Building and Distributing the Flutter Preview Device

fujino@
Initially written 11/9/23
go/flutter-preview-device-distribution

Summary

With the majority of the engineering work for the Flutter Preview device finished (#130277), the last outstanding task is to actually build the artifact on CI and make it available to end users. This document outlines our options for when and how to build and publish the artifact, and for how the tool will cache it.

~~The recommended option is to build the artifact from the engine tree only on release branches, uploading it the same way the other engine artifacts are, which will not require setting up new cloud storage infrastructure but also ensure users who flutter upgrade or delete their cache can re-download the artifact.~~

Updated: the recommended option is to build the device during packaging, but also upload it to cloud storage independently, namespaced by framework commit.

Background

The design doc for this project is at The 5 Minute Quick Start. TL;DR the Flutter Preview device is a pre-built debug Flutter desktop application that will be distributed with the Flutter SDK, and allow users to experience the hot reload workflow *without* needing to install a native build toolchain.

For the initial minimum viable product, the preview will only be available for Windows, but if that is successful macOS and Linux would be added.

The code to build the preview device is simply a hidden flutter tool sub-command: `flutter build _preview` (//flutter_tools/lib/src/commands/build_preview.dart). This command essentially just does:

```
flutter create flutter_preview
cd flutter_preview
flutter build -debug windows
cp build/windows/x64/runner/Debug/flutter_preview.exe
```

```
$FLUTTER_ROOT/bin/cache/artifacts/flutter_preview
```

This code is tested on every commit in //flutter_tools/test/integration.shard/build_preview_test.dart. This integration test builds the device and verifies it, however it does *not* upload this artifact for the reasons outlined in the cons of option 2 below.

Options

Option 1: Build and publish from the engine tree ~~(preferred)~~

On release candidate branches, add a new step to the windows host engine (which includes desktop engine artifacts) builder phase of the engine repo CI, that would clone the framework from the matching release candidate branch, and the preview device would be built with the local engine artifacts. They would then be uploaded like any other engine artifact.

The tool would need to be updated to download the artifact from cloud storage (currently it will use the artifact if it is present in the cache, else treat it as unavailable) if and only if the user is on a release branch.

Pros:

1. Users with existing Flutter SDK installations who get a new version via `flutter upgrade` can use the corresponding build of the preview device.
2. Users who delete their Flutter cache (because a stack overflow post suggested it as a bug workaround) will be able to re-download the preview device.

Cons:

1. Need to add Flutter preview-specific code to the engine repo
2. Added complexity on the engine CI side, as the script to build the preview device would need to clone the tool code from the framework's corresponding release candidate branch, and then use local engine artifacts from the current engine commit

Option 2: Build and publish within the release packaging phase

While packaging the Flutter SDK zip archives that are distributed through flutter.dev, we run the command `flutter build _preview`, so that the Flutter cache that is distributed to users includes the preview device. The tool will only list the preview device as being available if it is actually present on disk in the cache. There is no mechanism for downloading it (the user could theoretically run `flutter build _preview` themselves, but this would require the full desktop toolchain, defeating the purpose of the preview device).

Pros:

1. This is the simplest solution, just requiring a few lines being added to https://github.com/flutter/flutter/blob/master/dev/bots/prepare_package.dart#L527

Cons:

1. If the user changes their Flutter version, they can no longer use the preview device
2. If the user deletes their Flutter cache, they can no longer use the preview device

Option 3: Build and publish from the framework tree

During framework post-submit CI, we build and upload the artifact to cloud storage, namespaced by the *framework* commit. The tool would be updated to download the binary from cloud storage.

Pros:

1. The entire implementation of the Flutter Preview feature would be in a single repository

Cons:

1. We would need to implement new infrastructure publishing artifacts from the Flutter repo
2. We would need to ensure that packaging builds for releases do not begin *until* publishing the preview artifact was finished (today, the existence of the target framework commit is enough to signal packaging *can* begin).

Option 4: Build within the packaging phase, publish separately from the package (preferred)

This is like option 2, except instead of solely relying on user's getting the preview device already in their cache by downloading the SDK from their website, we would, from the packaging script, publish the preview device to cloud storage.

Pros:

1. All the pros from Option 1
2. None of the cons from Option 2

Cons:

1. We would need to figure out where to upload the preview device, as there is no precedent for uploading artifacts associated with a framework commit.

The release package zips are namespaced like: `https://storage.googleapis.com/flutter_infra_release/releases/$CHANNEL/$PLATFORM/flutter_$PLATFORM_$TAG-$CHANNEL.tar.xz` (source). While we *could* reconstruct this path from the tool by inspecting the git branch and the git tag, this would be brittle and easily broken by anomalies in the git state from release bugs.

I would recommend creating a new directory in cloud storage for storing artifacts namespaced by framework commit (which can be more reliably identified by the tool), such as `https://storage.googleapis.com/flutter_infra_release/framework_artifacts/$REVISION/flutter_preview.zip`.