youi-tv build and generate

You.i TV uses **CMake** as its build system. CMake is cross-platform, meaning that developers can select the build environment of their choice. Depending on your target platform(s), you may be required to do development from more than one platform. The table below helps identify the dependencies.

Development Platform	IDE	Target Platforms
macOS	• Xcode (iOS, tvOS, macOS) • Android Studio (Android) • command line + editor of your choice	`android`, `ios`, `osx`[1], `tizen-nacl`[2], `tvos`, `osx` for Roku[3,4]
Linux	• Android Studio (Android) • command line + editor of your choice	`android`, `linux`[1], `linux`[4] for Roku, `tizen-nacl`[2], `webos3`, `webos4`
Windows	• Visual Studio (PS4, UWP) • Android Studio (Android) • command line + editor of your choice	`android`, `ps4`, `tizen-nacl`[2], `uwp`[1], `win64`

[1] Supported as a development platform only.

[2] Tizen (`tizen-nacl` target) must always be built from the command line, there is no IDE support.

[3] macOS is supported for initial development with Roku Cloud Solution.

[4] Specify target `osx` or `linux` and additional compiler flag `-d YI_BUILD_CLOUD_SERVER=ON`

Command Line Build Tools

The `youi-tv` CLI includes commands for generating project files and building your application from the command line. Even if you plan to do most of your development through an IDE, it's useful to know about these commands.

Commands:

- `youi-tv generate`
- `youi-tv build`

The `generate` command builds a framework of files for your project, and depending on the type of target platform, also builds the related IDE project file. For example, `youi-tv generate -p osx` builds a project you can open in Xcode. And

`youi-tv generate -p uwp` builds a project for Visual Studio.

The `build` command builds a debug or release binary executable for your application. It's intelligently engineered to know whether or not you've already generated the required project files for the requested target platform, and it runs `generate` for you, if needed.

> PRO TIP
>
> Also try our handy shortcut command, `youi-tv run`, which builds, deploys, and launches an app, and generates the project if necessary.

Basic Build

The easiest way to build your application is to execute the following from a shell in your project folder:

```
youi-tv build -p <platform>
```

where platform is the target platform you want to build for. Use the table above or `youi-tv build --help` to know which target platforms are available from your current development platform.

Generate Options

The `youi-tv generate` command has the following options:

Argument	Description
`-b, --build_directory DIRECTORY`	Optional. Where to place generated project files. Default: `<app>/build/<platform>`
`-c, --config CONFIGURATION`	Optional. The configuration type [`Debug`, `Release`] to send to generator. It is only required for generators that do not support multiple configurations.
`--clean`	Optional. Removes old project files before generating.
`-d, --define NAME=VALUE`	While debugging your application with the Metro bundler (yarn server), enable Hermes with `-d YI_JAVASCRIPT_EXECUTOR=hermes`. For production builds and store submissions, in addition to the define above, use `--local` and `YI_LOCAL_BYTECODE` to enable Hermes bytecode in your app. The full string is `--local -d YI_JAVASCRIPT_EXECUTOR=hermes -d YI_LOCAL_BYTECODE=ON` . Hermes builds for Tizen can only be done on a Linux development platform. To enable Hermes on Tizen, add `-d YI_ARCH=armv7-gcc` to the generate command.
`-g, --generator GENERATOR`	Optional. The name of the generator to use. If not mentioned, the platform's default generator is used.
`-p, --platform PLATFORM`	**Required**. The name of the target platform. Use `youi-tv generate --help` to see valid choices for your current development platform.

`--youi_version ENGINE_HINT`	Optional. Can be a path, such as `/path/to/5.0`, or semantic version, such as `5.0.1`, and the project gets generated against the mentioned You.i Platform version.
Plus the following options, available for React Native builds only	
`--bundler_configuration [file]`	Optional. Pass a config file to Metro, for example, `path/to/config.js`
`--dev`	Optional. If included, the JS file is bundled with the dev flag enabled.
`--directory`	Optional. If included, adds the JS files of the listed directory, and create a bundle for each one.
`--file [file]`	Optional. The entry point for the application, for JS bundling.
`--inline`	Optional. If included, JS bundles are compiled directly into the source code instead of fetched from a yarn server.
`--iterate`	Optional. If included, Only JS bundles that need to be updated by changes to their source JS file will be re-bundled. If omitted, all JS bundles will be deleted before creating all required bundles. Note that changes to a JS file that is a dependency of the source JS file will not cause the file to be automatically re-bundled. Use without `--iterate` to capture all changes.
`--local`	Optional. If included, packages JS bundles with the app instead of fetching from a yarn server. If you want to deploy the production version of your Cloud Solution app on AWS, you **must** use the `--local` build argument.
`--minify`	Optional. If included, the JS file is bundled with the minify flag enabled.
`--ram_bundle`	Optional. To learn more about using RAM bundling with You.i Platform, see Optimizing Start-up Performance Using RAM Bundling.
`--remote [ip address]`	Optional, default if `--local` is not used. Specify the address where the Metro Bundler can be found. JS bundles are fetched from that bundler. When building for your local development platform, `localhost` is used. Otherwise, your development platform's IP address is used. Override by providing an IP address.
`--sourcemap_output [file]`	Optional. File name to store the sourcemap for resulting bundle in, ex. /tmp/index.map
`--generated_bundle_file [file]`	Optional. Use this option to skip the default bundle generation and specify a custom bundle file instead. You must use this option with any of the following options: `--local`, `--inline`, or `--ram_bundle`.

Use `youi-tv generate --help` for a complete list of options and their usage.

Build Options

The `youi-tv build` command has the same options as `youi-tv generate`, plus the following additional options:

Argument	Description
`-a, --arg BUILD_ARGUMENT`	Custom argument that can be passed to CMake when building. These are passed down to the native build system. For example when `-j8` is passed to Makefiles, it builds with 8 threads, etc. Multiple arguments on the command line are supported. (default: [ ])
`-t, --target TARGET`	• CopyAssets: Copies the assets from the project's AE assets directory to the location required by the platform for execution. • CleanAssets: Cleans up the assets which were copied by the CopyAssets target. • Package: Packages the application for the specific platform. This target is only available for platforms which require a packaged application.

Use `youi-tv build --help` for a complete list of options and their usage.

CMake Configuration

Compiler Warnings

By default, You.i Platform treats all compiler warnings as errors. This helps you find potential bugs that might be masked as a code compilation warning. Although we don't recommend changing this functionality, your project may have expected warnings. To change the default behavior, edit (or add) the following option in your project's `youi/CMakeLists.txt` file. To treat warnings as errors, set to `ON`. To ignore compiler warnings, set to `OFF`.

```
option(YI_TREAT_WARNINGS_AS_ERRORS "Warnings will result in fa
```

Hermes JavaScript Engine

Hermes JavaScript Engine optimizes the start-up performance of your You.i React Native app. See Target Platforms to know more about which target platforms support Hermes. To learn more about Hermes in general, see Facebook's topic on the Hermes JavaScript Engine ↗.

Hermes builds for Tizen can only be done on a Linux development platform. To enable Hermes on Tizen, add `-d YI_ARCH=armv7-gcc` to the generate command. Note that Hermes requires Tizen 2017 TVs or higher, so enabling Hermes may require you to target your Tizen app to specific models ↗.

As described above, you can use the command line parameter `-d` to define your Hermes preferences at build time. You can also configure these parameters permanently for your projects by adding cache variables at the top of your project's `CMakeLists.txt` file. Note that when testing with the Metro bundler (yarn server), you only need the first of these two settings.

```
set(YI_JAVASCRIPT_EXECUTOR "hermes" CACHE STRING "JS executor:
set(YI_LOCAL_BYTECODE "ON" CACHE STRING "Hermes ByteCode: ON")
```

The `YI_LOCAL_BYTECODE` option only works in conjunction with `--local` (which bundles assets with the app for production builds and store submissions). If desired, edit your `package.json` file to configure use of `--local` more permanently for your project.

```
"youi" : {
  "defaultBundleMode": "local"
}
```

The opposite of `--local` is `--remote` (meaning assets are remotely loaded from the Metro bundler). Our build system uses `--remote` by the default if `--local` isn't specified.

> IMPORTANT
>
> Hermes doesn't support the `Intl` library. Using `Intl` in a Hermes build may cause app runtime issues.

Default Location for Generated Files

If you use the default build directory (that is, you didn't specify the `-b` option), the built files are placed in the following locations:

Note For C++ projects, omit the `youi` directory in the paths below.

macOS

- Generated Project Files Path: `MyApp/youi/build/osx/MyApp.xcodeproj`
- Generated Files for a Debug Version App: `MyApp/youi/build/osx/Debug/MyApp`
- Generated files for a Release Version App: `MyApp/youi/build/osx/Release/MyApp`

iOS

- Generated Project Files Path: `MyApp/youi/build/ios/MyApp.xcodeproj`
- Generated Files for a Debug Version App: `MyApp/youi/build/ios/Debug-iphoneos/MyApp`
- Generated Files for a Release Version App: `MyApp/youi/build/ios/Release-iphoneos/MyApp`

tvOS

- Generated Project Files Path: `MyApp/youi/build/tvos/MyApp.xcodeproj`
- Generated Files for a Debug Version App: `MyApp/youi/build/tvos/Debug-appletvos/MyApp`
- Generated Files for a Release Version App: `MyApp/youi/build/tvos/Release-appletvos/MyApp`

Android

- Generated Project Files Path: `MyApp/youi/build/android/project/build.gradle`
- Generated Files for a Debug Version App: `MyApp/youi/build/android/project/MyApp/build/outputs/apk/MyApp-debug.apk`
- Generated Files for a Release Version App: `MyApp/youi/build/android/project/MyApp/build/outputs/apk/MyApp-release.apk`

UWP

- Generated Project Files Path: `MyApp\youi\build\uwp\MyApp.sln`
- Generated Files for a Debug Version App: `MyApp\youi\build\uwp\AppPackages\MyApp\MyApp_1.0.0.0_x64_Debug_Test`
- Generated Files for a Release Version App: `MyApp\youi\build\uwp\AppPackages\MyApp\MyApp_1.0.0.0_x64_Test`

Roku

- Generated Project Files Path: `MyApp/youi/build/osx/MyApp.xcodeproj` (Mac Only)
- Generated Files for a Debug Version App:
  - Mac: `MyApp/youi/build/osx/Debug/MyApp`
  - Linux: `MyApp/youi/build/linux/Debug/bin/MyApp`
- Generated Files for a Release Version App:
  - For Mac: `MyApp/youi/build/osx/Release/MyApp`
  - For Linux: `MyApp/youi/build/linux/Release/bin/MyApp`

Tizen

- Generated Project Files Path: `MyApp/youi/build/tizen-nacl/<ConfigurationType>/MyApp-<ConfigurationType>.wgt`
- Generated Files for a Debug Version App: `MyApp/youi/build/tizen-nacl/Debug/MyApp-Debug.wgt`
- Generated Files for a Release Version App: `MyApp/youi/build/tizen-nacl/Release/MyApp-Release.wgt`

PS4

- Generated Project Files Path: `MyApp\youi\build\ps4\MyApp.sln`

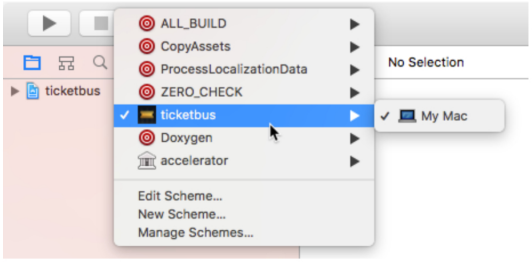
webOS

- Generated Files for a Debug Version App: `MyApp/youi/build/webos<3 or 4>/Debug/tv.youi.myapp_5.x.x_arm.ipk`
- Generated Files for a Release Version App: `MyApp/youi/build/webos<3 or 4>/Release/tv.youi.myapp_5.x.x_arm.ipk`

Build and Run from Xcode

Once your project file is generated with `youi-tv generate`, you can build and run the app from Xcode:

1. Start Xcode. Open the generated Xcode project file from your project's build folder. (See above for output file locations.) For example, if your project is called HelloWorld and you're building for the macOS platform, open:
2. `build/osx/HelloWorld.xcodeproj`
   a. Xcode opens the project and indexes the project files.
3. Select your application target from the dropdown menu to the right of the **Play** button.



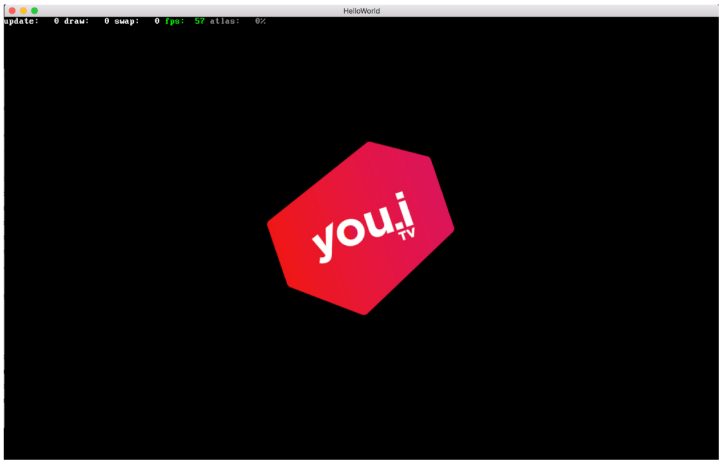
4. Press the **Play** button in the top-left corner, or press **Cmd-R** to run the app. You

may be prompted to enable developer mode.

> IMPORTANT
>
> With Xcode 13, you may see an error about a legacy build system. You can either:
>
> - Switch to the new build system in Xcode, by choosing **File** > **Project Settings** > **Build System** > **New Build System (Default)**, or
> - Remain on the "Legacy Build System (Deprecated)" and ignore further messages by enabling **Do not show a diagnostic issue about build system deprecation.**
>
> If you are building with You.i CLI, you can also pass the option `-UseModernBuildSystem=1` to use the new build system and disable the error.

For iOS and tvOS builds, you can switch between any real device plugged into your Mac and between the simulators for various types of devices. Click the dropdown menu to the right of the **Play** button and select the device or simulated device of your choice.

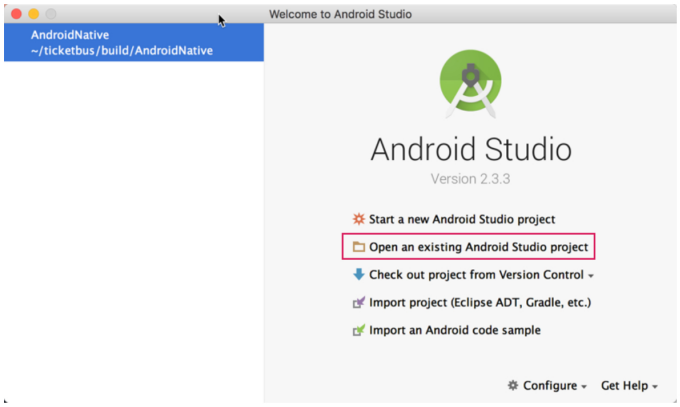
Your app launches in a new window. For example, if trying the **HelloWorld** C++ sample app, you'll see a spinning You.i TV logo, and know that your configuration is successful.



Build and Run from Android Studio

Once your project file is generated with `youi-tv generate`, you can build and run the app from Android Studio:

1. Start **Android Studio.**
2. Click **Open an existing Android Studio project** in the **Welcome** window.



3. In the **Open File** or **Project** window, open the generated Android Studio project file from your project's build folder. (See above for output file locations.) For example, if your project is called HelloWorld and you're building for the android

platform, open `build/android/project` and click **OK.**

4. Select the project folder from the IDE.
5. Ensure that the **build variant** is **debug**.

   Ensure that an appropriate physical device is connected, or that an appropriate Android Virtual Device (AVD) is configured and running. You can switch between any real device plugged into your computer and between the simulators for various types of devices by selecting a different deployment target.

Build and Run from Visual Studio

Once your project file is generated with `youi-tv generate`, you can build and run the app from Visual Studio:

1. Start **Visual Studio**.
2. Select **Open Project...** from the **Visual Studio Start** page.
3. Open the generated Visual Studio project file from your project's build folder. (See above for output file locations.) For example, if your project is called HelloWorld and you're building for the ps4 platform, open `build\ps4\HelloWorld.sln`.
4. Click **Open** in the **Open Project** window.

   The Solution Explorer displays a list of all projects.
5. Select your project and click **Play**.
6. Use the **Play dropdown list** to select a **target** machine.