Enabling in iOS Library

> **CAUTION**
>
> This documentation is still **experimental** and details are subject to changes as we iterate. Feel free to share your feedback on the discussion inside the working group for this page.
>
> Moreover, it contains several **manual steps**. Please note that this won't be representative of the final developer experience once the New Architecture is stable. We're working on tools, templates and libraries to help you get started fast on the New Architecture, without having to go through the whole setup.

You have defined the JavaScript specs for your native modules as part of the prerequisites, and you are now ready to migrate your library to the New Architecture. Here are the steps you can follow to accomplish this.

1. Updating your Podspec for the New Architecture

The New Architecture makes use of CocoaPods.

Add Folly and Other Dependencies

The New Architecture requires some specific dependencies to work properly. You can set up your podspec to automatically install the required dependencies by modifying the `.podspec` file. In your `Pod::Spec.new` block, add the following line:

```
Pod::Spec.new do |s|
  # ...
+  install_modules_dependencies(s)
  # ...
end
```

At this link, you can find the documentation of the `install_modules_dependencies` function.

If you need to explicitly know which `folly_flags` React Native is using, you can query them using the `folly_flag` function.

2. Extend or Implement the Code-generated Native Interfaces

The JavaScript spec for your native module or component will be used to generate native interface code for each supported platform (i.e., Android and iOS). These native interface files will be generated when a React Native application that depends on your library is built.

While this generated native interface code **will not ship as part of your library**, you do need to make sure your Objective-C or Java code conforms to the protocols provided by these native interface files. You can use the Codegen script to generate your library's native interface code in order to use it **as a reference**.

```
cd <path/to/your/app>
node node_modules/react-native/scripts/generate-codegen-artifacts.js \
    --path <your app>/ \
    --outputPath <an/output/path> \
```

This command will generate the boilerplate code required by iOS in the output path provided as a parameter.

The files that are output by the script **should not be committed**, but you'll need to refer to them to determine what changes you need to make to your native modules in order for them to provide an implementation for each generated `@protocol` / native interface.

Conform to the protocols provided by the native interface code

Update your native module or component to ensure it implements/extends the native interface that has been generated from your JavaScript specs.

Following the example set forth in the previous section, your library might import `MyAwesomeSpecs.h`, extend the relevant native interface, and implement the necessary methods for this interface:

```
#import <MyAwesomeSpecs/MyAwesomeSpecs.h>

@interface MyAwesomeModule () <StringGetterSpec>
@end

RCT_EXPORT_METHOD(getString:(NSString *)string
                  callback:(RCTResponseSenderBlock)callback)
{
  // Implement this method
}

- (std::shared_ptr<TurboModule>)getTurboModule:(const ObjCTurboModule::InitParams
&)params
{
  return std::make_shared<StringGetterSpecJSI>(params);
}
```

For an existing native module, you will likely already have one or more instances of `RCT_EXPORT_METHOD`. To migrate to the New Architecture, you'll need to make sure the method signature uses the structs provided by the Codegen output.

Is this page useful? 

Edit this page

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