Android Publishing Guide

Create App's .apk file

To generate a release build for Android, we first need to make a small change to the AndroidManifest.xml file found in platforms/android. Edit the file and change the line:

```
<application android:debuggable="true" android:hardwareAccelerated="true"
android:icon="@drawable/icon" android:label="@string/app_name">
```

and change android:debuggable to "false":

```
<application android:debuggable="false" android:hardwareAccelerated="true"
android:icon="@drawable/icon" android:label="@string/app_name">
```

Now we can tell cordova to generate our release build:

```
$ cordova build --release android
```

Then, we can find our unsigned APK file in platforms/android/ant-build. In our example, the file was platforms/android/ant-build/HelloWorld-release-unsigned.apk. Now, we need to sign the unsigned APK and run an alignment utility on it to optimize it and prepare it for the app store. If you already have a signing key, skip these steps and use that one instead.

Note: The jarsigner and zipalign commands have to be executed from the platforms/android/ant-build directory

Let's generate our private key using the keytool command that comes with the JDK. If this tool isn't found, refer to the installation guide:

```
$ keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -
validity 10000
```

You'll first be prompted to create a password for the keystore. Then, answer the rest of the nice tools's questions and when it's all done, you should have a file called my-release-key.keystore created in the current directory.

Note: Make sure to save this file somewhere safe, you'll need it to submit updates to your app!

To sign the unsigned APK, run the jarsigner tool which is also included in the JDK:

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore
HelloWorld-release-unsigned.apk alias_name
```

This signs the apk in place. Finally, we need to run the zip align tool to optimize the APK:

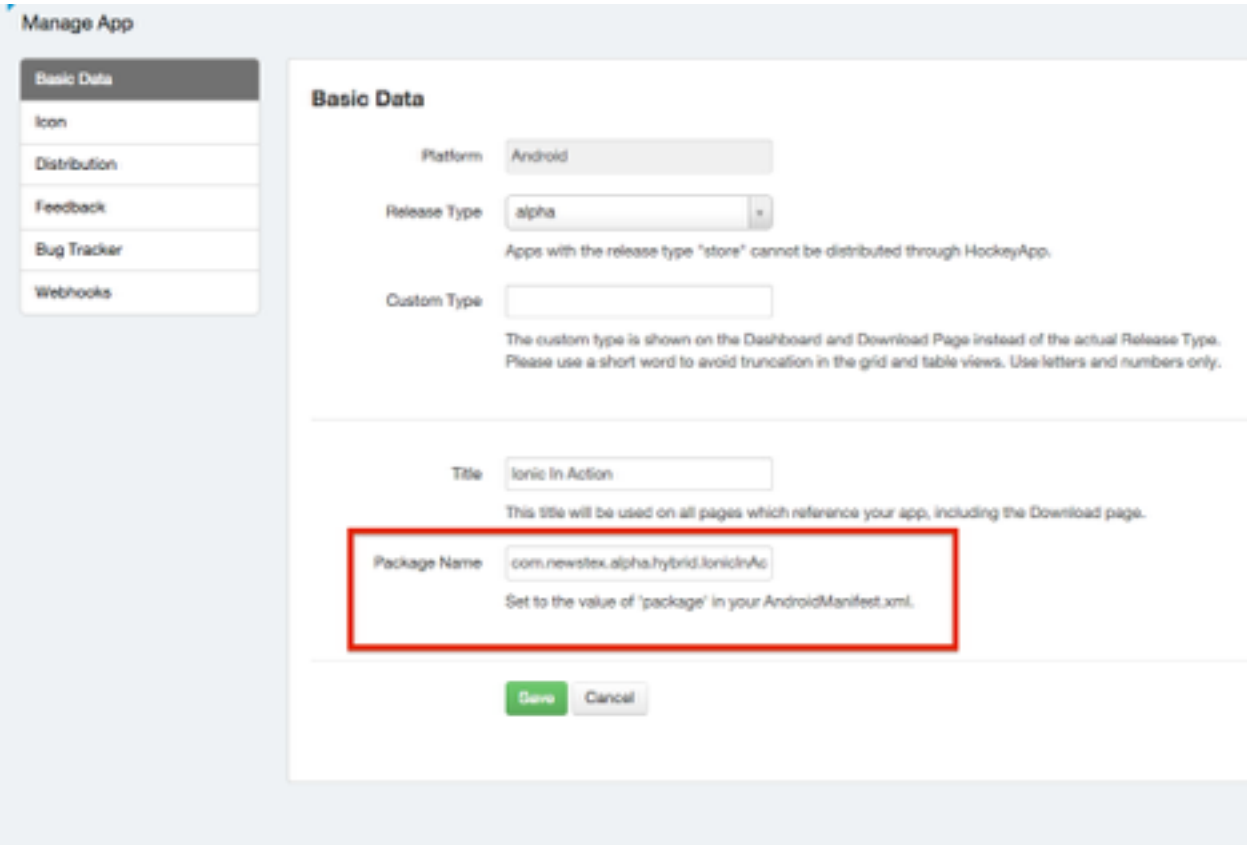
```
$ zipalign -v 4 HelloWorld-release-unsigned.apk HelloWorld.apk
```

Note: if the .apk file already exists, delete it and generate a new one with the line above.

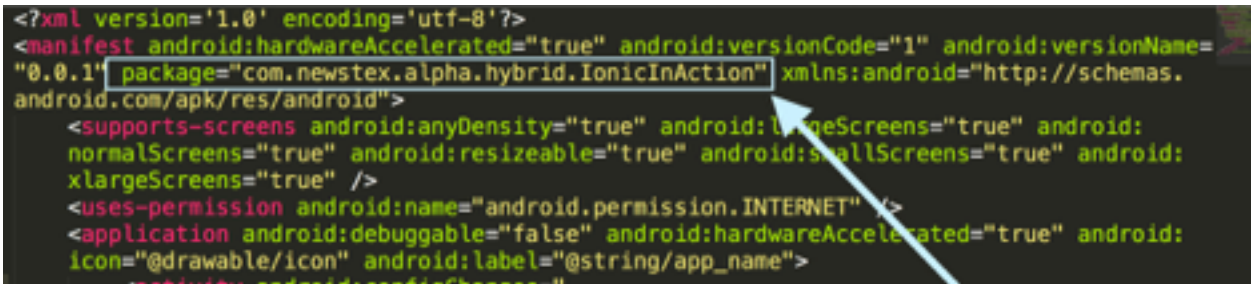
Now we have our final release binary called HelloWorld.apk to release.

Distribute to Hockey App

IMPORTANT: An app with a Package Name on Hockey must match the package value in the AndroidManifest.xml. before we can upload the .apk file

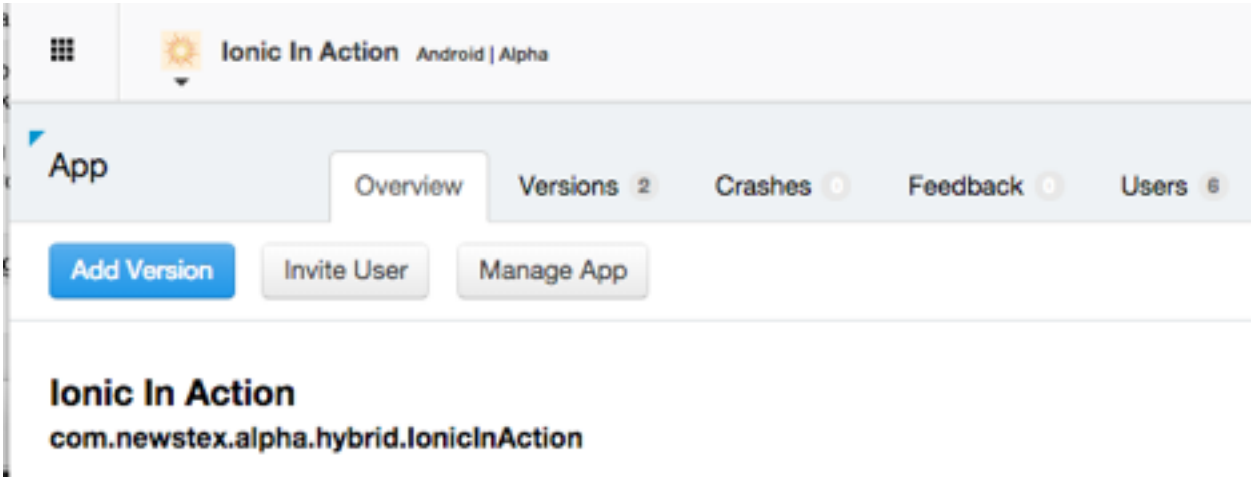


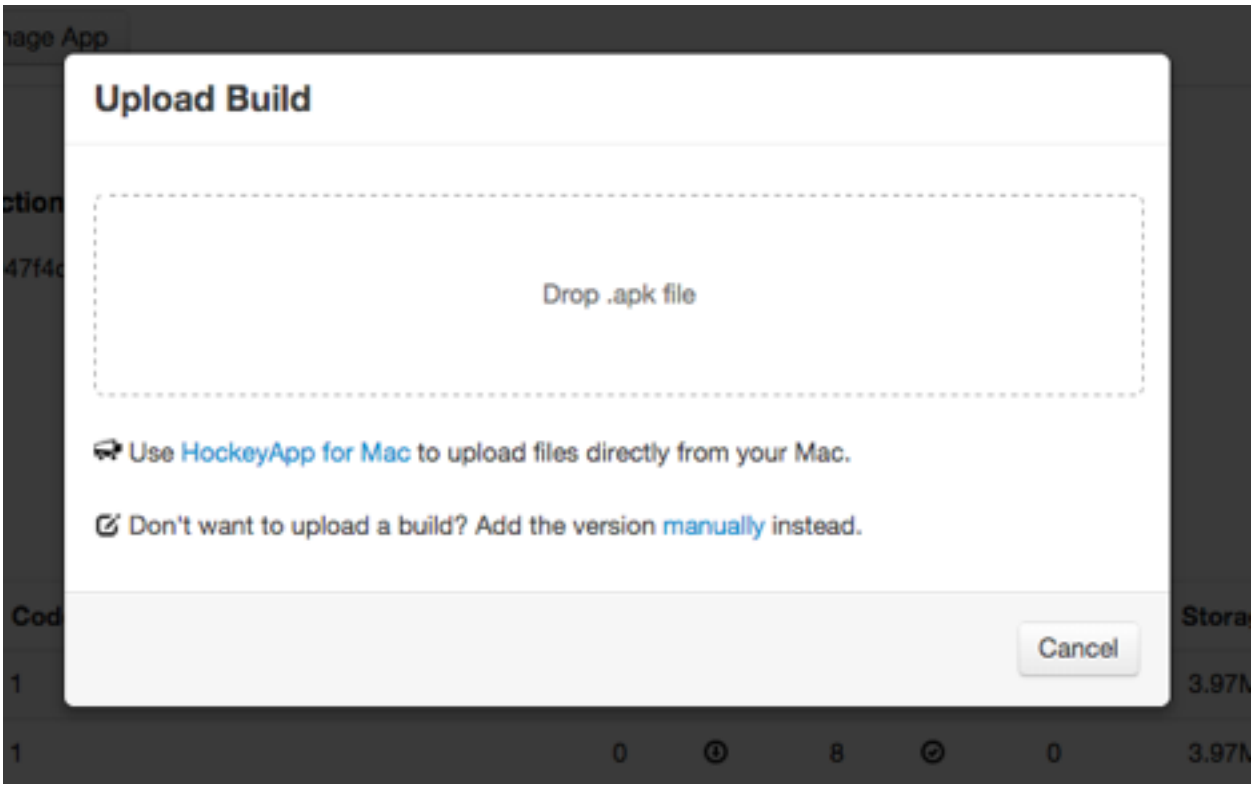
AndroidManifest.xml



Once we know for certain that the Package Name from Hockey matches the package value in the AndroidManifest.xml we can simply go to the App page on Hockey and click the “Add Version” button.

Remember the .apk file would be in platforms/android/ant-build





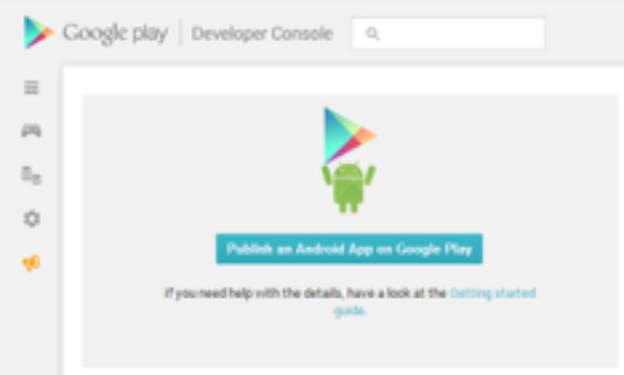
Then just follow the instructions to finish the upload process.

Distribute to GooglePlay

Now that we have our release APK ready for the Google Play Store, we can create a Play Store listing and upload our APK.

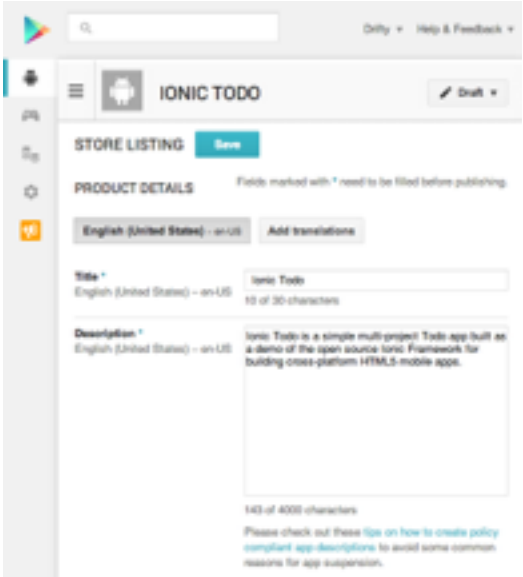
To start, you'll need to visit the Google Play Store Developer Console and create a new developer account. Unfortunately, this is not free. However, the cost is only $25.

Once you have a developer account, you can go ahead and click "Publish an Android App on Google Play" as in the screenshot below:



New google play app

Then, you can go ahead and click the button to edit the store listing (We will upload an APK later). You'll want to fill out the description for the app. Here is a little preview from when we filled out the application with the Ionic Todo app:



When you are ready, upload the APK for the release build and publish the listing. Be patient and your hard work should be live in the wild!

Note: see full guide at http://ionicframework.com/docs/guide/publishing.html