1) Download and install Perl for MS Windows from this location

https://strawberryperl.com/download/5.32.1.1/strawberry-perl-5.32.1.1-64bit.msi

2) Download openssl-1.0.1u.tar.gz from this location

https://www.openssl.org/source/old/1.0.1/openssl-1.0.1u.tar.gz

3) Configure OpenSSL.

|  |
| --- |
| perl Configure VC-WIN32 no-asm --prefix=d:\qt\openssl\openssl-1.0.1u-vs2012-static-x32 no-shared |

|  |
| --- |
| ms\do\_ms |

|  |
| --- |
| nmake -f ms\nt.mak |

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

4) Download file qtbase-opensource-src-5.1.0.zip from this location.

https://download.qt.io/archive/qt/5.1/5.1.0/submodules/qtbase-opensource-src-5.1.0.zip

5) Make directory d:\qt\qtbase-opensource-src-5.1.0

6) Initialize the empty repository:

|  |
| --- |
| git init |

7) Extract files from downloaded archive.

8) Add files to stash and make the first commit.

|  |
| --- |
| git add –A  git commit -m "Add stock version" |

9) Amend the file mkspecs\win32-msvc2012\qmake.conf

|  |
| --- |
| QMAKE\_CFLAGS\_RELEASE = -O2 -MT  QMAKE\_CFLAGS\_RELEASE\_WITH\_DEBUGINFO += -O2 -MT -Zi  QMAKE\_CFLAGS\_DEBUG = -Zi -MTd |

10) Before Qt 5.2.0 the rcc.exe utility does not support long parameters name.

|  |
| --- |
| --name, --root, --compress, --threshold, --binary, --namespace, --version, --help, --no-compress, --project |

Amend the file src\tools\rcc\main.cpp. Add support of these parameters:

|  |
| --- |
| } else if (opt == QLatin1String("-name") || opt == QLatin1String("--name")) {  if (!(i < argc-1)) {  errorMsg = QLatin1String("Missing target name");  break;  }  library.setInitName(args[++i]);  } else if (opt == QLatin1String("-root") || opt == QLatin1String("--root")) {  if (!(i < argc-1)) {  errorMsg = QLatin1String("Missing root path");  break;  } |

11) Launch the Visual Studio Developer Command Prompt

|  |
| --- |
| "c:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\vcvarsall.bat" |

12) Configure

|  |
| --- |
| configure ^  -static ^  -debug-and-release ^  -prefix "d:\qt\5.1.0.0.a" ^  -platform win32-msvc2012 ^  -opensource ^  -confirm-license ^  -make libs ^  -nomake tools ^  -nomake examples ^  -nomake tests ^  -qt-freetype ^  -openssl-linked ^  OPENSSL\_LIBS="-lWs2\_32 -lGdi32 -lAdvapi32 -lCrypt32 -lUser32 -llibeay32 -lssleay32" ^  OPENSSL\_LIBS\_DEBUG="-llibeay32 -lssleay32" ^  OPENSSL\_LIBS\_RELEASE="-llibeay32 -lssleay32" ^  -I "d:\qt\openssl\openssl-1.0.1u-vs2012-static-x32\include" ^  -L "d:\qt\openssl\openssl-1.0.1u-vs2012-static-x32\lib" |

Since version 1.1.0 OpenSSL have changed their library names:

|  |
| --- |
| libeay32.dll -> libcrypto.dll  ssleay32.dll -> libssl.dll |

Please note this.

13) Build

|  |
| --- |
| nmake |

14) Install

|  |
| --- |
| nmake install |

15) Put the qt.conf file into bin directory. The contents of the qt.conf file are listed below.

|  |
| --- |
| [Paths]  Prefix = .. |

16) Copy OpenSSL libraries into target Qt library location:

|  |
| --- |
| xcopy d:\qt\openssl\openssl-1.0.1u-vs2012-static-x32\lib\\*.lib d:\qt\5.1.0.0.a\lib\ |

17) Using the substitute.py script change all absolute paths to the $$[QT\_INSTALL\_LIBS] macro in all \*.prl files.

These files are in lib directory.

|  |
| --- |
| d:\\qt\\5.1.0.0.a\\lib -> $$[QT\_INSTALL\_LIBS]  d:\\qt\\openssl\\openssl-1.0.1u-vs2012-static-x32\\lib -> $$[QT\_INSTALL\_LIBS] |

18) Build the example examples\network\download\download.pro.

|  |
| --- |
| cd "examples\network\download"  d:\qt\5.1.0.0.a\bin\qmake.exe  nmake release |

19) Try to use compiled application download.exe in order to test OpenSSL.

|  |
| --- |
| download.exe https://somedomain.com/somelocation/somefile.zip |

20) Use this command in order to obtain all absolute paths and unwanted static strings from compiled file.

|  |
| --- |
| grep -P d:\\\\ --color -a -i \*.exe |

21) Using the substitute.py script remove all absolute paths and unwanted static strings from libraries files.