**TDLib**

**Telegram Database Library**

**TDLib** (Telegram Database Library) is a cross-platform, fully functional Telegram client. We designed it to help third-party developers create their own **custom apps** using the Telegram platform.

* **Cross-platform**. TDLib can be used on Android, iOS, Windows, macOS, Linux, WebAssembly, FreeBSD, Windows Phone, watchOS, tvOS, Tizen, Cygwin. It should also work on other \*nix systems with or without minimal effort.
* **Multilanguage**. TDLib can be easily used with **any programming language** that is able to execute C functions. Additionally it already has native bindings to Java (using JNI) and C# (using C++/CLI).
* **Easy to use**. TDLib takes care of all **network implementation details**, **encryption** and **local data storage**.
* **High-performance**. In the Telegram Bot API, each TDLib instance handles more than **24,000** active bots simultaneously.
* **Well-documented**. All TDLib API methods and public interfaces are [**fully documented**](https://core.telegram.org/tdlib/docs/).
* **Consistent**. TDLib guarantees that all updates will be delivered in the **right order**.
* **Reliable**. TDLib remains **stable** on slow and unreliable Internet connections.
* **Secure**: All local data is **encrypted** using a user-provided encryption key.
* **Fully-asynchronous**. Requests to TDLib don't block each other or anything else, responses will be sent when they are available.

TDLib is fully **open source**, all code is available on [GitHub](https://github.com/tdlib/td).

**TDLib** is a fully functional Telegram client which takes care of all networking, local storage and data consistency details. In this tutorial we describe the main concepts understanding of which is required for effecient TDLib usage.

**Dependencies**

* C++14 compatible compiler (Clang 3.4+, GCC 4.9+, MSVC 19.0+ (Visual Studio 2015+), Intel C++ Compiler 17+)
* OpenSSL
* zlib
* gperf (build only)
* CMake (3.0.2+, build only)
* PHP (optional, for documentation generation)

**Building**

The simplest way to build TDLib is to use our [TDLib build instructions generator](https://tdlib.github.io/td/build.html). You need only to choose your programming language and target operating system to receive complete build instructions.

In general, you need to install all TDLib [dependencies](https://core.telegram.org/tdlib/docs/index.html#dependencies) as described in [Installing dependencies](https://core.telegram.org/tdlib/docs/index.html#installing-dependencies). Then enter directory containing TDLib sources and compile them using CMake:

mkdir build

cd build`

cmake -DCMAKE\_BUILD\_TYPE=Release ..

cmake --build .

To build TDLib on low memory devices you can run [SplitSource.php](https://github.com/tdlib/td/blob/master/SplitSource.php) script before compiling main TDLib source code and compile only needed targets:

mkdir build

cd build

cmake -DCMAKE\_BUILD\_TYPE=Release ..

cmake --build . --target prepare\_cross\_compiling

cd ..

php SplitSource.php

cd build

cmake --build . --target tdjson

cmake --build . --target tdjson\_static

cd ..

php SplitSource.php --undo

In our tests clang 6.0 with libc++ required less than 500 MB of RAM per file and GCC 4.9/6.3 used less than 1 GB of RAM per file.

**Installing dependencies**

**macOS**

* Install the latest Xcode command line tools, for example, via xcode-select --install.
* Install other [dependencies](https://core.telegram.org/tdlib/docs/index.html#dependencies), for example, using [Homebrew](https://brew.sh/):

brew install gperf cmake openssl

* Build TDLib with CMake as explained in [building](https://core.telegram.org/tdlib/docs/index.html#building). You will likely need to manually specify path to the installed OpenSSL to CMake, e.g.,

cmake -DCMAKE\_BUILD\_TYPE=Release -DOPENSSL\_ROOT\_DIR=/usr/local/opt/openssl/ ..

**Windows**

* Download and install Microsoft Visual Studio 2015 or later.
* Download and install [gperf](https://sourceforge.net/projects/gnuwin32/files/gperf/3.0.1/). Add the path to gperf.exe to the PATH environment variable.
* Install [vcpkg](https://github.com/Microsoft/vcpkg#quick-start).
* Run the following commands to install TDLib dependencies using vcpkg:

cd <path to vcpkg>

.\vcpkg.exe install openssl:x64-windows openssl:x86-windows zlib:x64-windows zlib:x86-windows

* Download and install [CMake](https://cmake.org/download/); choose "Add CMake to the system PATH" option while installing.
* Build TDLib with CMake as explained in [building](https://core.telegram.org/tdlib/docs/index.html#building), but instead of cmake -DCMAKE\_BUILD\_TYPE=Release .. use

cmake -DCMAKE\_TOOLCHAIN\_FILE=<path to vcpkg>/scripts/buildsystems/vcpkg.cmake ..

To build 32-bit/64-bit TDLib using MSVC, you will need to additionally specify parameter -A Win32/-A x64 to CMake. To build TDLib in Release mode using MSVC, you will need to additionally specify parameter --config Release to the cmake --build . command.

**Linux**

* Install all [dependencies](https://core.telegram.org/tdlib/docs/index.html#dependencies) using your package manager.