The Flutter Engine is window toolkit agnostic. If you want to build Flutter embedders on one of the platforms not supported out of the box (i.e, iOS & Android), this page is for you.

This is a very low level API and is not suitable for beginners.

- The window toolkit agnostic component of the Flutter engine is available as a dynamic library in the `flutter_engine` GN target in `//shell/platform/embedder:flutter_engine` .
- That target must be built as part of the host GN build. Such build are already available for desktop Linux & Mac. If you want to target another platform, you will have to configure a GN toolchain for the same.
- You may build this target yourself or download the same artifacts uploaded by the buildbots on each commit.
  - The Mac buildbot uploads the artifacts to a known location. Access it here `https://storage.googleapis.com/flutter_infra_release/flutter/FLUTTER_ENGINE/darwin-x64/FlutterEmbedder.framework.zip` .
    - Replace `FLUTTER_ENGINE` with the SHA of the Flutter engine you wish to use.
  - The Linux buildbot uploads the artifacts to a known location. Access it here `https://storage.googleapis.com/flutter_infra_release/flutter/FLUTTER_ENGINE/linux-x64/linux-x64-embedder`
    - Replace `FLUTTER_ENGINE` with the SHA of the Flutter engine you wish to use.
    - The binary is not stripped and contains debug information. Embedders are advised to strip the binary before deployment.
  - The Windows buildbot uploads the artifacts to a known location. Access it here `https://storage.googleapis.com/flutter_infra_release/flutter/FLUTTER_ENGINE/windows-x64/windows-x64-embedder.zip`
    - Replace `FLUTTER_ENGINE` with the SHA of the Flutter engine you wish to use.
  - You can also obtain that SHA from the `engine.version` file in your Flutter framework checkout. This allows you to exactly match the engine version with the Flutter framework version.
- The Flutter engine API has no platform specific dependencies, has a stable ABI and is available in its entirety in a single C header file available here.
- To use as a guide, you may use this example embedder that uses GLFW for window management and rendering.

While we do not object to teams creating custom builds of the Flutter engine for their purposes, we do not support this configuration. Not supporting it means that we do not commit to any timelines for fixing bugs that may come up in such a configuration, even for customers for which we would usually be willing to make commitments (see the [[Issue Hygiene]] page). It also means that we encourage teams to view such configurations as short-term solutions only and encourage teams to transition away from such configurations at the earliest possible opportunity.

We do not expect custom engine builds to be long-term sustainable. They are not supported on any platform where we plan to be the publisher of a Flutter runtime distinct from the applications that run on the runtime, and they require significant effort to port to our new target platforms such as Web and desktop. There is also an expensive maintenance burden (for example, if we add new features, a custom engine build would need to be updated to support that feature).

We would generally recommend using custom engine builds only when porting Flutter to platforms that are not supported out of the box, for example in embedded hardware.