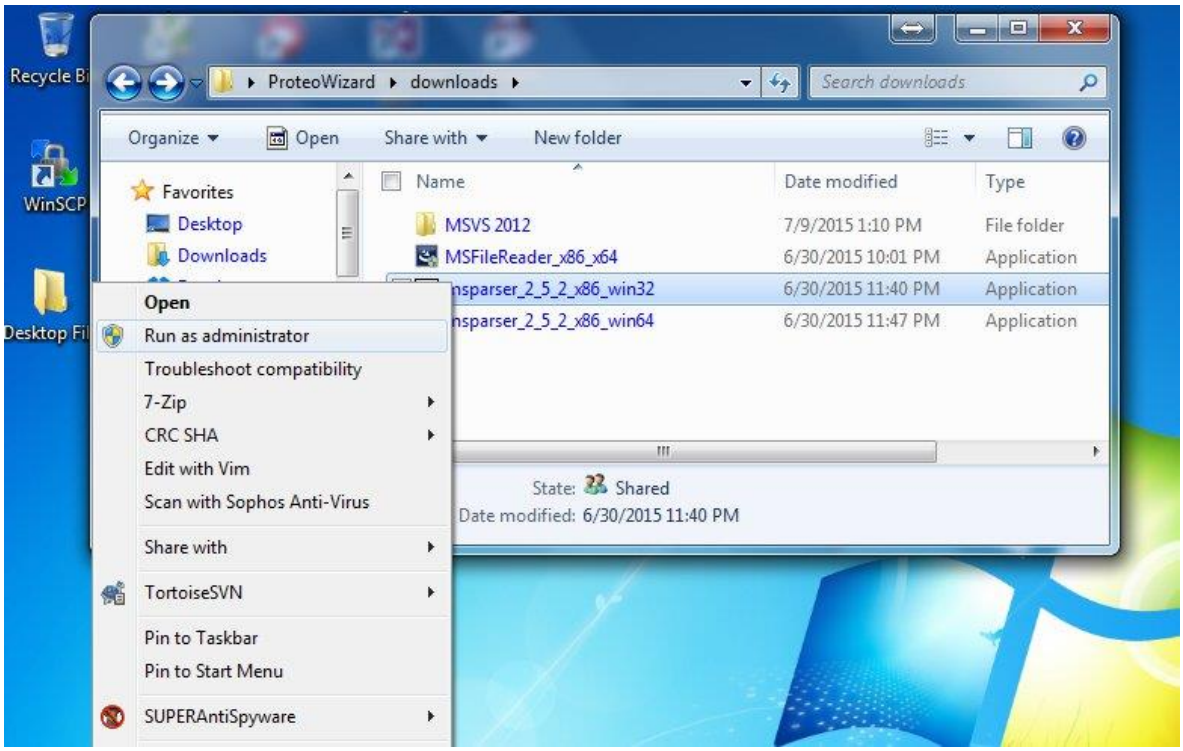


- Verify that you have the "**quickbuild.bat**" Windows batch file in the folder created so you can run it and build Proteowizard following the steps mentioned farther below.

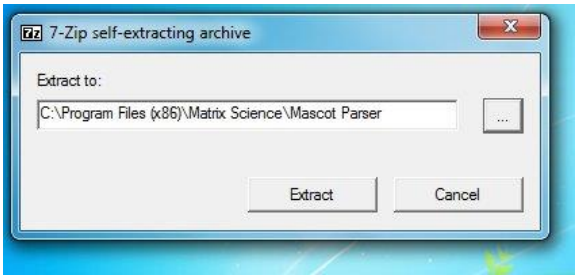
- Download **MSFileReader**
  - You will need to accept license agreements and register with Thermo. Go to https://thermo.flexnetoperations.com/control/thmo/RegisterMemberToAccount
  - You will be downloading MSFileReader 3.0
  - Register on the Thermo website
  - Then look under **Utility Software** for software named **MSFileReader 3.0 SP2** (or a newer version if it exists)
  - Download to your laptop and unzip using instructions above for unpacking tar and BZ2 files.
  - Right click on the installer file and select to "Run as Administrator" (this should avoid any install issue)
  - Accept Default settings for the installation
    - **NOTE:** Make sure to install both 32-bit and 64-bit if and when possible

- Download **Mascot Parser**
  - You will need go to the Mascot website to register with Matrix Science. Go to
  - http://www.matrixscience.com/msparser_download.html
  - After you register, they will send you an email. Select the link to download v2.5.*(or the most recent) for Windows (32-bit and 64-bit versions)

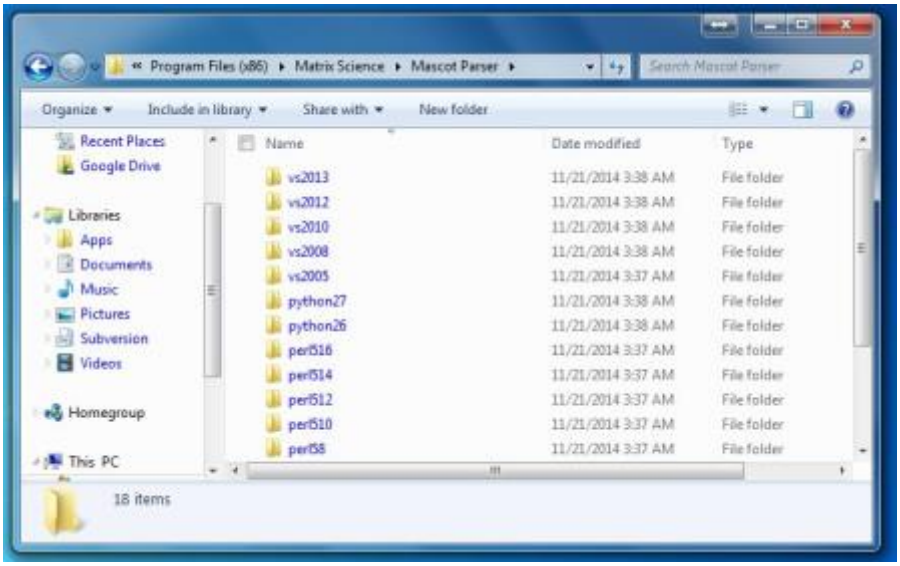
  - Go to each installer file downloaded and right click and select **Run as administrator.**



- Go to each installer file downloaded and right click and select **Run as administrator.**
- A window will pop up to select where to extract the files to:
  - Extract the 32-bit file to C:**Program Files(x86)\Matrix Science\Mascot Parser**
  - Extract the 62-bit file to C:**Program Files\Matrix Science\Mascot Parser**



(Below: sample results of 32-bit file extraction)



- **Building from Windows Command line**
  - Open a windows DOS prompt by going to the Start Menu of your Windows machine (alternative terminals encouraged as well; i.e. Console2 lets you navigate through the entire output when not creating your own output log file)
  - In text search box, type "cmd" to open terminal
  - In the terminal use the "cd" command to change directory to where you have the "**quickbuild.bat**" file.

- Once in that directory, type '**quickbuild.bat'** or '**quickbuild.bat - -i-agree-to-the-vendor-licenses' (to include the Vendor tools you installed in the steps above)** and the build script will execute
  - **NOTE: TIPS for first time users**
    - you may want to run **clean.bat** first before running **quickbuild.bat**
    - If you need to create a **quickbuild** log to assist you in debugging, please direct the output of your build into a log file by doing the following:

      quickbuild.bat (any flags) > quickbuildLog.txt

      After about a few minutes to a few hours (depending on how much of pwiz you want to build), the Windows DOS prompt will move to the next line meaning the build has completed and the quickbuildLogFile.txt is ready for viewing.

      (**sample flags**: toolset=.., address-model=.., --i-agree-to-the-vendor-licenses…)

      NOTE: Some of the additional flags that may be used are:

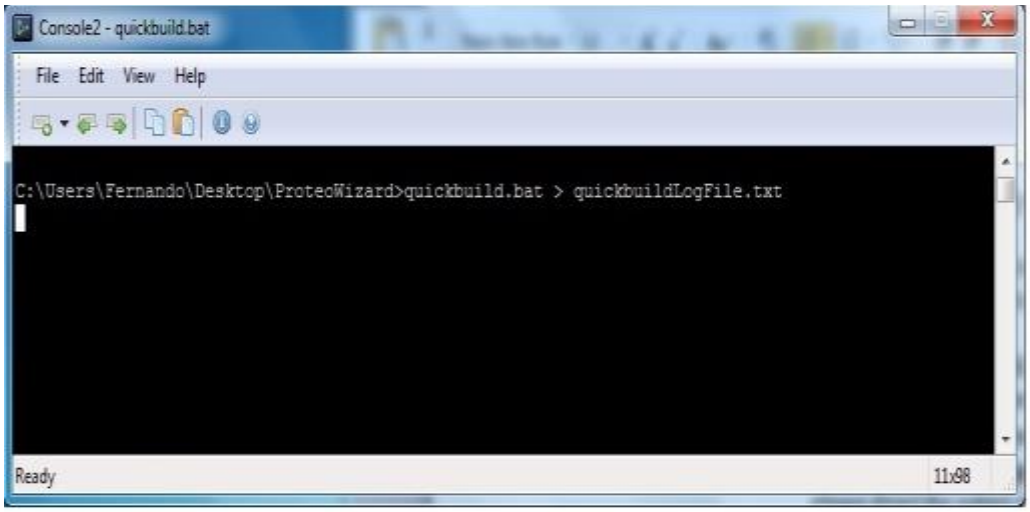
      address-model: You may use **address-model=**64 or 32 based on the type of development your are aiming for. 64 bit is often preferred because of the larger address space that is covered.

      toolset: You may use **toolset=msvc-**10.0,11.0 or 12.0, etc. Since Visual Studio 2013 is commonly used with proteowizard, you must use the corresponding Visual C++ compiler, which would be **msvc-12.0** . For Visual Studio 2012, use toolset-msvc-11.0, and for Visual Studio 2010, use toolset=msvc-10.0

      j4 or j8: This means that the developer wants to use either 4 or 8 cores respectively for parallel compilation. This will speed up the build time.

      without-compassxtract: --without-compassxtract build flag which will allow Bruker support with only the Baf2Sql API.

(Below: when build run is complete, cursor will move to next line)



- Feel free to email Juan Fernando at arguello at stanford.edu

How to Build Sample ProteoWizard Applications:

Next steps are to get an idea of how to build sample applications: hello_pwiz, mscat, and others.

- Building **hello_pwiz**
  - **hello_pwiz.cpp** is a simple example application to count the number of Spectra in a MSDataFile and then say hello to user, print the number of spectra counted, and print out the name of the output file created with these results. If an input filename is not provided when **hello_pwiz** is run, it will tell the user the correct usage.
  - After creating your own local repository of the proteowizard codebase you will find this application in the folder **technical.** An example path could be **C:\project\proteowizard\doc\technical\hello_pwiz**
    - In this folder you will find the files: **hello_pwiz.cpp** and **Jamfile.jam** which you will need to move to another location in the next steps
  - Go to the directory where your local repository was downloaded – your pwiz root directory.
  - Go to the **<pwiz root directory>/doc/technical/hello_pwiz** folder.

- In that folder you will find **hello_pwiz.cpp,** and a **Jamfile.jam** file to be used specifically with hello_pwiz.cpp
- Create a new folder **<pwiz root directory>/hello_pwiz** Copy those 2 files, the **.cpp** and **.jam** file, into that new folder
- To compile and build hello_pwiz.cpp type the following:
  - **quibckbuild.bat hello_pwiz --i-agree-to-the-vendor-licenses toolset=msvc-12.0**
  - Additional flags you can use: -j8 -j4 variant=release address-model=64 optimization=space --hash OR --without-compassxtract

- After a few minutes the build will complete and will create a **hello_pwiz.exe** executable file in your <pwiz root directory>/**hello_pwiz** folder
- To run the hello_pwiz application on sample data and produce an output file, type the following command:

  pwiz root>**hello_pwiz\hello_pwiz.exe example_data\tiny.pwiz.1.1.mzML**

- Your sample output should show something like:

  Hello, pwiz!
  # of spectra: 4
  Writing file **example_data\tiny.pwiz.1.1.mzML.out**

- **NOTE**: in the folder **example_data** there are other sample data files. Also, you may choose to copy a sample data file to your **hello_pwiz** folder so that the output file is also written and created into the **hello_pwiz** folder and not **example_data.**

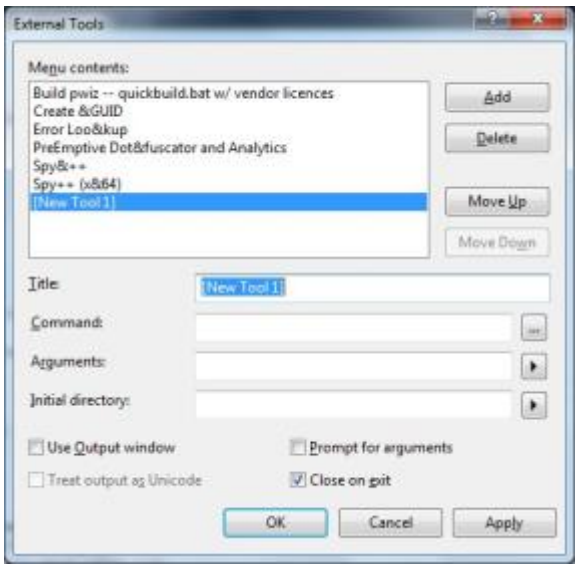
- Building **mscat**
  - **mscast.cpp** is an example application that iterates through the spectra and chromatograms in a data file, and writes out the **m/z-intensity** and **time-intensity** pairs . . . . .

ProteoWizard Building Direct from MSVS

- These instructions will be for executing the following build command
  - 'quickbuild.bat **--i-agree-to-the-vendor-licenses'**
  - you can adjust or enhance later for your specific needs.

- You can use a DOS command prompt to execute the command 'quickbuild.bat –help' to see how you would like to adjust or add new commands at a later time.

- Open up Microsoft Visual Studio 2013.
- Go to **Tools** drop down menu
- Go to bottom of menu and select **External Tools**
- Click on the **Add** button



- In the **Title** text box, type the name of your new tool/command (i.e. **Build pwiz -- quickbuild.bat w/ vendor licenses**)
- In the **Command** text box, enter the path to where **quickbuild.bat** is located (**i.e. C:\project1Pwiz\Proteowizard\quickbuild.bat**)
- In the **Arguments** text box, enter **--i-agree-to-the-vendor-licenses**
- In the **Initial directory box**, select **Solution Directory** from the drop down menu
- Select the text box that says **Use Output Window**
  - You may also decide to select **Prompt for arguments** for when building all of proteowizard or just parts, as well as using some of the different flags mentioned previously.
- **NOTE:** Make sure to move the Title of the new command to the **TOP** of the list of commands by using the **Move Up** button.
- Click **Apply** and then OK.

Setup a Hot Key combination for new command:

Next steps are to configure a hot key command to quickly run quickbuild.bat while you have Visual Studio open.

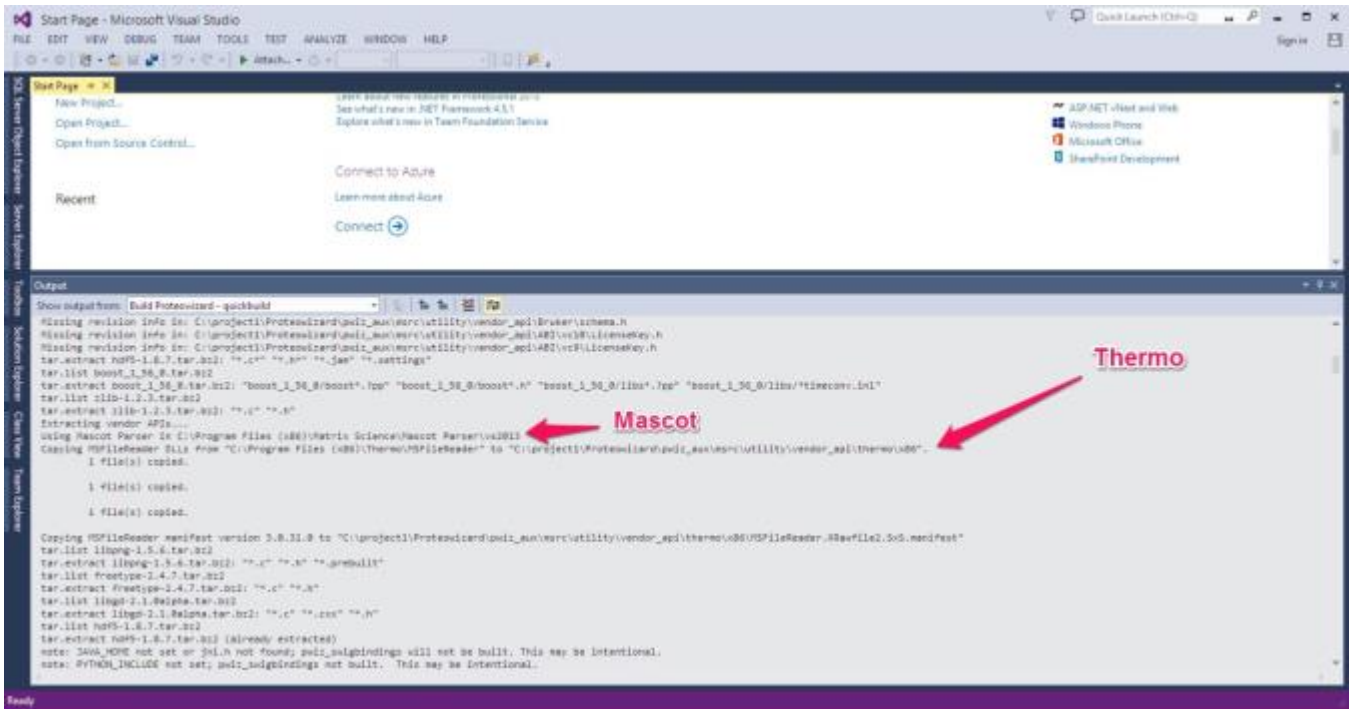
- Now go back to **Tools** menu and select **Customize** from bottom of drop down menu
- Click on **Keyboard** at bottom of open window
- In the **Show Commands Containing** text box, type in **Tools.ExternalCommand**
    - If you followed the NOTE above, you will need to select 'Tools.ExternalCommand**1'** which means you moved the command you created above to the top of the list of possible commands. Command 1, 2, etc. corresponds to where in the list of commands your new command was created.
- In the **Press shortcut keys:** box, type in a set of key strokes to serve as a hotkey for your new command to build.
    - i.e. **Ctrl + Alt + Shift + Q** for the quickbuild command
    - the more keys involved, the less chance you will run a build by mistake, when fingers hit a random key. :)
- Then click **Assign**
- Then click **Ok** and you are ready to use your new command.

USING YOUR NEW COMMAND:

Next is how to execute the new command/tool you added to your Visual Studio list of tools.

- Open your Visual Studio IDE.
- Click in **Start Page** panel (or Team Explorer Connect or others) to move focus of IDE to that panel
- Then press your hot key commands for building such as **Ctrl + Alt + Shift + B**
- If you opted for **Prompt for arguments** above, type in the specific arguments specific to your needs when a new window pops up. Then click **OK.**
- Below see that the **Output Window** is being filled with the ongoing results of your build run.

(Below: shows Mascot and MSFileReader if installed properly and you use the agree-to vendor-licenses flag as mentioned previously)



- Feel free to email Juan Fernando – arguello at stanford.edu