Building the Spout libraries

Visual Studio 2017 projects are available to build the Spout SDK as a dll, or to create a C-compatible dll that can be used with other compilers. Pre-built binaries for 32 bit and 64 bit are also available in the respective "Binaries" folder.

CMAKE build

If you are using a compiler other than Visual Studio and wish to make modifications, these libraries can be built using Cmake. (https://cmake.org).

Installation

1) Download and install CMake for Windows. The easiest way is to use the installer. At the time of writing this was "cmake-3.19.2-win64-x64.msi". Get it from the downloads page (https://cmake.org/download/).
2) As they instruct, if you are upgrading to the latest version, you have to un-install any previous version.
3) Run the installer. Default options are OK, but it's useful to create a Desktop icon. "Finish" to complete.

Generating a project

1) On the desktop, find the CMake icon and open the CMake GUI.
2) For "Where is the source code:", click "Browse Source", navigate to wherever you saved the Spout repository and select the root folder, usually named "Spout2".
3) For "Where to build the binaries:", click "Browse Build" and navigate to "Spout2\BUILD".
4) At bottom click "Configure" to open the configuration dialog.
5) For "Specify the generator for this project", select your compiler and other options you may require. "Optional platform ..." will be empty. Default build is 32 bit. Leave it at that for now and other defaults. Click "Finish".

After completion you will see various build settings in red.

- ✔ SPOUT_BUILD_CMT - for Visual Studio compilers, this sets a project option to include the runtime libraries into the dll. Then the user does not need to install the runtime separately. Check it off if you know what you are doing. Leave it on if you are unsure.
- ✔ SPOUT_BUILD_LIBRARY - builds a C-compatible library "SpoutLibrary" which could be of interest if you are not using Visual Studio.
- ✔ SPOUT_BUILD_SPOUTDX - builds the Spout DirectX11 support class "SpoutDX" as a dynamic link library. Default is off.
- ✔ SPOUT_BUILD_SPOUTDX_EXAMPLES – build the examples for using the SpoutDX support class. Default is off.

Finally Click "Generate".

Building the projects

When you see "Generating done", click "Open Project".
In the compiler IDE you will see the following projects :

ALL_BUILD	
Spout_static	(Spout SDK static library)
SpoutDX	(SpoutDX support class - option)

SpoutLibrary	(C compatible Spout library - option)
Spout.dll	(Spout SDK dll)
Tutorial04	(DirectX11 SpoutDX sender example - option)
Tutorial07	(DirectX11 SpoutDX receiver example - option)
WinSpoutDXreceiver	(Windows SpoutDX receiver example - option)
WinSpoutDXsender	(Windows SpoutDX sender example - option)
WinSpoutDXvideo	(Windows SpoutDX video example - option)
ZERO_CHECK	

Change to "Release" and build "ALL_BUILD". When it has finished, browse to the "BUILD" folder you previously selected. In the "Binaries" folder you will find :

Win32
- SpoutSDK.lib
- SpoutSDK.dll
- SpoutLibrary.lib
- SpoutLibrary.dll
- SpoutDX.lib
- SpoutDX.dll
- Spout_static.lib

- SpoutSDK - the Spout SDK built as a dynamic link library
- SpoutLibrary - a C-compatible library dynamic link library
- SpoutDX - Spout DirectX support class as a dynamic link library
- Spout_static.lib - the Spout SDK built as a static library

SpoutDX examples

Note that when building the SpoutDX examples with Visual Studio project, ALL_BUILD is the default “Startup Project”. Right click on the executable project you are interested in and select as “Startup Project” from the context menu.

Changing the CMake options

1) Close compiler IDE
2) Start CMake GUI if it has been closed
3) Select any of the options available and check ON or OFF
4) Click "Generate" again to set the new options.
5) "Open Project" and re-build

Changing Platform

For example to build 64 bit instead of the default 32 bit.

1) From the CMake GUI select "File > Delete cache" and do it.
2) Click "Configure"
3) This time, select the "Optional platform" that you want. For Visual Studio there is a drop-down list and the "x64" option.
4) "Generate", "Open Project", change to "Release" and build.

In the "Binaries" folder you will find an "x64" folder with the 64 bit versions of the libraries. They have the same names so be careful not to mix them up.

Credit

Thanks and credit for the CMake files which were first developed and contributed by Alexandre Buge (https://github.com/Qlex42).