Work is ongoing to extend Flutter to support desktop as a target environment, allowing developers to create Windows, macOS, and Linux applications with Flutter.

For instructions on using Flutter to build desktop applications, including how to set up your development environment, see the flutter.dev documentation

Current Status

A high-level overview of the status of each platform is provided below. For details see the list of desktop-related bugs, as well as the embedding source.

Windows

The Windows embedding is in beta. It is Win32-based, a UWP version is in development.

macOS

The macOS embedding is in beta.

Linux

The Linux embedding is in beta. It is GTK-based; ideally we would like to create a library that allows embedding Flutter regardless of whether you're using GTK, Qt, wxWidgets, Motif, or another arbitrary toolkit for other parts of your application, but have not yet determined a good way to do that. Support for other toolkits may be added in the future.

Plugins

Writing plugins is supported on all platforms, however not all plugins have desktop support. See the flutter.dev documentation for more details.

Add-to-App

Currently there is no support for Add-to-App for any desktop platform. If you are familiar with doing native development on your platform(s), it is possible to integrate the desktop Flutter libraries in your own app. There is not currently much guidance, so you will be well off the beaten path, but the information below will help get you started.

Getting the Libraries

Unless you want to build the Flutter engine from source, you will need a prebuilt library. The easiest way to get the right version for your version of Flutter is to run `flutter precache` with the `--linux`, `--macos`, or `--windows` flag (depending

on your platform). They will be downloaded to `bin/cache/artifacts/engine/` under your Flutter tree.

C++ Wrapper

The Windows library provides a C API. To make it easier to use, there is a C++ wrapper available which you can build into your application to provide a higher-level API surface. The source for it is downloaded with the library. You will need to build it as part of your application.

Documentation

See the headers that come with the library (or wrapper) for your platform for information on using them. It may be helpful to look at the basic application produced by `flutter create` to see how it calls the APIs.

Building

In addition to linking the Flutter library, your application will need to bundle your Flutter assets (as created by `flutter build bundle`). On Windows and Linux you will also need the ICU data from the Flutter engine (look for `icudtl.dat` under the `bin/cache/artifacts/engine` directory in your Flutter tree).