Building Zabbix agent on Windows

Overview

This section demonstrates how to build Zabbix Windows agent binaries from sources with or without TLS.

Compiling OpenSSL

The following steps will help you to compile OpenSSL from sources on MS Windows 10 (64-bit).

1. For compiling OpenSSL you will need on Windows machine:
    1. C compiler (e.g. VS 2017 RC),
    2. NASM (https://www.nasm.us/),
    3. Perl (e.g. Strawberry Perl from http://strawberryperl.com/),
    4. Perl module Text::Template (cpan Text::Template).
2. Get OpenSSL sources from https://www.openssl.org/. OpenSSL 1.1.1 is used here.
3. Unpack OpenSSL sources, for example, in E:\openssl-1.1.1.
4. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 RC.
5. Go to the OpenSSL source directory, e.g. E:\openssl-1.1.1.
    1. Verify that NASM can be found:

```
e:\openssl-1.1.1> nasm --version
NASM version 2.13.01 compiled on May  1 2017
```

6. Configure OpenSSL, for example:

```
e:\openssl-1.1.1> perl E:\openssl-1.1.1\Configure VC-WIN64A no-shared
no-capieng no-srp no-gost no-dgram no-dtls1-method no-dtls1_2-method  -
-api=1.1.0 --prefix=C:\OpenSSL-Win64-111-static --
openssldir=C:\OpenSSL-Win64-111-static
```

    - Note the option 'no-shared': if 'no-shared' is used then the OpenSSL static libraries libcrypto.lib and libssl.lib will be 'self-sufficient' and resulting Zabbix binaries will include OpenSSL in themselves, no need for external OpenSSL DLLs. Advantage: Zabbix binaries can be copied to other Windows machines without OpenSSL libraries. Disadvantage: when a new OpenSSL bugfix version is released, Zabbix agent needs to recompiled and reinstalled.
    - If 'no-shared' is not used, then the static libraries libcrypto.lib and libssl.lib will be using OpenSSL DLLs at runtime. Advantage: when a new OpenSSL bugfix version is released, probably you can upgrade only OpenSSL DLLs, without recompiling Zabbix agent. Disadvantage: copying Zabbix agent to another machine requires copying OpenSSL DLLs, too.
7. Compile OpenSSL, run tests, install:

```
e:\openssl-1.1.1> nmake
e:\openssl-1.1.1> nmake test
...
All tests successful.
```

```
Files=152, Tests=1152, 501 wallclock secs ( 0.67 usr +  0.61 sys =
1.28 CPU)
Result: PASS
e:\openssl-1.1.1> nmake install_sw
```

'install_sw' installs only software components (i.e. libraries, header files, but no documentation). If you want everything, use "nmake install".

Compiling PCRE

1. Download PCRE library (mandatory library since Zabbix 4.0) from pcre.org, version 8.XX; not pcre2 (ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.41.zip)
2. Extract to directory *E:\pcre-8.41*
3. Install CMake from https://cmake.org/download/, during install select: and ensure that cmake\bin is on your path (tested version 3.9.4).
4. Create a new, empty build directory, preferably a subdirectory of the source dir. For example, *E:\pcre-8.41\build*.
5. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 and from that shell environment run cmake-gui. Do not try to start Cmake from the Windows Start menu, as this can lead to errors.
6. Enter *E:\pcre-8.41* and *E:\pcre-8.41\build* for the source and build directories, respectively.
7. Hit the "Configure" button.
8. When specifying the generator for this project select "NMake Makefiles".
9. Create a new, empty install directory. For example, *E:\pcre-8.41-install*.
10. The GUI will then list several configuration options. Make sure the following options are selected:
    - **PCRE_SUPPORT_UNICODE_PROPERTIES** ON
    - **PCRE_SUPPORT_UTF** ON
    - **CMAKE_INSTALL_PREFIX** *E:\pcre-8.41-install*
11. Hit "Configure" again. The adjacent "Generate" button should now be active.
12. Hit "Generate".
13. In the event that errors occur, it is recommended that you delete the CMake cache before attempting to repeat the CMake build process. In the CMake GUI, the cache can be deleted by selecting "File > Delete Cache".
14. The build directory should now contain a usable build system - *Makefile*.
15. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 and navigate to the *Makefile* mentioned above.
16. Run NMake command:

```
E:\pcre-8.41\build> nmake install
```

Compiling Zabbix

The following steps will help you to compile Zabbix from sources on MS Windows 10 (64-bit). When compiling Zabbix with/without TLS support the only significant difference is in step 4.

1. On a Linux machine check out the source from SVN:

```
$ svn co svn://svn.zabbix.com/tags/4.2.0
$ cd 4.2.0/
$ ./bootstrap.sh
$ ./configure --enable-agent --enable-ipv6 --prefix=`pwd`
$ make dbschema
$ make dist
```

2. Copy and unpack the archive, e.g. zabbix-4.2.0.tar.gz, on a Windows machine.
3. Let's assume that sources are in e:\zabbix-4.2.0. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 RC. Go to E:\zabbix-4.2.0\build\win32\project.
4. Compile zabbix_get, zabbix_sender and zabbix_agent.
    - without TLS:

      ```
      E:\zabbix-4.2.0\build\win32\project> nmake /K
      PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-
      install\lib
      ```

    - with TLS:

      ```
      E:\zabbix-4.2.0\build\win32\project> nmake /K -f Makefile_get
      TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include
      TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre-8.41-
      install\include PCRELIBDIR=E:\pcre-8.41-install\lib
      E:\zabbix-4.2.0\build\win32\project> nmake /K -f Makefile_sender
      TLS=openssl TLSINCDIR="C:\OpenSSL-Win64-111-static\include
      TLSLIBDIR="C:\OpenSSL-Win64-111-static\lib"
      PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-
      install\lib
      E:\zabbix-4.2.0\build\win32\project> nmake /K -f Makefile_agent
      TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include
      TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre-8.41-
      install\include PCRELIBDIR=E:\pcre-8.41-install\lib
      ```

5. New binaries are located in e:\zabbix-4.2.0\bin\win64. Since OpenSSL was compiled with 'no-shared' option, Zabbix binaries contain OpenSSL within themselves and can be copied to other machines that do not have OpenSSL.

Compiling Zabbix with LibreSSL

The process is similar to compiling with OpenSSL, but you need to make small changes in files located in the `build\win32\project` directory:

- In `Makefile_tls` delete `/DHAVE_OPENSSL_WITH_PSK`. i.e. find

  ```
  CFLAGS =    $(CFLAGS) /DHAVE_OPENSSL /DHAVE_OPENSSL_WITH_PSK
  ```

  and replace it with

  ```
  CFLAGS =    $(CFLAGS) /DHAVE_OPENSSL
  ```

- In `Makefile_common.inc` add `/NODEFAULTLIB:LIBCMT` i.e. find

```
/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO
/PDB:$(TARGETDIR)\$(TARGETNAME).pdb
```

and replace it with

```
/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO
/PDB:$(TARGETDIR)\$(TARGETNAME).pdb /NODEFAULTLIB:LIBCMT
```