C# Wrapper

Documentation and Users Guide

December 7th 2015, version 1.1

1.0 INTRO

The C# wrapper for wolfSSL is a way for C# developers to easily integrate wolfSSL TLS/DTLS functionality into their product. It contains wrappers for the functions needed to open and maintain a TLS or DTLS connection with the option of using PSK.

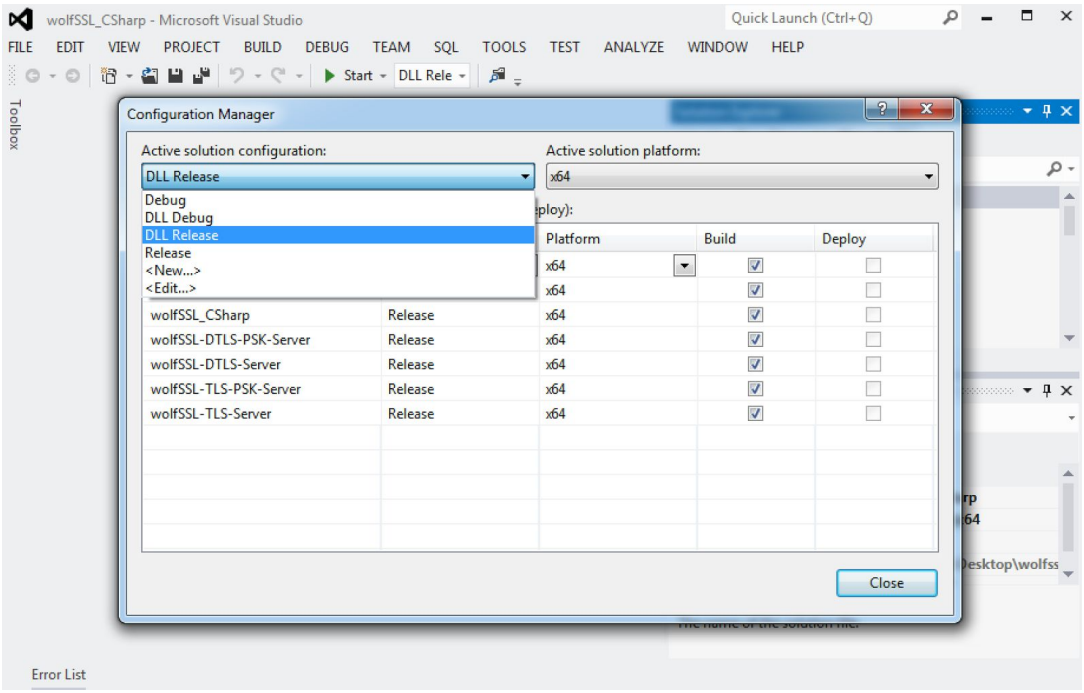
1.1 Current Status Notes

2.0 INSTALLATION

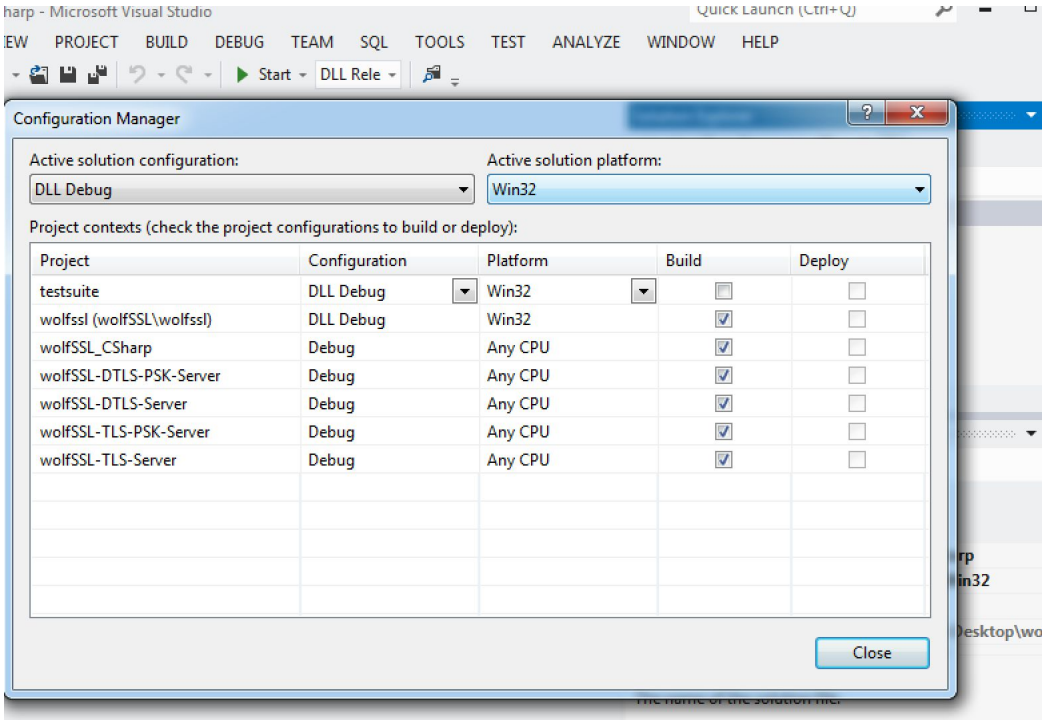
2.1 Dependencies

It is required to have the wolfSSL DLL library. This is automatically built with the Visual Studio project wolfssl/wrapper/CSharp/wolfSSL_CSharp.

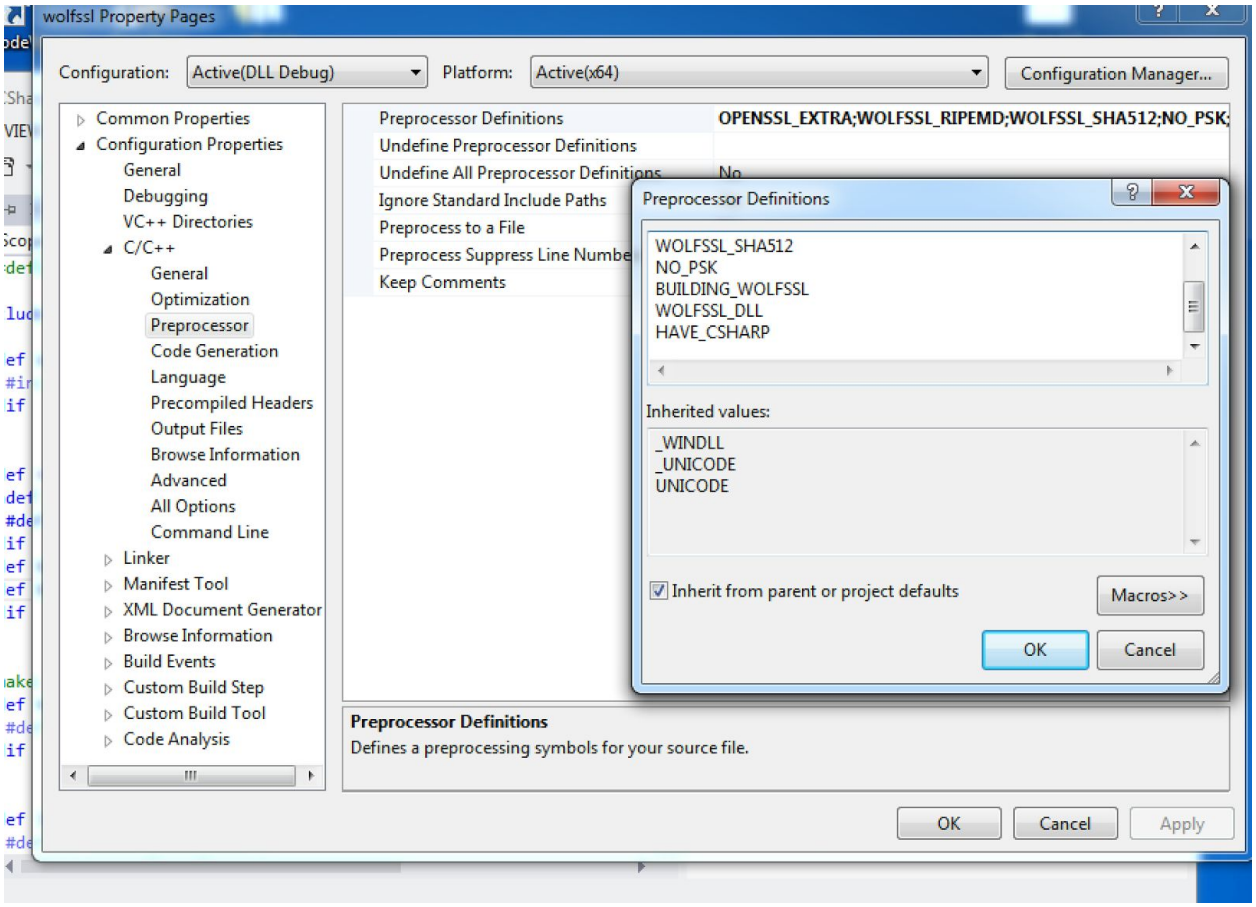
First is to set up the build option used, click BUILD->Configuration Manager… In the top left change to prefered Active solution configuration, either DLL Debug or DLL Release and then select the Active solution platform being built on either 32bit(Win32) or 64bit(x64) machine. In the first screenshot it was built on a 64bit OS.



For a 32bit system it would look like the following screenshot.

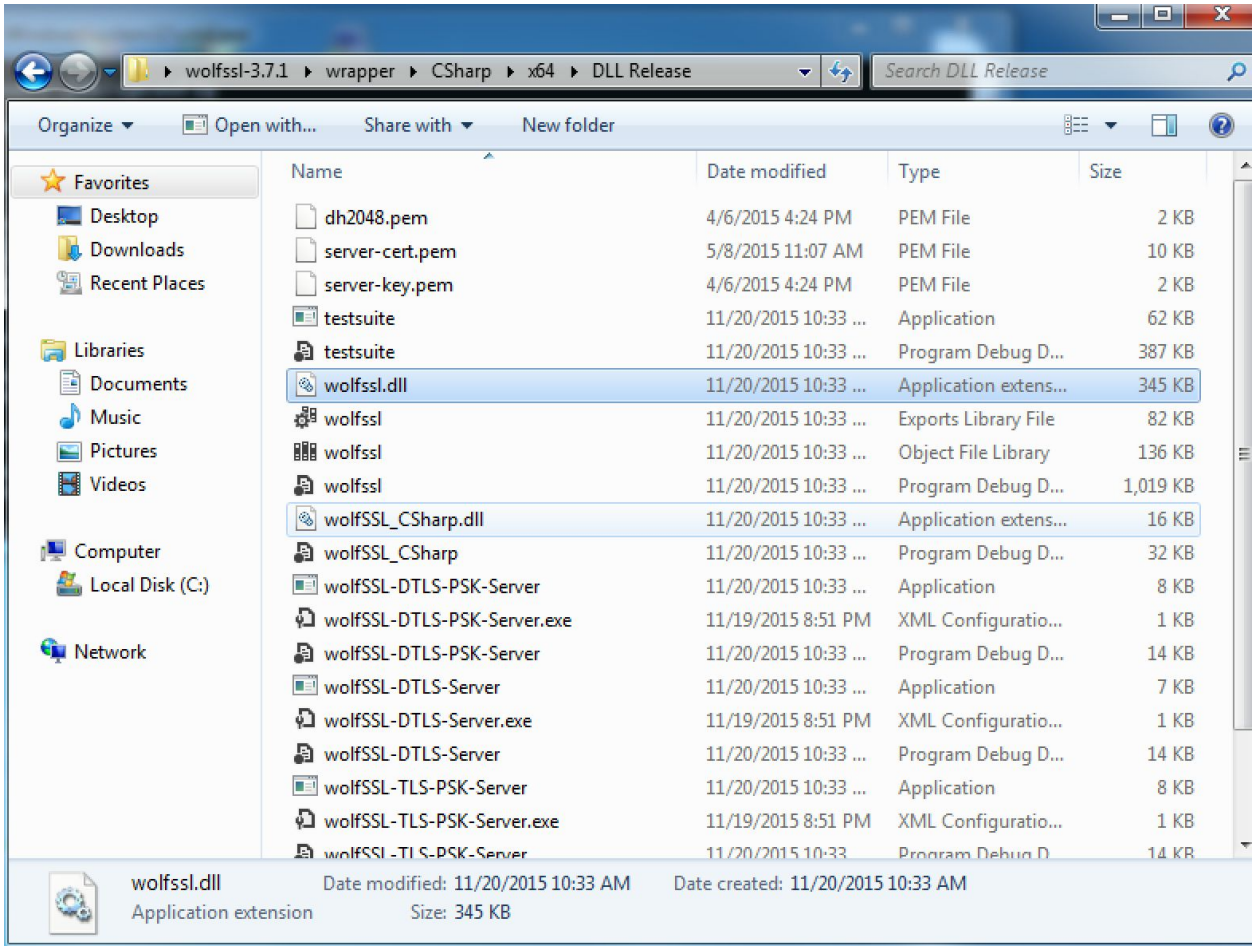


After setting up the build options a preprocessor flag HAVE_CSHARP needs to be added. These flags can be adjusted by right clicking on wolfssl/wolfssl project in Visual Studio. Selecting properties then expanding C/C++ and opening Preprocessor option. Click the down arrow next to Preprocessor Definitions and select edit.



2.2 Building C# wrapper

To build the wolfSSL library, the CSharp wrapper, and examples, click BUILD->Build Solution. This will create a wolfssl.dll and wolfSSL_CSharp.dll in a created DLL Release or DLL Debug folder depending on which DLL option was selected. The new folder created with the build of wolfSSL can be found in the wolfssl/wrapper/CSharp directory. Note that on 64bit builds it will put this directory into the folder x64.



3.0 EXAMPLE SERVER

There are a couple of example servers included; TLS, DTLS, and another two with each using PSK. Building of each example can be done in a similar fashion as above. All examples will create an exe file to then run, and the server will be bound to 0.0.0.0 allowing any IP to connect. Since allowing any IP to connect a firewall warning could pop up this is expected and for testing from external connections click allow.

If testing the connection from a linux type system using the C wolfSSL client examples make sure the library on the linux system has been built with configuration options to have DTLS and PSK.

When building and testing the example IO callbacks the suite used is a static PSK one. To be able to use static PSK suites wolfSSL will need to be built with the preprocessor flag WOLFSSL_STATIC_PSK.

4.0 EXTRA

This is a list of some standard ways for using the created DLLs in custom projects other than provided examples. One way is by placing the created wolfSSL_CSharp.dll and wolfssl.dll in the directory C:\Windows\system. A second option of changing the environment variable path is available to allow for the loader to find the wolfssl and wolfSSL_CSharp DLLs. Finally a third option of just using the existing wolfSSL_CSharp solution and adding on to it in a similar fashion as the examples.

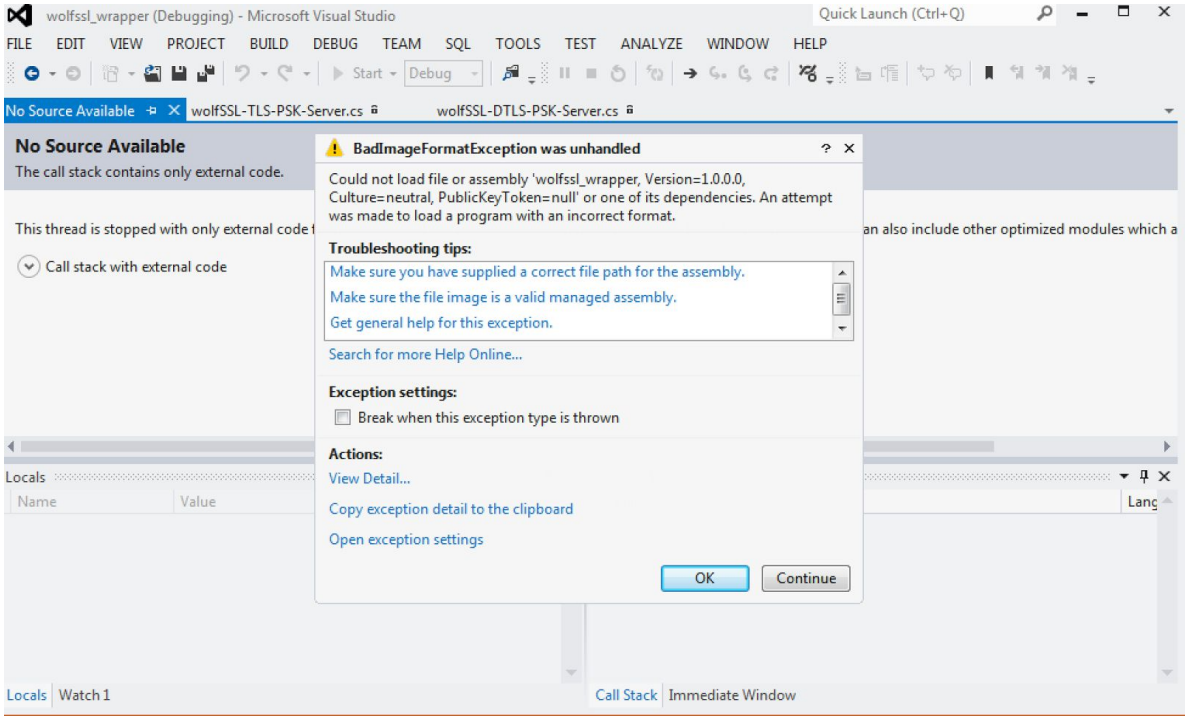
C# wrapper logging errors can be set up with a function callback being passed to wolfSSL.SetLogging. An example of this can be seen in wolfSSL-TLS-Server.cs.

Though the project uses preset call backs for reading and writing when a new CTX structure is created there is the option to set custom callbacks with the wolfssl.SetIORecv and wolfssl.SetIOSend. These functions require the input of the CTX structure to set and a function that fulfills the CallbackIORecv_delegate and CallbackIOSend_delegate requirements.

Client connect has been added but not fully tested the server side was focused on.

5.0 Troubleshooting

- If you get the following error it is likely because of a mismatch in build architectures.



To fix make sure all build configurations are for the same architecture ie x64.

- If the application runs really quick and does not complete a connection check the path for loading in the certificate and key, also check that HAVE_CSHARP preprocessor is added to wolfssl.

- If you get the error unable to locate wolfSSL_CSharp.dll make sure to add reference to the created wolfSSL_CSharp.dll for the project. This can be done by right clicking on the project and selecting add reference. Then select solution->projects and chose wolfSSL_CSharp or by installing the created wolfssl.dll and wolfSSL_CSharp.dll.