React Native Config Schema Edit

The CLI command `npx react-native config` outputs project and dependencies configuration in JSON format to `stdout`.

The following describes the schema for projects and dependencies provided by React Native for Windows.

> See the React Native CLI Platforms doc for a description of the schemas for iOS and Android.

The schema fields are tagged with the following:

TAG	DESCRIPTION
auto	Item is always calculated by `config`. An override file should NEVER provide it.
req	Item is required. If an override file exists, it MUST provide it. If no override file exists, `config` will try to calculate it.
opt	Item is optional. If an override file exists, it MAY provide it. If no override file exists, `config` may try to calculate it.

`projectConfig`

Schema:

Copy

```
{
  folder: string,
  sourceDir: string,
  solutionFile: string,
  project: {
    projectFile: string,
    projectName: string,
    projectLang: string,
    projectGuid: string,
  },
}
```

Top-Level Fields:

The top-level object has the following fields:

FIELD	TYPE	TAG	DESCRIPTION
`folder`	string	auto	Absolute path to the app root folder, determined by `react-native config`, ex: *`c:\path\to\my-app`*
`sourceDir`	string	req	Relative path to the windows implementation under *folder*, ex: *`windows`*
`solutionFile`	string	req	Relative path to the app's VS solution file under *`sourceDir`*, ex: *`MyApp.sln`*
`project`	object	req	Object describing the app's VS project
`useWinUI3`	boolean	opt	If true, use WinUI 3. If false, use Windows XAML and WinUI 2. If missing, the value from `rnwRoot\PropertySheets\ExperimentalFeatures.props` will be used.

Project Object Fields:

The top-level `project` has the following fields:

FIELD	TYPE	TAG	DESCRIPTION
`projectFile`	string	req	Relative path to the VS project file under *`sourceDir`*, ex: *`MyApp\MyApp.vcxproj`* for *`c:\path\to\my-app\windows\MyApp\MyApp.vcxproj`*
`projectName`	string	auto	Name of the project, determined from *`projectFile`*, ex: *`MyApp`*
`projectLang`	string	auto	Language of the project, `cpp` (for C++) or `cs` (for C#), determined from *`projectFile`*
`projectGuid`	string	auto	Project identifier, determined from *`projectFile`*

Example `react-native.config.js` for a *`MyApp`*:

Copy

```
module.exports = {
  project: {
    windows: {
      sourceDir: 'windows',
      solutionFile: 'MyApp.sln',
      project: {
        projectFile: 'MyApp\\MyApp.vcxproj',
      },
    },
  },
};
```

dependencyConfig

provide a manual override file: `react-native.config.js` .

Schema:

Copy

```
{
  folder: string,
  sourceDir: string,
  solutionFile: string,
  projects: [
    {
      projectFile: string,
      directDependency: bool,
      projectName: string,
      projectLang: string,
      projectGuid: string,
      cppHeaders: [],
      cppPackageProviders: [],
      csNamespaces: [],
      csPackageProviders: []
    },
  ],
  nugetPackages: [
    {
      packageName: string,
      packageVersion: string,
      cppHeaders: [],
      cppPackageProviders: [],
      csNamespaces: [],
      csPackageProviders: [],
    },
  ],
}
```

Top-Level Fields:

The top-level object has the following fields:

			`c:\path\to\app-name\node_modules\my-module`
`sourceDir`	string	opt, req if projects defined	Relative path to the windows implementation under *folder*, ex: `windows`
`solutionFile`	string	opt	Relative path to the module's VS solution file under `sourceDir`, ex: `MyModule.sln`
`projects`	array	opt	Array of VS projects that must be added to the consuming app's solution file, so they are built
`nugetPackages`	array	opt	Array of NuGet packages including native modules that must be added as a dependency to the consuming app. It can be empty, but by its nature it can't be calculated

Project Object Fields:

Objects in the `projects` array have the following fields:

FIELD	TYPE	TAG	DESCRIPTION
`projectFile`	string	req	Relative path to the VS project file under `sourceDir`, ex: `MyModule\MyModule.vcxproj` for `c:\path\to\app-name\node_modules\my-module\windows\MyModule\MyModule.vcxproj`
`directDependency`	bool	req	Whether to add the project file as a dependency to the consuming app's project file. true for projects that provide native modules
`projectName`	string	auto	Name of the project, determined from `projectFile`, ex: `MyModule`
`projectLang`	string	auto	Language of the project, cpp or cs, determined from `projectFile`
`projectGuid`	string	auto	Project identifier, determined from `projectFile`

`cppPackageProviders`	array	opt	Array of fully qualified cpp `IReactPackageProviders`, i.e.: *`MyModule::ReactPackageProvider`*
`csNamespaces`	array	opt	Array of cs namespaces, i.e.: *`MyModule`*, to be transformed into `using MyModule;`
`csPackageProviders`	array	opt	Array of fully qualified cs `IReactPackageProviders`, i.e.: *`MyModule.ReactPackageProvider`*

NuGet Package Object Fields:

Objects in the `nugetPackages` array have the following fields:

FIELD	TYPE	TAG	DESCRIPTION
`packageName`	string	req	Name of the NuGet package to install
`packageVersion`	string	req	Version of the NuGet package to install
`cppHeaders`	array	req	Array of cpp header include lines, i.e.: *`winrt/NugetModule.h`*, to be transformed into `#include <winrt/NugetModule.h>`
`cppPackageProviders`	array	req	Array of fully qualified cpp `IReactPackageProviders`, i.e.: *`NugetModule::ReactPackageProvider`*
`csNamespaces`	array	req	Array of cs namespaces, i.e.: *`NugetModule`*, to be transformed into `using NugetModule;`
`csPackageProviders`	array	req	Array of fully qualified cs `IReactPackageProviders`, i.e.: *`NugetModule.ReactPackageProvider`*

Example `react-native.config.js` for a *`MyModule`*:

```
        sourceDir: 'windows',
        solutionFile: 'MyModule.sln',
        projects: [
          {
            projectFile: 'MyModule\\MyModule.vcxproj',
            directDependency: true,
          }
        ],
      },
    },
  },
};
```

‹ React Native Windows Components and APIs

Using PlatformColor and Responding to Themes ›