Bundled Hermes

This page gives an overview of **how** Hermes and React Native **are built**.

If you're looking into instructions on how to use Hermes in your app, you can find instructions on this other page: using Hermes

> **CAUTION**
>
> Please note that this page serves as a technical deep dive and is targeted for users which are building extensions on top of Hermes or React Native. General users of React Native should not need to know in-depth information on how React Native and Hermes interact.

What is 'Bundled Hermes'

Starting with React Native 0.69.0, every version of React Native will be **built alongside** to a Hermes version. We call this distribution model **Bundled Hermes**.

From 0.69 on, you will always have a JS engine that has been built and tested alongside each React Native version that you can use.

Why we moved to 'Bundled Hermes'

Historically, React Native and Hermes followed two **distinct release processes** with distinct versioning. Having distinct releases with distinct numbers created confusion in the OSS ecosystem, where it was not clear if a specific version of Hermes was compatible with a specific version of React Native (i.e. you needed to know that Hermes 0.11.0 was compatible only with React Native 0.68.0, etc.)

Both Hermes and React Native, share the JSI code (Hermes here and React Native here). If the two JSI copies of JSI get out of sync, a build of Hermes won't be compatible with a build of React Native. You can read more about this ABI incompatibility problem here.

To overcome this problem, we've extended the React Native release process to download and build Hermes and made sure only one copy of JSI is used when building Hermes.

Thanks to this, we can release a version of Hermes whenever we release a version of React Native, and be sure that the Hermes engine we built is **fully compatible** with the React Native version we're releasing. We're shipping this version of Hermes alongside the React Native version we're doing, hence the name *Bundled Hermes*.

How this will impact app developers

As mentioned in the introduction, if you're an app developer, this change **should not affect** you directly.

The following paragraphs describe which changes we did under the hood and explains some of the rationales, for the sake of transparency.

iOS Users

On iOS, we've moved the `hermes-engine` you're using.

Prior to React Native 0.69, users would download a pod (here you can find the podspec).

On React Native 0.69, users would instead use a podspec that is defined inside the `sdks/hermes-engine/hermes-engine.podspec` file in the `react-native` NPM package. That podspec relies on a pre-built tarball of Hermes that we upload to Maven and to the React Native GitHub Release, as part of the React Native release process (i.e. see the assets of this release).

Android Users

On Android, we're going to update the `android/app/build.gradle` file in the default template the following way:

```
dependencies {
    // ...
```

```
    if (enableHermes) {
+       implementation("com.facebook.react:hermes-engine:+") {
+           exclude group:'com.facebook.fbjni'
+       }
-       def hermesPath = "../../node_modules/hermes-engine/android/";
-       debugImplementation files(hermesPath + "hermes-debug.aar")
-       releaseImplementation files(hermesPath + "hermes-release.aar")
    } else {
        implementation jscFlavor
    }
}
```

Prior to React Native 0.69, users will be consuming `hermes-debug.aar` and `hermes-release.aar` from the `hermes-engine` NPM package.

On React Native 0.69, users will be consuming the Android multi-variant artifacts available inside the `android/com/facebook/react/hermes-engine/` folder in the `react-native` NPM package. Please also note that we're going to remove the dependency on `hermes-engine` entirely in one of the future version of React Native.

Android Users on New Architecture

Due to the nature of our native code build setup (i.e. how we use the NDK), users on the New Architecture will be **building Hermes from source**.

This aligns the build mechanism of React Native and Hermes for users on the New Architecture (they will build both framework from source). This means that such Android users might experience a performance hit at build time on their first build.

You can find instructions to optimize your build time and reduce the impact on your build on this page: Speeding up your Build phase.

Android Users on New Architecture building on Windows

Users building React Native App, with the New Architecture, on Windows machines need to follow those extra steps to let the build work correctly:

- Make sure the environment is configured properly, with Android SDK & node.
- Install cmake with Chocolatey

- Install either:
  - Build Tools for Visual Studio 2022.
  - Visual Studio 22 Community Edition - Picking only the C++ desktop development is sufficient.
- Make sure the Visual Studio Command Prompt is configured correctly. This is required as the proper C++ compiler environment variable is configured in those command prompt.
- Run the app with `npx react-native run-android` inside a Visual Studio Command Prompt.

Can users still use another engine?

Yes, users are free to enable/disable Hermes (with the `enableHermes` variable on Android, `hermes_enabled` on iOS). The 'Bundled Hermes' change will impact only **how Hermes is built and bundled** for you.

Starting with React Native 0.70, the default for `enableHermes` / `hermes_enabled` is `true`.

How this will impact contributor and extension developers

If you're a React Native contributor or you're building an extension on top of React Native or Hermes, please read further as we explain how Bundled Hermes works.

How is Bundled Hermes working under the hood?

This mechanism relies on **downloading a tarball** with the Hermes source code from the `facebook/hermes` repository inside the `facebook/react-native` repository. We have a similar mechanism in place for other native dependencies (Folly, Glog, etc.) and we aligned Hermes to follow the same setup.

When building React Native from `main`, we will be fetching a tarball of `main` of facebook/hermes and building it as part of the build process of React Native.

When building React Native from a release branch (say `0.69-stable`), we will instead use a **tag** on the Hermes repo to **synchronize the code** between the two repos. The specific tag name used will then be stored inside the `sdks/.hermesversion` file inside React Native in the release branch (e.g. this is the file on the 0.69 release branch).

In a sense, you can think of this approach similarly to a **git submodule**.

If you're building on top of Hermes, you can rely on those tags to understand which version of Hermes was used when building React Native, as the version of React Native is specified in the tag name (e.g. `hermes-2022-05-20-RNv0.69.0-ee8941b8874132b8f83e4486b63ed5c19fc3f111`).

Android implementation details

To implement this on Android, we've added a new build inside the `/ReactAndroid/hermes-engine` of React Native that will take care of building Hermes and packaging for consumption (See here for more context).

You can now trigger a build of Hermes engine by invoking:

```
// Build a debug version of Hermes
./gradlew :ReactAndroid:hermes-engine:assembleDebug
// Build a release version of Hermes
./gradlew :ReactAndroid:hermes-engine:assembleRelease
```

from the React Native `main` branch.

You won't need to install extra tools (such as `cmake`, `ninja` or `python3`) in your machine as we configured the build to use the NDK versions of those tools.

On the Gradle consumer side, we also shipped a small improvement on the consumer side: we moved from `releaseImplementation` & `debugImplementation` to `implementation`. This is possible because the newer `hermes-engine` Android artifact is **variant aware** and will properly match a debug build of the engine with a debug build of your app. You don't need any custom configuration here (even if you use `staging` or other build types/flavors).

However, this made this line necessary in the template:

```
  exclude group:'com.facebook.fbjni'
```

This is needed as React Native is consuming `fbjni` using the non-prefab approach (i.e. unzipping the `.aar` and extracting `.so` files). Hermes-engine, and other libraries, are using prefab instead to consume fbjni. We're looking into addressing this issue in the future so the Hermes import will be a oneliner.

iOS implementation details

The iOS implementation relies on a series of scripts that lives in the following locations:

- `/scripts/hermes`. Those scripts contain logic to download the Hermes tarball, unzip it, and configure the iOS build. They're invoked at `pod install` time if you have the `hermes_enabled` field set to `true`.
- `/sdks/hermes-engine`. Those scripts contain the build logic that is effectively building Hermes. They were copied and adapted from the `facebook/hermes` repo to properly work within React Native. Specifically, the scripts inside the `utils` folder are responsible of building Hermes for all the Mac platforms.

Hermes is built as part of the `build_hermes_macos` Job on CircleCI. The job will produce as artifact a tarball which will be downloaded by the `hermes-engine` podspec when using a published React Native release (here is an example of the artifacts created for React Native 0.69 in `build_hermes_macos`).

Prebuilt Hermes

If there are no prebuilt artifacts for the React Native version that is being used (i.e. you may be working with React Native from the `main` branch), then Hermes will need to be built from source. First, the Hermes compiler, `hermesc`, will be built for macOS during `pod install`, then Hermes itself will be built as part of the Xcode build pipeline using the `build-hermes-xcode.sh` script.

Building Hermes from source

Hermes is always built from source when using React Native from the `main` branch. If you are using a stable React Native version, you can force Hermes to be built from source by

setting the `CI` envvar to `true` when using CocoaPods: `CI=true pod install`.

Debug symbols

The prebuilt artifacts for Hermes do not contain debug symbols (dSYMs) by default. We're planning on distributing these debug symbols for each release in the future. Until then, if you need the debug symbols for Hermes, you will need to build Hermes from source. A `hermes.framework.dSYM` will be created in the build directory alongside each of the Hermes frameworks.

I'm afraid this change is impacting me

We'd like to stress that this is essentially an organizational change on *where* Hermes is built and *how* the code is syncronized between the two repositories. The change should be fully transparent to our users.

Historically, we used to cut a release of Hermes for a specific version of React Native (e.g. `v0.11.0 for RN0.68.x`).

With 'Bundled Hermes', you can instead rely on a tag that will represent the version used when a specific version of React Native was cut.

*Last updated on **Aug 21, 2023***