Flutter is a series of connected repositories. Within each repository there are files (normally named `DEPS`) which determine which version of the other repositories the repository currently works with.

The three relevant repositories for this document are:

- Flutter. This is the top half of the Flutter project: flutter/flutter on GitHub
- Engine. This is the Flutter Engine, the lower half of the Flutter project: flutter/engine on GitHub
- Dart SDK. This is the language used for big parts of Flutter: dart-lang/sdk on GitHub

This document describes how to update the version of the Dart SDK that the Engine uses, and then update the version of the Engine that Flutter uses, so that Flutter gets the latest Dart.

The Flutter repo does not use a `DEPS` file and `gclient` for its dependencies. Instead, the Flutter Engine version is in `flutter/bin/internal/engine.version`.

Dart Autoroller

Dart is now automatically rolled into the Flutter engine repository on a regular basis. See Autorollers for more information.

Using dart_roll_helper.py to roll the version of Dart used by the engine

The `dart_roll_helper.py` script automates a large number of the Dart SDK roll steps, including:

- Updating the Dart revision in DEPS
- Updating the Dart dependencies in DEPS
- Syncing dependencies with `gclient sync`
- Generating GN files for relevant engine configurations
- Building relevant engine configurations
- Running tests in 'example/flutter_gallery' and 'packages/flutter'
- Launching flutter_gallery in release and debug mode
- Running license.sh and updating license files in 'flutter/ci/licenses_golden'
- Generating a commit with relevant Dart SDK commit logs (optional)

The following environment variables can be set instead of being passed as arguments:

- `$FLUTTER_HOME`: the absolute path to the 'flutter' directory
- `$ENGINE_HOME`: the absolute path to the 'engine/src' directory
- `$DART_SDK_HOME`: the absolute path to the root of a Dart SDK project

These paths are required for the tool to work.

Ensure your repository is in a clean state, as `gclient sync` may delete files.

To do a roll, run the following command (if the environment variables described above are not set, `--REPO-home` flags should be provided for each repository):

```
python $ENGINE_HOME/tools/dart/dart_roll_helper.py --create-commit $TARGET_SDK_HASH
```

If the script completes without errors, move on to step 10 in the next section to complete the roll.

Manually rolling the version of Dart used by the engine

1. Set up your Engine and Flutter environments by following the instructions described in the pages linked to from our contributing guide.
2. Build the engine according to the instructions on the [[Compiling the engine]] page.
3. Select a target Dart revision, typically use the latest revision. Check that the tests for that revision have all passed (all green) on the Dart buildbot and the Internal Dart Flutter buildbot.
4. Create a PR (see [[Tree hygiene]]) that updates `dart_revision` in DEPS to your selected revision. Invoke `gclient sync` in the src directory to ensure versions corresponding to the DEPS file are synched up.
5. Update all Dart-dependent DEPS entries using `engine/src/tools/dart/create_updated_flutter_deps.py` script. In case script complains that dart dependency was removed, remove entry from flutter DEPS file manually. If the list of library source files or patch files is modified, update the file libraries.yaml and regenerate the corresponding json file.
6. Invoke `gclient sync` in the src directory to ensure versions corresponding to the DEPS file are synched up
7. Build the debug, profile, and release versions of the engine, following the normal [[Compiling the engine]] instructions. You will need to build the host versions of the engine too. Here is a script with the build commands:

```
#!/bin/bash -e

set -ex

cd ~/engine/src
flutter/tools/gn --goma --runtime-mode=debug
flutter/tools/gn --goma --runtime-mode=profile
flutter/tools/gn --goma --runtime-mode=release
flutter/tools/gn --goma --android --runtime-mode=debug
flutter/tools/gn --goma --android --runtime-mode=profile
flutter/tools/gn --goma --android --runtime-mode=release
cd out
find . -mindepth 1 -maxdepth 1 -type d | xargs -n 1 sh -c 'ninja -C $0 -j1000 ||
exit 255'
```

7. Test the engine version you just built against the top of tree flutter/flutter version using the `--local-engine` option (see [[The flutter tool]]) against the Flutter Gallery app.

```
cd $FLUTTER_HOME/examples/flutter_gallery
flutter run --release --local-engine=android_release
flutter run --local-engine=android_debug_unopt
flutter test --local-engine=host_debug
```

In the debug version of flutter, run flutter_gallery and do a hot-reload and hot-restart of the app and ensure they both work.

```
cd $FLUTTER_HOME/packages/flutter
flutter test --local-engine=host_debug
```

8. Make sure your path contains `engine/src/third_party/dart/tools/sdks/dart-sdk/bin`, run the script `flutter/ci/licenses.sh` in the src directory, update

`flutter/ci/licenses_golden/licenses_third_party` by copying `out/license_script_output/licenses_third_party` into it. Include this change in your pull request. **If any licenses changed, make sure to also update the actual `LICENSE` file.**

9. It is useful to include the output of `git log --oneline <old revision>...<new revision>` (executed in the Dart SDK repository) in the commit message.
10. `git push origin <your_roll_branch_name>`
11. Get a code review of the pull request done, check to make sure the flutter engine build bot is green and merge the pull request.
12. Wait for the flutter engine build bot to build your change and go green.
13. Once the bot cycles green, the [[Autorollers]] will roll the engine into the flutter/flutter repo. When this happens, monitor the flutter build bots. If there is a failure, revert the Dart roll in flutter/engine, debug the problem and fix it or file a P0 issue against Dart. Please monitor the flutter benchmarks dashboard and if any regressions are noticed please file P0 issues for all regressions **and revert the Dart roll**. The next roll is blocked until these issues/regressions are fixed.
14. When you are done please update the spread sheet at http://go/dart-flutter-rolls with the git hash of the Dart revision that was rolled into the flutter engine. This hash will be used for rolling Dart into Google's internal code repository and would be picked as a potential candidate for a dev release. Also make sure you send an email to the next person on the list and make sure the person acknowledges picking up the roll baton.