Telegram Desktop User Guide

Introduction

This document will explain how to build Telegram Desktop locally and run it with the new features we have implemented. We have forked the original repository from GitHub and built new features into this forked repository which is open for public viewing.

Build Instructions

Instructions for building Telegram Desktop can be found at the bottom of the README of the forked repo. The instructions contain steps to build the application on Windows, macOS and GNU/Linux using Docker. The initial setup for building the application will take about an hour, so be sure to reserve enough time for this process. We have successfully built the application on Windows 64-bit and macOS.

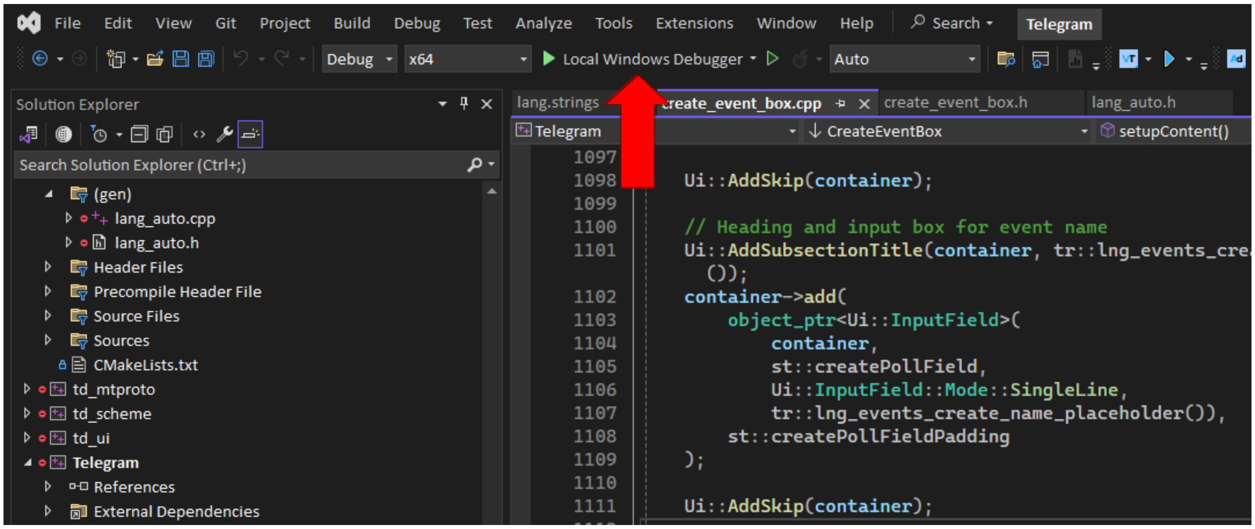
When configuring API credentials, use the test credentials provided in the documentation. The credentials as of writing this user guide is as follows:

api_id: 17349
api_hash: 344583e45741c457fe1862106095a5eb

Be sure to build the application using the **dev** branch. This is where our new features have been implemented. View our demo video to see these features in action.

Windows 64-bit

To build and run the application on Windows 64-bit, follow the instructions listed in the README. Windows 64-bit uses Visual Studio as the IDE. Before running `tdesktop\Telegram\build\prepare\win.bat` in the command prompt, be sure to install the latest version of Graphviz. This is an issue we ran into and reported on the original repo. Note that installing Qt Visual Studio Tools is an optional step. Once you have followed all the steps in the README, you can open the resulting Telegram.exe file or click the Local Windows Debugger built into Visual Studio to start the application. See Figure 1 below for the location of the debug button.

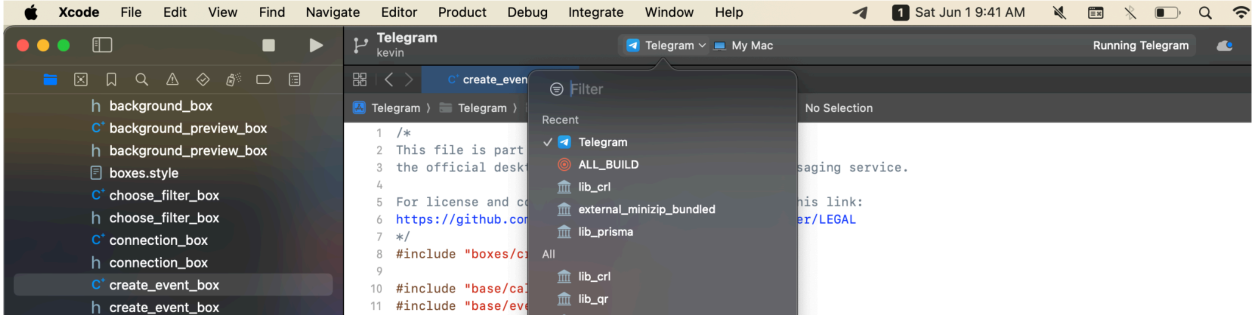


```
Ui::AddSkip(container);

// Heading and input box for event name
Ui::AddSubsectionTitle(container, tr::lng_events_cre
    ());
container->add(
    object_ptr<Ui::InputField>(
        container,
        st::createPollField,
        Ui::InputField::Mode::SingleLine,
        tr::lng_events_create_name_placeholder()),
    st::createPollFieldPadding
);

Ui::AddSkip(container);
```

Figure 1: Location of the Local Windows Debugger is shown with the red arrow.

macOS

To build and run the application on macOS, follow the instructions listed in the README. macOS uses Xcode as the IDE. When you build the project in Xcode you will need to make sure that you select the Telegram target to build everything. Then you can click on the “play” button to build.



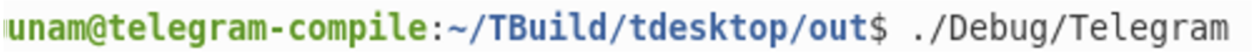
```
/*
This file is part
the official deskt                                   saging service.

For license and c                                    his link:
https://github.co                                    er/LEGAL
*/
#include "boxes/c

#include "base/ca
#include "base/ev
```

Figure 2: Location of drop down for selecting a build target.

Linux

To build and run the application on any Linux distribution, follow these instructions. However, before getting started with the instructions manual, make sure to install docker, pipenv and a global installation of poetry. It is also advisable to have the docker configured to run without the need for superuser (‘sudo’) to safeguard against undesired privilege escalation.

After building the Telegram app, it can be run using the executable found in the BuildPath/tdesktop/out/Debug folder, as shown in Figure 3 below.



```
unam@telegram-compile:~/TBuild/tdesktop/out$ ./Debug/Telegram
```

Figure 3: Command to run the Telegram executable.