Appcelerator CLI Tasks

This document provides information on how to build cloud and mobile applications with Appcelerator CLI.

- Build cloud applications
    - Create an Arrow project
    - Run an Arrow project
    - Publish an Arrow project to Arrow Cloud
    - Generate Cloud components
- Build mobile applications
    - Create a Alloy project
    - Create a Titanium Classic project
    - Build an Alloy or Titanium Classic project
        - Android emulator
        - Android device
        - iOS simulator
        - iOS device
        - Windows Phone emulator
        - Windows Phone device
        - Windows computer
    - Package an application
        - Google Play APK
        - iOS ad hoc distribution
        - iTunes store
        - Windows Phone Store
        - Windows Store
    - Clean your build folder

Build cloud applications

For more tasks, see API Builder Tools Project.

Create an Arrow project

To create a new Arrow project, run the `appc new` command. The CLI will prompt you to fill in the necessary information to create a project, such as the name of the project. To create a new project with all the information specified in the command, run:

```
appc new -t arrow -n <PROJECT_NAME>
```

Run an Arrow project

To run an Arrow project locally, run the `appc run` command from the project directory.

Publish an Arrow project to Arrow Cloud

To deploy your Arrow application to the Arrow Cloud, run the `appc publish` command from the project directory.

Generate Cloud components

To generate components, such as Models, Connectors, or Block, for your Arrow projects, run the `appc generate` command. When prompted for the type of component, select **Arrow Component**, then follow the subsequent directions.

The components generated by the command are modular components that need to be published using the `appc publish` command, then installed to your Arrow projects using the `appc install` command.

Build mobile applications

Once you have your development environment correctly configured, you can create and build mobile applications.

Create a Alloy project

An alloy project will be created by default when the `appc new` command is run. The CLI will prompt you to fill in the necessary information to create a project, such as which platforms the application runs on and the name of the project. To create a new project with all the information specified in the command, run:

```
appc new -t app --id <APP_ID> -n <APP_NAME>
## Example
appc new -t app --id com.appcelerator.sample -n SampleProject
```

A new app will be created for all supported platforms by default, depending on the operating system.

Create a Titanium Classic project

To create a classic Titanium project, follow the same directions when creating an Alloy project, except add the `--classic` flag to generate a Titanium classic project.

```
appc new -t app --id <APP_ID> -n <APP_NAME> --classic
```

Build an Alloy or Titanium Classic project

To build an Alloy or Titanium project to test on a device, simulator or emulator, run the `appc run` command from the project directory. The CLI will prompt you to fill in the necessary information to build the project, such as which platform you want build the project for. More detailed examples are listed below.

Once the application is installed and launched, use native tools to test, debug and profile your application. See Debugging and Profiling.

SDK version setting precedence

The CLI checks several settings to see which SDK version to use to build your application. The following is a list of locations in the order of precedence. If an SDK version is not defined in that location, the next location is checked.

1. `tiapp.xml` file version specified with the `sdk-version` tag.
   To change this version, manually edit the `tiapp.xml` file with a text editor or use Studio.
2. `--sdk` command-line option with the `appc run` command.
3. `app.sdk` setting specified with the `appc ti config` command.
   To check the version, run `appc ti config` and to change the version, run `appc ti config app.sdk <sdk_version>`.
4. SDK select version.
   To check or change this version, run `appc ti sdk select`.

Android emulator

You need to create an Android emulator or setup Genymotion before running these commands.

- To create an Android emulator, see Native Android Debugging and Testing Tools: Creating an emulator.
- To setup Genymotion, see Installing Genymotion.

To build for an emulator, run `appc run -p android`. Because no other options were specified, the CLI launches your default Android emulator and installs the application on it.

To launch a specific Android or Genymotion emulator, add the `-C <EMULATOR_NAME>` option.

```
appc run -p android -C "Google Nexus 7 - 4.4.2 - API 19 - 800x1280"
```

To retrieve a list of Android or Genymotion emulators, run the `ti info -p android` command.

Android device

To build for an Android device, connect your device to the computer with a USB cable, then run:

```
appc run -p android -T device -C <DEVICE_ID>
## Example
appc run -p android -T device -C deadbeef
```

iOS simulator

To build for an iOS simulator, run `appc run -p ios`. Because no other options were specified, the CLI launches the application on the default iOS simulator.

To use a specific simulator, add the `-C <DEVICE_UDID>` option.

```
appc run -p ios -C "D27F9E87-7E09-48D8-9DD1-10277A0B51A"
```

To retrieve the simulator's UDID, run the `appc ti info -p ios` command.

iOS device

Before deploying to an iOS device for testing, you need to generate a development certificate and development provisioning profile. See Deploying to iOS devices.

The CLI allows you to install your application either to iTunes or directly to the device connected to your computer with a USB cable. If you select to install to iTunes, you need to sync your device to iTunes in order to install the application.

To build for an iOS device, run:

```
appc run -p ios -T device  -C <DEVICE_UDID> [-V "<DEVELOPER_CERTIFICATE_NAME>"  -P
<PROVISIONING_PROFILE_UUID>]
## Example
appc run -p ios -T device -C itunes -V "Loretta Martin (GE7BAC5)" -P
"11111111-2222-3333-4444-555555555555"
appc run -p ios -T device -C "iOS Device" -V "Loretta Martin (GE7BAC5)" -P
"11111111-2222-3333-4444-555555555555"
```

If you omit the `-V` and `-P` options, the CLI will prompt you with options. You can also retrieve the information from Xcode's Devices window or with the `appc ti info -p ios` command.

Windows Phone emulator

Support for Windows 8.1 and Windows Phone SDKs has been deprecated as of SDK 6.3.0.GA and has be removed in SDK 7.0.0.GA.

Windows Phone emulator requires CLI and SDK 4.1.0 or later.

To build for a Windows Phone emulator, you need to obtain your Windows publisher ID. After you have created your Windows Dev Center account, log in to https://dev.windowsphone.com/en-us/Account/Summary to get your publish GUID.

To build for a Windows Phone emulator, run:

```
appc run -p windows [-C <DEVICE_UDID> -I <WINDOWS_PUBLISHER_ID>]
## Example
appc run -p windows -C 8-1-1 -I "CN=00000000-0000-1000-8000-000000000000"
```

If you omit any of the optional parameters, the CLI will prompt you with options.

Windows Phone Publisher GUID

Prior to Release 5.0.0, you need to pass the `-G` option with your Windows Phone Publisher GUID rather than using the `-I` option with your Windows Publisher ID.

Windows Phone device

Support for Windows 8.1 and Windows Phone SDKs has been deprecated as of SDK 6.3.0.GA and has be removed in SDK 7.0.0.GA.

Windows Phone emulator requires CLI and SDK 4.1.0 or later.

To build for a Windows Phone emulator, you need to obtain your Windows publisher ID. After you have created your Windows Dev Center account, log in to https://dev.windowsphone.com/en-us/Account/Summary to get your publish GUID.

To deploy to a Windows Phone device, connect the device to your computer with a USB cable, then run:

```
appc run -p windows -T wp-device [-C <DEVICE_UDID> -I <WINDOWS_PUBLISHER_ID>]
## Example
appc run -p windows -T wp-device -C 0 -I "CN=00000000-0000-1000-8000-000000000000"
```

If you omit any of the optional parameters, the CLI will prompt you with options.

Windows Phone Publisher GUID
Prior to Release 5.0.0, you need to pass the `-G` option with your Windows Phone Publisher GUID rather than using the `-I` option with your Windows Publisher ID.

Windows computer

Support for Windows 8.1 and Windows Phone SDKs has been deprecated as of SDK 6.3.0.GA and has be removed in SDK 7.0.0.GA.

Windows Phone emulator requires CLI and SDK 4.1.0 or later.

To deploy the application to your local Windows machine, you will need a certificate. If you do not have a certificate, the CLI will launch the certificate maker for your to create one.

To deploy to your local machine, run:

```
appc run -p windows -T ws-local [-R <PFX_CERTIFICATE_FILE> -P <PFX_PASSWORD> -I
<WINDOWS_PUBLISHER_ID>]
```

If you omit any of the optional parameters, the CLI will prompt you with options. Prior to Release 5.0.0, the `-I` option was not required.

Package an application

To package a mobile application, run the `appc run` command from the project directory.

Google Play APK

Before packaging an APK file for distribution, you need to generate a keypair and certificate for your application. See Distributing Android apps.

To package an APK for Google Play, run:

```
appc run -p android -T dist-playstore [-K <KEYSTORE_FILE> -P <KEYSTORE_PASSWORD> -L
<KEYSTORE_ALIAS> -O <OUTPUT_DIRECTORY>]
## Example
appc run -p android -T dist-playstore -K ~/android.keystore -P secret -L foo -O
./dist/
```

If you omit any of the optional parameters, you will be prompted by the CLI to enter these values. If the password for the private key of the keystore is different from the password for the keystore, add the `--key-password` <KEYPASS> option with the password of your private key.

iOS ad hoc distribution

Before packaging for the iOS ad hoc distribution, you need to generate a distribution certificate and distribution provisioning profile. See Distributing iOS apps.

To package an IPA file for Ad Hoc distribution, run:

```
appc run -p ios -T dist-adhoc [-R <DISTRIBUTION_CERTIFICATE_NAME> -P
<PROVISIONING_PROFILE_UUID> -O <OUTPUT_DIRECTORY>]
## Example
appc run -p ios -T dist-adhoc -R "Pseudo, Inc." -P
"FFFFFFFF-EEEE-DDDD-CCCC-BBBBBBBBBBBB" -O ./dist/
```

If you omit any of the optional parameters, the CLI will prompt you with options.

iTunes store

Before packaging for the iTunes Store, you need to generate a distribution certificate and distribution provisioning profile, and have an iTunes Connect account. See Distributing iOS apps.

> ⚠ If you are using Mac OS X 10.9 (Mavericks), make sure you grant CLI access to the computer as described in Installing the iOS SDK: Note for Mavericks.

To package an APP bundle for the iTunes store, run:

```
appc run -p ios -T dist-appstore [-R <DISTRIBUTION_CERTIFICATE_NAME> -P
<PROVISIONING_PROFILE_UUID>]
## Example
appc run -p ios -T dist-appstore -R "Pseudo, Inc." -P
"AAAAAAAA-0000-9999-8888-777777777777"
```

The CLI installs the package to Xcode's Organizer and launches Organizer for you to start the submission process.

If you omit any of the optional parameters, the CLI will prompt you with options.

Windows Phone Store

> ⛔ Support for Windows 8.1 and Windows Phone SDKs has been deprecated as of SDK 6.3.0.GA and has be removed in SDK 7.0.0.GA.

> ⚠ Windows Phone emulator requires CLI and SDK 4.1.0 or later.

To package for the Windows Phone Store (ARM architecture), you need to obtain your Windows publisher ID. After you have created your Windows Dev Center account, log in to https://dev.windowsphone.com/en-us/Account/Summary to get your publish GUID.

To package an APPX file for the Windows Phone Store, run:

```
appc run -p windows -T dist-phonestore [-I <WINDOWS_PUBLISHER_ID> -O
<OUTPUT_DIRECTORY>]
```

If you omit any of the optionals parameters, the CLI will prompt you with options.

> **Windows Phone Publisher GUID**
> Prior to Release 5.0.0, you need to pass the `-G` option with your Windows Phone Publisher GUID rather than using the `-I` option with your Windows Publisher ID.

Windows Store

> Support for Windows 8.1 and Windows Phone SDKs has been deprecated as of SDK 6.3.0.GA and has be removed in SDK 7.0.0.GA.

> Windows Phone emulator requires CLI and SDK 4.1.0 or later.

To package for the Windows Store (x86 architecture), you will need a certificate. If you do not have a certificate, the CLI will launch the certificate maker for your to create one.

To package an APPX file for the Windows Store, run:

```
appc run -p windows -T dist-winstore [-I <WINDOWS_PUBLISHER_ID> -R
<PFX_CERTIFICATE_FILE> -P <PFX_PASSWORD> -O <OUTPUT_DIRECTORY>]
```

If you omit any of the optional parameters, the CLI will prompt you with options. Prior to Release 5.0.0, the `-I` was not required.

Clean your build folder

If you run into issues building your application, you may need to clean your build folder. Run the `appc ti clean` command or to clean for a specific platform, add the `-p <PLATFORM>` option.

```
appc ti clean [-p <PLATFORM>]
## Examples
appc ti clean
appc ti clean -p ios
```