Publishing Your App

Joe Rogers
Android 310

Step 1: Create a developer account

- Cost is $25 and is a one time fee no matter how many apps you publish.
- As an individual you may want a custom google account so you are not using your personal account. (optional)
- If app charges or is using in-app purchases you will need to provide a physical address.

Step 1: cont.

- Update the display email which may be different than the Google account. If you have a registered website, use a variation.
- Review any other contact info to ensure it is correct as some will be automatically derived from the Google account.

Step 2: Update your application id

- Also known as package name, however the package may be different if using android studio and is controlled by the applicationId in build.gradle.

Once chosen and app is published the application id may NEVER be changed.

Step 2: cont.

- Consider a custom domain (no website needed), which ensures you uniquely control a top level domain. This adds a $10-25 (estimated) cost per year.
- com.example... is not legal and will be flagged if you try to use.
- Even if rewrite app, must use same name.

Step 2: cont.

- If consulting, please use the name of the company that hired you as part of the package name instead of your personal domain or your consulting companies domain. The app will be transferred to the hiring eventually.

Step 3: Create a signing key

- The signing key is the permanent key used to sign release builds.
- Back up the keystore and password. If lost, you will be forced to create a new app.
- Choose a significantly strong password. You may use one for the key store and a different one for the actual key.

Step 3: cont.

- Ensure you set “max” years for the expiration of the key. Odds are app will not be around that long, but you never know.
- DO NOT STORE key or password in code repository or together.
- If working on behalf of another party, have them create/manage key.

Step 4: Open source licenses

- Ensure your open source license requirements are met.
- This usually is something in the overflow/setting area of the app.
- Easiest way to find them is look at build.gradle for “app”. Everything there should be listed.

Step 4: cont.

- Watch for implicit dependencies.
  - Double check the external libraries directory for additional libraries via the “projects” view under project.
  - Also check the app/build/intermediates/exploded-aar directory as well.
  - Finally, check the lib directory for any libraries added manually.

Step 5: Create/update app listing

- If first time publishing, you need to create a new app listing.
- You will need to fill out all required text sections with something. If you are “releasing” ensure the text is how you want it seen by the public. Spell check, etc.

Step 5: App listing cont.

- As of May 5, you will need to fill out a form to compute your content rating.
  - Relatively painless, but ensure it is accurate.
- You could also record a video to help describe the app. See Chris Lacy's Action Launcher 3 as an example of an independent dev video.

Step 6: Screenshots

- At minimum you will need two screen shots. However, I'd add as many as needed to get a "feel" for the app.
- Use a helper app to "fix" the status bar. I suggest: [Developers] Clean Status Bar.
  - Essentially overlays app and draws over status bar to prevent random notifications, etc from being seen.
  - Also keeps time consistent.

Step 6: Screenshots cont.

- If your app installs on tablet, you will be reminded to include tablet screen shots.
- You may also be reminded to update your app to support tablets as well.
  - Failure to add tablet support will give you the branding “Designed for Phones” which will indicate the tablet experience may be sub-optimal.

Step 7: Turn off debug mode

- One of the biggest mistakes is leaving an app in debug mode.
  - Anyone with ADB can attach and “explore” the app.

    Also lets people poke around in your app private folders without rooting the device.
  - Android Studio will automatically disable this if you didn’t “manually” configure it in the first place.

Step 8: Disable logging

- Another common mistake is having the app continue to dump excessive logging from released apps.
- By default debug should be “disabled”, but if logs are still there it can be turned back on.
- Assume if it is logged it will be read, so nothing sensitive should be logged.

Step 8: Logging cont.

- Two ways to remove
  - Wrap logs with if (BuildConfig.DEBUG). Since the log can not be reached it will be compiled out in a release build. Especially useful if complex “building” of string to log is also required.
  - Use proguard to strip all log messages below warning and error. (Just be sure not to log sensitive data as warning or error).

Step 9: Configure Proguard

- Used to “strip” unused code, assets to reduce the apk size (and download time).
- Recommended if use support library or Play services or many 3rd party libraries.
- Will “obfuscate” your code by default.
- Homework solution apk drops 700 KB with proguard and obfuscation enabled.

Step 9: Proguard cont.

- With Android Studio libraries can provide their own rules automatically. This is how Google Play Services works.
- However, some libraries have not been updated and will list rules you need to append to your app.
- May need to manually add some rules.

Step 9: Proguard cont.

- Proguard is enabled via the minifyEnabled keyword in the app's build.gradle file.
- If using first time, should run full regression against app to verify all required classes were "kept".

Step 9: Proguard cont.

- If publishing an obfuscated app, be sure to “copy” the proguard mapping files to a safe location.
- Exist in app/build/outputs/mapping/release
- Only way to “rebuild” stack trace to diagnose release build crashes.

Step 10: Update version info

- For every release, you need to update the version code and version string.
- The version string is the “human” readable version users will see.
- The version number is an integer used by the play store to determine latest version.
  - Easiest is to increment by one***

Step 10: Version cont.

- The version code/string do not have to make sense with respect to each other.
- *** If you decide to support different apks for different versions of Android, you will need to consider a version code scheme to ensure the right version goes to the right users.
  - See resources for details.

Step 11: Choose Alpha/Beta/Release

- There are 3 options for publishing in the store. Alpha, Beta and Release.
  - Alpha/Beta require a Google group, or Google+ community to control access.
- Alpha
  - Use for QA, Devs, and key stakeholders. Perhaps build a week, etc. Especially good for “external” stakeholders if consulting.

Step 11: Alpha/Beta/Release cont.

- Beta
  - Good for early versions for public testing. Ie users that are willing to deal with more bugs for earlier opportunities to use app.
- Release
  - Standard releases. Ie anyone can find in store and install.

Step 12: Build it

- By default there is a “Release” and “Debug” build variants.
  - May add others. Especially if using alpha/beta release channels.
- I urge you to change the “debug” variant to append “.debug” to the application id and version name to help ensure you don’t ship a “debug” build to the store.

Step 12: Build Signing

- If you are doing a release build or any other variant that requires release keys, easiest way is to select Build->Generate Signed Apk… in Android Studio.
- If need command line support by default it will generate an “unsigned” apk, that you can manually sign.

Step 12: Build Signing

- Alternately, I have logic that will read a “file” or you could read the env to get the signing info. The file should be “secure”. This makes it convenient, but less secure.

Step 13: Test it

- Run through all the screens on the app, especially if proguard was run. This ensures nothing unexpected was “compiled” out.
- Consider running a few ui exerciser “monkey” runs as well as it will run app in ways not expected.

Step 14: Upload to store

- Upload your new APK to the play store.
- Once you “publish” it, it may take 4-6 hours before it is visible to the general user base.
- First publish likely will take longer than others.

Resources

Resources

- Publishing Overview
- Google Play overview and links
- Clean Status Bar App (for screenshots)
- Proguard Manual
- Multiple-APKs version codes