

This directory contains the source code for the Bitcoin Core graphical user interface (GUI). It uses the Qt cross-platform framework.

The current precise version for Qt 5 is specified in qt.mk.

## Compile and run

See build instructions: Unix, macOS, Windows, FreeBSD, NetBSD, OpenBSD

When following your systems build instructions, make sure to install the Qt dependencies.

To run:

```
./src/qt/bitcoin-qt
```

## Files and Directories

### forms/

- A directory that contains Designer UI files. These files specify the characteristics of form elements in XML. Qt UI files can be edited with Qt Creator or using any text editor.

### locale/

- Contains translations. They are periodically updated and an effort is made to support as many languages as possible. The process of contributing translations is described in doc/translation\_process.md.

### res/

- Contains graphical resources used to enhance the UI experience.

### test/

- Functional tests used to ensure proper functionality of the GUI. Significant changes to the GUI code normally require new or updated tests.

### bitcoingui.(h/cpp)

- Represents the main window of the Bitcoin UI.

### \*model.(h/cpp)

- The model. When it has a corresponding controller, it generally inherits from QAbstractTableModel. Models that are used by controllers as helpers inherit from other Qt classes like QValidator.
- ClientModel is used by the main application `bitcoingui` and several models like `peertablemodel`.

### **\*page.(h/cpp)**

- A controller. `:NAMEpage.cpp` generally includes `:NAMEmodel.h` and `forms/:NAME.page.ui` with a similar `:NAME`.

### **\*dialog.(h/cpp)**

- Various dialogs, e.g. to open a URL. Inherit from `QDialog`.

### **paymentserver.(h/cpp)**

- (Deprecated) Used to process BIP21 payment URI requests. Also handles URI-based application switching (e.g. when following a `bitcoin:... link` from a browser).

### **walletview.(h/cpp)**

- Represents the view to a single wallet.

### **Other .h/cpp files**

- UI elements like `BitcoinAmountField`, which inherit from `QWidget`.
- `bitcoinstrings.cpp`: automatically generated
- `bitcoinunits.(h/cpp)`: BTC / mBTC / etc. handling
- `callback.h`
- `guiconstants.h`: UI colors, app name, etc.
- `guiutil.h`: several helper functions
- `macdockiconhandler.(h/mm)`: macOS dock icon handler
- `macnotificationhandler.(h/mm)`: display notifications in macOS

## **Contribute**

See `CONTRIBUTING.md` for general guidelines.

**Note:** Do not change `local/bitcoin_en.ts`. It is updated automatically.

## **Using Qt Creator as an IDE**

Qt Creator is a powerful tool which packages a UI designer tool (Qt Designer) and a C++ IDE into one application. This is especially useful if you want to change the UI layout.

**Download Qt Creator** On Unix and macOS, Qt Creator can be installed through your package manager. Alternatively, you can download a binary from the Qt Website.

**Note:** If installing from a binary grabbed from the Qt Website: During the installation process, uncheck everything except for **Qt Creator**.

## macOS

```
brew install qt-creator
```

## Ubuntu & Debian

```
sudo apt-get install qtcreator
```

## Setup Qt Creator

1. Make sure you've installed all dependencies specified in your systems build instructions
2. Follow the compile instructions for your system, run `./configure` with the `--enable-debug` flag
3. Start Qt Creator. At the start page, do: **New -> Import Project -> Import Existing Project**
4. Enter `bitcoin-qt` as the Project Name and enter the absolute path to `src/qt` as Location
5. Check over the file selection, you may need to select the `forms` directory (necessary if you intend to edit `*.ui` files)
6. Confirm the **Summary** page
7. In the **Projects** tab, select **Manage Kits...**

**macOS** - Under **Kits**: select the default "Desktop" kit - Under **Compilers**: select "**Clang (x86 64bit in /usr/bin)**" - Under **Debuggers**: select "**LLDB**" as debugger (you might need to set the path to your LLDB installation)

## Ubuntu & Debian

Note: Some of these options may already be set

- Under **Kits**: select the default "Desktop" kit
  - Under **Compilers**: select "**GCC (x86 64bit in /usr/bin)**"
  - Under **Debuggers**: select "**GDB**" as debugger
8. While in the **Projects** tab, ensure that you have the `bitcoin-qt` executable specified under **Run**
  - If the executable is not specified: click "**Choose...**", navigate to `src/qt`, and select `bitcoin-qt`
  9. You're all set! Start developing, building, and debugging the Bitcoin Core GUI