

# 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.

	EW PROJECT BUILD DEBUG	TEAM SQL TOOLS Start + DLL Rele + 🎜 <sub>→</sub>	S TEST ANALYZE	WINDOW HEL	p			
Configuration Manager Active solution configuration: DLL Release Debug		Active solution platform:			•	-	μ× 	
	DLL Debug DLL Release Release <new> <edit></edit></new>		Platform x64	Build	Deploy			
	wolfSSL_CSharp wolfSSL-DTLS-PSK-Server wolfSSL-DTLS-Server wolfSSL-TLS-PSK-Server wolfSSL-TLS-PSK-Server	Release Release Release Release Release	x64 x64 x64 x64 x64 x64 x64				•	т Т Х т
						rţ :6	p 4 	olfss
Error List				וכיוטוויביטי גווב שט	Close			×

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

inguration manager						
Active solution configuration: DLL Debug		Active solution p	Active solution platform: Win32			
		▼ Win32				
roject contexts (check the proje	ct configurations to build o	or deploy):				
Project	Configuration	Platform		Build	Deploy	
testsuite	DLL Debug	▼ Win32	•			
wolfssl (wolfSSL\wolfssl)	DLL Debug	Win32		1		
wolfSSL_CSharp	Debug	Any CPU		1		
wolfSSL-DTLS-PSK-Server	Debug	Any CPU		<b>V</b>		
wolfSSL-DTLS-Server	Debug	Any CPU		<b>V</b>		
wolfSSL-TLS-PSK-Server	Debug	Any CPU		1		00000
wolfSSL-TLS-Server	Debug	Any CPU		<b>V</b>		
						En .
						in32

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.

Common Properties	Preprocessor Definitions	OPENSSL_EXTRA;WOLFSSL_RIPEMD;WOLFSSL_SHA512;NC
Configuration Properties	Undefine Preprocessor Definitions	
General	Undefine All Preprocessor Definition	ans No
Debugging	Ignore Standard Include Paths	Preprocessor Definitions
VC++ Directories	Preprocess to a File	
⊿ C/C++	Preprocess Suppress Line Numbe	WOLFSSL_SHA512
General	Keep Comments	NU_PSK BUILDING WOLESSI
Optimization		WOLFSSL DLL
Preprocessor		HAVE_CSHARP
Code Generation		
Language		- P
Precompiled Headers		Inherited values:
Output Files		WINDLL
Browse Information		UNICODE
Advanced		UNICODE
All Options		
Command Line		
Linker		
> Manifest Tool		✓ Inherit from parent or project defaults Macros>:
> XIVIL Document Generator		
Build Events		OK Cancel
Custom Build Sten		
Custom Build Tool		
Code Analysis	Preprocessor Definitions	
	Defines a preprocessing symbols for y	our source file.

#### 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.

wolfssl-	3.7.1 ▶ wrapper ▶ CSharp ▶ x64 ▶ DLL Rel	ease 🔻 🍫	Search DLL Release	
)rganize 🔻 🔳 Ope	n with Share with 🔻 New folder			≣ ▼ 🔟
Favorites	Name	Date modified	Туре	Size
📃 Desktop	dh2048.pem	4/6/2015 4:24 PM	PEM File	2 KB
\rm Downloads	server-cert.pem	5/8/2015 11:07 AM	PEM File	10 KB
🔠 Recent Places	server-key.pem	4/6/2015 4:24 PM	PEM File	2 KB
	📧 testsuite	11/20/2015 10:33	Application	62 KB
🖥 Libraries	🗿 testsuite	11/20/2015 10:33	Program Debug D	387 KB
Documents	🔕 wolfssl.dll	11/20/2015 10:33	Application extens	345 KB
J Music	츓 <sup>명</sup> wolfssl	11/20/2015 10:33	Exports Library File	82 KB
Pictures	🔠 wolfssl	11/20/2015 10:33	Object File Library	136 KB
Videos	Wolfssl	11/20/2015 10:33	Program Debug D	1,019 KB
	🚳 wolfSSL_CSharp.dll	11/20/2015 10:33	Application extens	16 KB
📕 Computer	wolfSSL_CSharp	11/20/2015 10:33	Program Debug D	32 KB
🏭 Local Disk (C:)	wolfSSL-DTLS-PSK-Server	11/20/2015 10:33	Application	8 KB
	wolfSSL-DTLS-PSK-Server.exe	11/19/2015 8:51 PM	XML Configuratio	1 KB
Network	wolfSSL-DTLS-PSK-Server	11/20/2015 10:33	Program Debug D	14 KB
	wolfSSL-DTLS-Server	11/20/2015 10:33	Application	7 KB
	wolfSSL-DTLS-Server.exe	11/19/2015 8:51 PM	XML Configuratio	1 KB
	wolfSSL-DTLS-Server	11/20/2015 10:33	Program Debug D	14 KB
	wolfSSL-TLS-PSK-Server	11/20/2015 10:33	Application	8 KB
	wolfSSL-TLS-PSK-Server.exe	11/19/2015 8:51 PM	XML Configuratio	1 KB
	wolfSSI -TI S-PSK-Server	11/20/2015 10:33	Program Debug D	14 KR

## 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.

wolfssl_wrapper (Debugging) - Microsoft V	Quick L	aunch (Ctrl+Q)			
FILE EDIT VIEW PROJECT BUILD E	EBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HEL	LP			
<ul> <li>○ - ○   B - ▲ ■ ■   9 - C -</li> </ul>	▶ Start - Debug - B <sub>+</sub> II = S S <sub>0</sub> → S <sub>1</sub> C C C	<b>,</b>	ালি হিক ∎ গাগায়়		
No Source Available 😐 🗙 wolfSSL-TLS-PSK-Se	rver.cs 🕯 wolfSSL-DTLS-PSK-Server.cs 🕯		<b>*</b>		
No Source Available	A BadImageFormatException was unhandled	? X			
The call stack contains only external code.	Could not load file or assembly 'wolfssl_wrapper, Version=1.0.0.0,				
This thread is stopped with only external code t	Culture=neutral, PublicKeyToken=null' or one of its dependencies. An attem was made to load a program with an incorrect format.	npt	an also include other optimized modules which a		
	Troubleshooting tips:				
Call stack with external code	Make sure you have supplied a correct file path for the assembly.	<u>^</u>			
	Make sure the file image is a valid managed assembly.	H			
	Get general help for this exception.	*			
	Search for more Help Online				
	Exception settings:				
	Break when this exception type is thrown				
•	Actions:		•		
Locals	View Detail		- 4 ×		
Name Value	Copy exception detail to the clipboard		Lang		
	Open exception settings				
	OK Contin	nue			
			*		
Locals Watch 1	Call Stack Immediate Window				

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.