React Native Gradle Plugin

This guide describes how to configure the **React Native Gradle Plugin** (often referred as RNGP), when building your React Native application for Android.

Using the plugin

The React Native Gradle Plugin is distributed as a separate NPM package which is installed automatically with `react-native`.

The plugin is **already configured** for new projects created using `npx react-native init`. You don't need to do any extra steps to install it if you created your app with this command.

If you're integrating React Native into an existing project, please refer to the corresponding page: it contains specific instructions on how to install the plugin.

Configuring the plugin

By default, the plugin will work **out of the box** with sensible defaults. You should refer to this guide and customize the behavior only if you need it.

To configure the plugin you can modify the `react` block, inside your `android/app/build.gradle`:

```
apply plugin: "com.facebook.react"

/**
 * This is the configuration block to customize your React Native Android app.
 * By default you don't need to apply any configuration, just uncomment the lines you
need.
 */
react {
    // Custom configuration goes here.
  }
```

Each configuration key is described below:

root

This is the root folder of your React Native project, i.e. where the `package.json` file lives. Default is `..`. You can customize it as follows:

```
root = file("../")
```

reactNativeDir

This is the folder where the `react-native` package lives. Default is `../node_modules/react-native`. If you're in a monorepo or using a different package manager, you can use adjust `reactNativeDir` to your setup.

You can customize it as follows:

```
reactNativeDir = file("../node_modules/react-native")
```

codegenDir

This is the folder where the `react-native-codegen` package lives. Default is `../node_modules/react-native-codegen`. If you're in a monorepo or using a different package manager, you can adjust `codegenDir` to your setup.

You can customize it as follows:

```
codegenDir = file("../node_modules/@react-native/codegen")
```

cliFile

This is the entrypoint file for the React Native CLI. Default is `../node_modules/react-native/cli.js`. The entrypoint file is needed as the plugin needs to invoke the CLI for bundling and creating your app.

If you're in a monorepo or using a different package manager, you can adjust `cliFile` to your setup. You can customize it as follows:

```
cliFile = file("../node_modules/react-native/cli.js")
```

`debuggableVariants`

This is the list of variants that are debuggable (see using variants for more context on variants).

By default the plugin is considering as `debuggableVariants` only `debug`, while `release` is not. If you have other variants (like `staging`, `lite`, etc.) you'll need to adjust this accordingly.

Variants that are listed as `debuggableVariants` will not come with a shipped bundle, so you'll need Metro to run them.

You can customize it as follows:

```
debuggableVariants = ["liteDebug", "prodDebug"]
```

`nodeExecutableAndArgs`

This is the list of node command and arguments that should be invoked for all the scripts. By default is `[node]` but can be customized to add extra flags as follows:

```
nodeExecutableAndArgs = ["node"]
```

`bundleCommand`

This is the name of the `bundle` command to be invoked when creating the bundle for your app. That's useful if you're using RAM Bundles. By default is `bundle` but can be customized to add extra flags as follows:

```
bundleCommand = "ram-bundle"
```

`bundleConfig`

This is the path to a configuration file that will be passed to `bundle --config <file>` if provided. Default is empty (no config file will be probided). More information on bundling config files can be found on the CLI documentation. Can be customized as follow:

```
bundleConfig = file(../rn-cli.config.js)
```

`bundleAssetName`

This is the name of the bundle file that should be generated. Default is `index.android.bundle`. Can be customized as follow:

```
bundleAssetName = "MyApplication.android.bundle"
```

`entryFile`

The entry file used for bundle generation. The default is to search for `index.android.js` or `index.js`. Can be customized as follow:

```
entryFile = file("../js/MyApplication.android.js")
```

`extraPackagerArgs`

A list of extra flags that will be passed to the `bundle` command. The list of available flags is in the CLI documentation. Default is empty. Can be customized as follows:

```
extraPackagerArgs = []
```

`hermesCommand`

The path to the `hermesc` command (the Hermes Compiler). React Native comes with a version of the Hermes compiler bundled with it, so you generally won't be needing to customize this. The plugin will use the correct compiler for your system by default.

`hermesFlags`

The list of flags to pass to `hermesc`. By default is `["-O", "-output-source-map"]`. You can customize it as follows

```
hermesFlags = ["-O", "-output-source-map"]
```

Using Flavors & Build Variants

When building Android apps, you might want to use custom flavors to have different versions of your app starting from the same project.

Please refer to the official Android guide to configure custom build types (like `staging`) or custom flavors (like `full`, `lite`, etc.). By default new apps are create with two build types (`debug` and `release`) and no custom flavors.

The combination of all the build types and all the flavors generates a set of **build variants**. For instance for `debug` / `staging` / `release` build types and `full` / `lite` you will have 6 build variants: `fullDebug`, `fullStaging`, `fullRelease` and so on.

If you're using custom variants beyond `debug` and `release`, you need to instruct the React Native Gradle Plugin specifying which of your variants are **debuggable** using the `debuggableVariants` configuration as follows:

```
apply plugin: "com.facebook.react"
```

```
react {
+ debuggableVariants = ["fullStaging", "fullDebug"]
}
```

This is necessary because the plugin will skip the JS bundling for all the `debuggableVariants`: you'll need Metro to run them. For example, if you list `fullStaging` in the `debuggableVariants`, you won't be able to publish it to a store as it will be missing the bundle.

What is the plugin doing under the hood?

The React Native Gradle Plugin is responsible for configuring your Application build to ship React Native applications to production. The plugin is also used inside 3rd party libraries, to run the Codegen used for the New Architecture.

Here is a summary of the plugin responsibilities:

- Add a `createBundle<Variant>JsAndAssets` task for every non debuggable variant, that is responsible of invoking the `bundle`, `hermesc` and `compose-source-map` commands.
- Setting up the proper version of the `com.facebook.react:react-android` and `com.facebook.react:hermes-android` dependency, reading the React Native version from the `package.json` of `react-native`.
- Setting up the proper Maven repositories (Maven Central, Google Maven Repo, JSC local Maven repo, etc.) needed to consume all the necessary Maven Dependencies.
- Setting up the NDK to let you build apps that are using the New Architecture.
- Setting up the `buildConfigFields` so that you can know at runtime if Hermes or the New Architecture are enabled.
- Setting up the Metro DevServer Port as an Android resource so the app knows on which port to connect.
- Invoking the React Native Codegen if a library or app is using the Codegen for the New Architecture.

Is this page useful?

Edit this page

Last updated on ***Jun 21, 2023***