# Using the Qt Cryptographic Architecture with Qt5

Alessandro Pasotti edited this page on Sep 24 · 22 revisions

This page contains notes regarding my experience of getting to know and finally using the Qt Cryptographic Architecture (QCA) with a Qt5 application.

The project website is very outdated, but the git repository is up to date and changes are still pushed regularly (as of Jan 2014).

## Building on Windows

The details in this section are applicable to a 64-bit Windows 7 machine with Qt 5.1.1 and MSVC 2012.

### Getting Started & Environment Setup

- We assume Qt 5.1.1 and MSVC 2012 is up and ready to use.
- Git clone it from the following repository: `git clone git://anongit.kde.org/qca.git`
- If you don't have CMake, install it from http://www.cmake.org/. I used CMAKE 2.8.12.
- If you want documentation to be generated which is recommended since the project website is outdated, install Doxygen before running CMake. The doxyfile generated will automatically enable dot graphs for which Graphviz is also required. Furthermore, Latex (I used Miktex distribution) and Ghostscript are required in order to generate the documentation.
- Also, depending on the plugins you want to use, install their dependencies. For the plugin provider of interest here (qca-ossl) the OpenSSL is enough. For Windows, download binaries here. Make sure to download the developer package. I used a package called Win64 OpenSSL v1.0.1f (16Mb). Make sure that you have the 64-bit Visual C++ 2008 Redistributables installed before you try to run the OpenSSL installer.

Clone this wiki locally

`https://github.com/JPNaude/`

Clone in Desktop

## Building QCA on Windows

- Run CMake on /CMakeLists.txt. I've used Qt Creator and set the build directory to be the same as the repository in the Qt Creator CMake Wizard. The CMake log will contain some important information:
  - It found Doxygen
  - Its being built with Qt5 support.
  - It found Open SSL

```
-- Found Doxygen: d:/tools/doxygen/bin/doxygen.exe (fo
-- Building with Qt5 support
-- Checking for certstore..
-- Using built in certstore.
-- certstore path: D:/work/software/public/qca/certs/r
-- mlock(2) does not take a void *
-- Found OpenSSL: optimized;d:/tools/OpenSSL-Win64/lib
-- Looking for EVP_md2
-- Looking for EVP_md2 - not found
-- Looking for EVP_aes_128_ctr
-- Looking for EVP_aes_128_ctr - not found
-- Configuring done
-- Generating done
-- Build files have been written to: D:/work/software/
```

- Next run `make`.
- The dlls were not all placed in the expected /bin folder:
  - qca(d).dll is placed in /src
  - qca-ossl(d).dll is placed in /lib/qca/crypto
- To generate the documentation, I had to edit the generated doxyfile.ini by replacing @CMAKE_SOURCE_DIR@ with the path to the QCA repository. To start generation, just run `doxygen doxyfile.in`.
- To test if the build was successful, run the `qcatool.exe` in /bin. Running `qcatool plugins` will provide information about available providers:

```
Qt Library Paths:
  D:/tools/qt/Qt5.1.0/5.1.0/msvc2012_64/plugins
  D:/work/software/public/qca/bin
Available Providers:
  (none)
```

- The first go indicates that no providers exists. To get some debug information, add the "--debug" switch to the qcatool command. It indicated us that no "crypto" directory exists. Therefore, copy the /build_dir/lib/qca/crypto directory into /build_dir/bin, and make sure your qca(d).dll file is either in the path, or copy it to the same directory. ALSO, make sure the /bin directory of your OpenSSL installation is in your system's PATH environment variable. Run the command again, this time it gives:

```
Qt Library Paths:
  D:/tools/qt/Qt5.1.0/5.1.0/msvc2012_64/plugins
  D:/work/software/public/qca/bin
Available Providers:
  qca-logger
  qca-ossl
    This product includes cryptographic software writt
    (eay@cryptsoft.com)
  qca-softstore
```

- By default, QCA and plugins are built in debug mode. To build in release mode, remove the CMakeCache.txt and add "-DCMAKE_BUILD_TYPE=Release" to the arguments in the Qt Creator CMake Wizard.

## Building on Linux (Ubuntu)

The details in this section are applicable to a 64-bit Ubuntu 12.04 machine with Qt 5.1.0 and GCC v4.6.3.

### Getting Started & Environment Setup

- We assume Qt 5.1.0 and GCC is up and ready to use.
- Install CMake (not at least v2.8.8 is needed to support Qt5, thus for me `sudo apt-get install cmake` did not work yet since that is still v2.8.7).
  - Get the latest install package. For example: `wget http://www.cmake.org/files/v2.8/cmake-2.8.12.1-Linux-i386.sh`
- Check that you have OpenSSL dev libraries install (libssl-dev): `apt-cache search libssl | grep SSL`

```
$ apt-cache search libssl | grep ssl
libssl-dev - SSL development libraries, header files a
libssl-doc - SSL development documentation documentati
libssl1.0.0 - SSL shared libraries
```

If its not installed, see this page for details on how to install it.

- Git clone QCA from the following repository: `git clone git://anongit.kde.org/qca.git
- If you are interested in generating the QCA documentation, see the notes on what is required in the "Building on Windows" section.

## Building QCA on Linux

The steps are (from \INSTALL):

```
cmake .
make
make install
/sbin/ldconfig, if necessary
```

- Run CMake on /CMakeLists.txt: `cmake .`

The first go indicates that Qt could not be found:

```
~/qt/qca$ cmake .
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- wor
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Could NOT find Doxygen (missing:  DOXYGEN_EXECUTABL
CMake Error at /usr/share/cmake-2.8/Modules/FindPackag
  Could NOT find Qt4 (missing: QT_QMAKE_EXECUTABLE QT_
  QT_RCC_EXECUTABLE QT_INCLUDE_DIR QT_LIBRARY_DIR QT_Q
  QT_QTCORE_LIBRARY QT_QTNETWORK_INCLUDE_DIR QT_QTNETW
  QT_QTTEST_INCLUDE_DIR QT_QTTEST_LIBRARY) (Required i
  "4.7.0")
```

To fix this, I had to add the following line right to the top of the CMakeLists.txt file in the QCA folder:

```
set (CMAKE_PREFIX_PATH "/opt/Qt/5.1.0/gcc_64")
```

Where the path pointed to my Qt5 install.

- If libnss2 is was not found on your system, it will also print the following:

```
.
.
-- PKGCONFIG() indicates that nss is not installed (in
.
.
```

To fix that, do the following: `sudo apt-get install libnss3-dev`

- Run cmake again. The CMake log will now contain some important information:
  - Its being built with Qt5 support.
  - It found Open SSL.

```
-- Building with Qt5 support
.
.
.
-- Found OpenSSL: /usr/lib/x86_64-linux-gnu/libssl.so;
.
.
.
etc.
```

- Next, run `make`.
- After it built, cd into /bin and run `./qcatool version`. You should see something like this:

```
qcatool version 2.1.0 by Justin Karneges
Using QCA version 2.1.0
```

- Next, run: `./qcatool plugins`. You should see something like this:

```
Qt Library Paths:
  /opt/Qt/5.1.0/gcc_64/plugins
  /home/plunify/qt/qca/bin
Available Providers:
  (none)
```

- To fix the (none) under providers, we need to copy the /lib/qca/crypto directory into /bin. Thus, from within /bin use `cp ../lib/qca/crypto/ . -r`.
- Run `./qcatool plugins` again:

```
Qt Library Paths:
  /opt/Qt/5.1.0/gcc_64/plugins
  /home/plunify/qt/qca/bin
Available Providers:
  qca-gcrypt
  qca-gnupg
  qca-logger
  qca-ossl
    This product includes cryptographic software writt
    (eay@cryptsoft.com)
  qca-softstore
```

Notice that the provider plugins are now loaded, most importantly the qca-ossl provider.