# MAPS.ME Development

## Issues

The simplest way to contribute is to [submit an issue](https://github.com/mapsme/omim/issues).

Please give developers as much information as possible: OS and application versions,

list of actions leading to a bug, a log file produced by the app.

When using the MAPS.ME app on a device, use the built-in "Report a bug" option:

it creates a new e-mail with a log file attached. Your issue will be processed much

faster if you send it to bugs@maps.me.

## Initializing the Repository

The repository needs a lot of header and configuration files with private keys and such.

To initialize, run `configure.sh` from its root, and press "Enter" when asked for a

repository. The script will create two files, `private.h` and `android/secure.properties`,

which are required for compiling the project. If you have a private repository with

these files (and some other, like private keystores), pass the link like this:

echo git@repository:org/omim-private.git | ./configure.sh

Without keys, downloading of maps won't work, as well as statistics and online routing

assisting. For android, you would need private keys to build a release version (but

a debug one can be built without keys).

## Setting up IDE

See [INSTALL.md](INSTALL.md) for command-line compilation instructions.

\* Install XCode and Command Line Tools, then run XCode and click "I Agree".

### Qt Creator

\* Download [Qt Creator](http://www.qt.io/download-open-source/) with Qt 5.

\* In `Qt/5.5/clang\_64/mkspecs/qdevice.pri` replace `10.10` with `10.11`, and

add a line `QMAKE\_MAC\_SDK = macosx10.11` (see installed versions in

`/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/`)

\* Start Qt Creator, open `omim.pro`, choose "qt" target and run the project.

\* To build the project faster, open "Project Settings", find "Build Steps", then

"Make Arguments" and put "-j8" there (without quotes).

Debugging may not work in Qt Creator. To enable it, try creating a Build & Run kit in

preferences with XCode's lldb as a debugger and a clang compiler.

At the moment configuration assumes you've cloned repository into omim (also a default name) directory.

If you are using shadow-dir for building artifacts (default behavior) and custom directory for repo -

you'll need to create a "data" symlink in the shadow-dir path to the /repo/data directory.

### XCode

\* Install [Homebrew](http://brew.sh/) and run `brew install qt5`.

\* Run XCode, open `xcode/omim.xcworkspace`.

\* Select "xcMAPS.ME" scheme and run the product.

## Coding Style

See [CPP\_STYLE.md](CPP\_STYLE.md). Use `clang-format` when in doubt.

## Pull Requests

All contributions to MAPS.ME source code should be submitted via github pull requests.

Each pull request is reviewed by MAPS.ME employees, to ensure consistent code style

and quality. Sometimes the review process even for smallest commits can be

very thorough.

To contribute you must sign the [license agreement](CLA.md): the same one you

sign for Google or Facebook open-source projects.

## Directories

### Core

\* `api` - external API of the application.

\* `base` - some base things, like macros, logging, caches etc.

\* `coding` - I/O classes and data processing.

\* `drape` - the new graphics library core.

\* `drape\_frontend` - scene and resource manager for the Drape library.

\* `generator` - map building tool.

\* `geocoder` -

\* `geometry` - geometry primitives we use.

\* `indexer` - processor for map files, classificator, styles.

\* `map` - app business logic, including a scene manager.

\* `platform` - platform abstraction classes: file paths, http requests, location services.

\* `routing` - in-app routing engine.

\* `routing\_common` -

\* `search` - ranking and searching classes.

\* `std` - standard headers wrappers, for Boost, STL, C-rt.

\* `storage` - map reading function.

\* `tracking` -

\* `traffic` - real-time traffic information.

\* `transit` -

\* `ugc` - user generated content, such as reviews.

### Other

Some of these contain their own README files.

\* `3party` - external libraries, sometimes modified.

\* `android` - Android UI.

\* `cmake` - CMake helper files.

\* `data` - data files for the application: maps, styles, country borders.

\* `debian` - package sources for Debian.

\* `descriptions` -

\* `editor` -

\* `feature\_list` -

\* `installer` - long-abandoned installer for Windows.

\* `iphone` - iOS UI.

\* `kml` - manipulation of KML files.

\* `local\_ads` -

\* `mapshot` - generate screenshots of maps, specified by coordinates and zoom level.

\* `metrics` -

\* `openlr` -

\* `partners\_api` - API for partners of the MAPS.ME project.

\* `pyhelpers` -

\* `qt` - desktop application.

\* `qt\_tstfrm` - widgets for visual testing.

\* `shaders` - shaders for rendering.

\* `skin\_generator` - a console app for building skin files with icons and symbols.

\* `software\_renderer` -

\* `stats` - Alohalytics statistics.

\* `testing` - common interfaces for tests.

\* `tools` - tools for building packages and maps, for testing etc.

\* `track\_analyzing` -

\* `track\_generator` - Generate smooth tracks based on waypoints from KML.

\* `xcode` - XCode workspace.

## Questions?

For any questions about developing MAPS.ME and relevant services - virtually about anything related,

please write to us at bugs@maps.me, we'll be happy to help.