Building OpenSSL for Visual Studio

Posted on June 20, 2012 by 3noch • 4 Comments

Building OpenSSL for Visual Studio on Windows is mostly straight-forward, but it has some quirks. I'll document the results of my wrestling here so that future attempts will be less painful.

What you need

You need to install…

- Visual Studio 2010 (this will likely work with older versions as well)
- ActivePerl [1]
- Latest version of OpenSSL source-code [2]

Setting up for the build

Unzip[3] the OpenSSL source code into two different folders, one for the 32-bit build and one for the 64-bit build[4]. So, for example, you might end up with `C:\openssl-src-32` and `C:\openssl-src-64`.

Building the 32-bit static libraries

1. Open the **Visual Studio Command Prompt (2010)**[5].
2. `cd` to your OpenSSL source folder for 32-bit (e.g. `cd C:\openssl-src-32`).
3. Run the following: [6]

```
perl Configure VC-WIN32 --prefix=C:\Build-OpenSSL-VC-32
ms\do_ms
nmake -f ms\nt.mak
nmake -f ms\nt.mak install
```

Your outputs will be in `C:\Build-OpenSSL-VC-32`.

Building the 32-bit static libraries with debug symbols

These steps will embed the debug symbols directly into the `.lib` files. **Don't expect to see any `.pdb` files.**

1. Open the **Visual Studio Command Prompt (2010)**.
2. `cd` to your OpenSSL source folder for 32-bit (e.g. `cd C:\openssl-src-32`).
3. Run the following:

```
perl Configure debug-VC-WIN32 --prefix=C:\Build-OpenSSL-VC-32-dbg
ms\do_ms
```

4. In a text editor (like **Notepad**), open `ms\nt.mak` and replace all occurrences of `/Zi` with `/Z7`. There should be three replacements.[7]
5. Run the following:

```
nmake -f ms\nt.mak
nmake -f ms\nt.mak install
```

Your outputs will be in `C:\Build-OpenSSL-VC-32-dbg`. Make sure you rename them to something like `libeay32-debug.lib` and `ssleay32-debug.lib`.

Building the 64-bit static libraries

RECENT POSTS

- UNC Paths with Python
- Building OpenSSL for Visual Studio
- StringLike in Python

CHECK OUT OUR CODE

Covenant Eyes on GitHub

SEARCH

Type to search, then press enter

1. Open the **Visual Studio x64 Win64 Command Prompt (2010)** (in the **Start** menu).
2. `cd` to your OpenSSL source folder for 64-bit (e.g. `cd C:\openssl-src-64`).
3. Run the following:

```
perl Configure VC-WIN64A --prefix=C:\Build-OpenSSL-VC-64
ms\do_win64a
nmake -f ms\nt.mak
nmake -f ms\nt.mak install
```

Your outputs will be in `C:\Build-OpenSSL-VC-64`.

Note: The outputs of the 64-bit build are still named `libeay32.lib` and `ssleay32.lib`. You'll have to rename them more sensibly yourself.

Building the 64-bit static libraries with debug symbols

These steps will embed the debug symbols directly into the `.lib` files. **Don't expect to see any `.pdb` files.**

1. Open the **Visual Studio x64 Win64 Command Prompt (2010)**.
2. `cd` to your OpenSSL source folder for 64-bit (e.g. `cd C:\openssl-src-64`).
3. Run the following:

```
perl Configure debug-VC-WIN64A --prefix=C:\Build-OpenSSL-VC-64-dbg
ms\do_win64a
```

4. In a text editor (like **Notepad**), open `ms\nt.mak` and replace all occurrences of `/Zi` with `/Z7` **except on the line starting with `ASM`**. There should be two replacements. [8]
5. Run the following:

```
nmake -f ms\nt.mak
nmake -f ms\nt.mak install
```

Your outputs will be in `C:\Build-OpenSSL-VC-64-dbg`. Make sure you rename them to something like `libeay64-debug.lib` and `ssleay64-debug.lib`.

What not to do

I tried every method under the sun to get a Windows build of OpenSSL that would link against Visual Studio projects. I learned a great deal along the way. Here's what I learned **not** to do:

- Don't blindly follow the Windows 32-bit/64-bit installation instructions provided in the OpenSSL source folder. Get guidance online.
- Don't build OpenSSL in Cygwin. It's easy. It won't link against Visual Studio.
- Don't build OpenSSL in MSYS or MinGW. It's hard. It won't link against Visual Studio.
- Don't try to use NASM like the Windows installation instructions mention. It's not necessary for Visual Studio builds. (It only supports 32-bit anyway.)
- Strawberry Perl doesn't always work in these weird configurations. ActivePerl seemed more stable.
- Don't try to build 32-bit and 64-bit OpenSSL in the same folder. The first build will leave artifacts that will mess up the second build. (Running a clean isn't enough, apparently.)
- Don't try to build 32-bit OpenSSL inside of Visual Studio's 64-bit command prompt and vice versa. It doesn't work.

References

These were very helpful places:

- https://github.com/freelan-developers/freelan-buildtools/blob/master/INSTALL.md

---

Footnotes:

1. Do not use Strawberry Perl (see comments for this post).

2. OpenSSL version 1.0.1c was the latest at the time of writing.

3. 7-zip is good for unzipping `.tar.gz` files on Windows. It's a two-step process.

4. OpenSSL's build scripts are not clever enough to handle two different platform builds in sequence. Separate platform builds must start from scratch.

5. You can find it somewhere in the **Start** menu.

6. Using `ms\\ntdll.mak` will build the shared library instead.

7. The `/Zi` option works, but it's hard to find the right `.pdb` file without specifying more options. For the sake of simplicity, the `/Z7` option just embeds all the debug symbols into the `.lib` files. Read more here.

8. For the 64-bit build, Visual Studio uses MASM (`ml64.exe`) to compile assembly code. According to MASM's documentation, the `/Z7` option is not supported.

Filed Under: Documentation

← StringLike in Python UNC Paths with Python →

4 Responses to *Building OpenSSL for Visual Studio*

Noah says:
February 22, 2013 at 10:19 am

Thanks for documenting this. For the record, Strawberry Perl does not work.

Reply

3noch says:
February 22, 2013 at 12:45 pm

Thank you, Noah! I've updated the footnote to reflect your input.

Reply

erik says:
June 17, 2013 at 9:54 am

Thank you very much for posting this. My first attempt failed because nasm is not set up right on my machine. I built it again with no-asm and that worked fine. I tried 1.0.1c and 1.0.1e, both OK. -erik

Reply

Yury Schkatula says:
October 23, 2013 at 6:41 pm

Good staff – thanks. The only trick here is if you rely on MSYS perl instead of ActivePerl: http://stackoverflow.com/questions/7680189/openssl-cant-build-in-vc-2010So both facts allowed me to build LIB files, finally!

Reply

Leave a reply

Name *

Email *

Website

Visual Text

Path: p

Post Comment

Powered by Headway, the drag and drop WordPress theme

Go To Top

Copyright © 2013 Dev Room