Autolinking Native Modules

Edit

Autolinking is a mechanism that allows your React Native app project to discover and use native modules and view managers provided by React Native libraries.

This document covers autolinking for the Windows platform. It is an extension to the React Native CLI Autolinking doc.

Add a library using your favorite package manager and run the build:

```
yarn add react-native-webview
npx react-native run-windows
```

Copy

That's it. No more editing native files to use native code.

How does it work

From the React Native CLI Autolinking doc:

> Each platform defines its own `platforms` configuration. It instructs the CLI on how to find information about native dependencies. This information is exposed through the `config` command in a JSON format. It's then used by the scripts run by the platform's build tools. Each script applies the logic to link native dependencies specific to its platform.

Autolinking process

Autolinking is performed automatically as a part of the `run-windows` command:

1. At build time, autolinking is performed first, before `msbuild.exe` is invoked and the build actually started. It uses the information provided by `config` to both generate and modify certain native files consumed by your app project.

   i. The `AutolinkedNativeModules.g.targets` file contains the necessary references to the dependency projects that must be built.

   > Your app's solution file may also be modified to ensure the dependency projects will be built.

   ii. The `AutolinkedNativeModules.g.(cpp|cs)` files contain a `RegisterAutolinkedNativeModulePackages` method which registers all of the specified `IReactPackageProvider` s from the dependencies.

2. At build time, while `msbuild.exe` is running, but before compiling your app project, a check will verify that the autolinked files are up-to-date, and warn you if they aren't.

   > If you're using `run-windows` this check should always pass. However, if you've manually edited the generated files, or changed your npm dependencies and are building manually with Visual Studio, then the check might fail. See manually run autolinking.

3. At runtime, when your app is starting up it will call `RegisterAutolinkedNativeModulePackages` , registering the native dependencies with React Native, making them available to JS code.

Manually run autolinking

If you would like to run the autolinking process outside of the build, you can use the `autolink-windows` CLI command, i.e.:

```
npx react-native autolink-windows
```

Copy

OPTIONS	
`--logging`	Verbose output logging
`--check`	Only check whether any autolinked files need to change
`--sln` [string]	Override the app solution file determined by `react-native config`, e.g. *`windows\myApp.sln`*
`--proj` [string]	Override the app project file determined by `react-native config`, e.g. *`windows\myApp\myApp.vcxproj`*
`--no-telemetry` [boolean]	Disables sending telemetry that allows analysis of usage and failures of the react-native-windows CLI
`-h`, `--help`	output usage information

This sends telemetry to Microsoft by default. You can prevent the telemetry from being sent by using the `--no-telemetry` command line option. See the `@react-native-windows/cli` README for more details.

Skipping autolinking

If you would like to skip the autolinking process during `run-windows` you can pass `--no-autolink` option:

‹ Using Community Native Modules

Native Modules (Advanced) ›