QEverCloud

Generated by Doxygen 1.9.4

Chapter 1

QEverCloud

Unofficial Evernote Cloud API for Qt

1.1 What's this

This library presents the complete Evernote SDK for Qt. All the functionality that is described on Evernote site is implemented and ready to use. In particular OAuth authentication is implemented.

Read doxygen generated documentation for detailed info.

The documentation can also be generated in the form of a .qch file which you can register with your copy of Qt Creator to have context-sensitive help. See below for more details.

1.2 How to contribute

See contribution guide for detailed info.

1.3 Downloads

Prebuilt versions of the library can be downloaded from the following locations:

- Stable version:
  - Windows binaries:
    - MSVC 2019 32 bit Qt 5.15.2
    - MSVC 2019 64 bit Qt 5.15.2
  - Mac binary (built with Qt 5.15.2)
  - Linux binary built on Ubuntu 20.04 with Qt 5.12.8
- Unstable version:
  - Windows binaries:
    - MSVC 2019 32 bit Qt 5.15.2
    - MSVC 2019 64 bit Qt 5.15.2
  - Mac binary (built with Qt 5.15.2)
  - Linux binary built on Ubuntu 20.04 with Qt 5.12.8

1.4 How to build

QEverCloud uses CMake build system which can be used as simply as follows (on Unix platforms):

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=<...> ../
make
make install
```

The library can be built and shipped either as a static library or a shared library. Dll export/import symbols necessary for Windows platform are supported.

QEverCloud uses C++14 standard. CMake automatically checks whether the compiler is capable enough of building QEverCloud so if the pre-build configuration step was successful, the build step should be successful as well. Known capable compilers are g++ 9 or later and Visual Studio 2019 or later.

The recommended version of Qt5 for building the library is the latest Qt 5.15. However, it is also known to build and work with Qt 5.12.

QEverCloud depends on the following Qt components:

- Qt5Core
- Qt5Widgets
- Qt5Network
- (Optional) Qt5WebKit an Qt5WebKitWidgets
- (Optional) Qt5WebEngine and Qt5WebEngineWidgets

The dependencies on Qt5WebKit or Qt5WebEngine are only actual if the library is built with OAuth support. But even then there is an option to build the library with OAuth support but without the dependency on either of these components. More on this below.

By default the library is built with OAuth support and uses Qt5WebEngine for it. The following cmake parameters are available to alter this behaviour:

- `-DBUILD_WITH_OAUTH_SUPPORT=NO` would disable building with OAuth support entirely.
- `-DUSE_QT5_WEBKIT=ON` would build the library with OAuth using Qt5WebKit for web page rendering.
- `-DQEVERCLOUD_USE_SYSTEM_BROWSER=ON` would build the library with OAuth not using either Qt5WebKit or Qt5WebEngine but instead delegating some portion of OAuth procedure to the system browser.

If Qt5's Qt5Test module is found during the pre-build configuration step, the unit tests are enabled and can be run with `make test` and more verbose `make check` commands.

Other available CMake configurations options:

BUILD_DOCUMENTATION - when *ON*, attempts to find Doxygen and in case of success adds *doc* target so the documentation can be built using `make doc` command after the pre-build configuration step. By default this option is on.

BUILD_QCH_DOCUMENTATION - when *ON*, passes instructions on to Doxygen to build the documentation in *qch* format. This option only has any meaning if *BUILD_DOCUMENTATION* option is on. By default this option is off.

BUILD_SHARED - when *ON*, CMake configures the build for the shared library. By default this option is on.

BUILD_WITH_Q_NAMESPACE - when *ON*, `Q_NAMESPACE` and `Q_ENUM_NS` macros are used to add introspection capabilities to enumerations within `qevercloud` namespace. Qt >= 5.8 is required to enable this option. By default this option is enabled.

BUILD_TRANSLATIONS - when *ON*, builds and installs translation files for translatable strings from QEverCloud.

If *BUILD_SHARED* is *ON*, `make install` installs CMake module necessary for applications using CMake's `find_package` command to find the installation of QEverCloud.

It is possible to build the library with enabled sanitizers using additional CMake options:

- `-DSANITIZE_ADDRESS=ON` to enable address sanitizer
- `-DSANITIZE_MEMORY=ON` to enable memory sanitizer
- `-DSANITIZE_THREAD=ON` to enable thread sanitizer
- `-DSANITIZE_UNDEFINED=ON` to enable undefined behaviour sanitizer

1.5 Include files for applications using the library

Two "cumulative" headers - *QEverCloud.h* or *QEverCloudOAuth.h* - include everything needed for the general and OAuth functionality correspondingly. More "fine-grained" headers can also be used if needed.

1.6 Seeding random numbers generator for Qt < 5.10

QEverCloud requires random numbers generator for OAuth procedure. When QEverCloud is built against Qt >= 5.10, it uses `QRandomGenerator` which is cryptographically secure on supported platforms and is seeded by Qt internals. With Qt < 5.10 QEverCloud uses `qrand`. It requires the client application to call `qsrand` with seed value before using OAuth calls of QEverCloud. So if you are using QEverCloud built with Qt < 5.10, make sure to call `qsrand` before using QEverCloud's OAuth.

1.7 Related projects

- `QEverCloudGenerator` repository hosts code generating parser of `Evernote Thrift IDL files`. This parser is used to autogenerate a portion of QEverCloud's headers and sources.
- `libquentier` is a library for creating feature rich full sync Evernote clients built on top of QEverCloud
- `Quentier` is an open source desktop note taking app capable of working as Evernote client built on top of libquentier and QEverCloud