Guidelines for Publishing To Play Store

By Roy Koganti

Table Of Contents

General Launch Checklist For Android Apps

Before publishing apps on Google Play and distributing them to users, there are many steps you need to take to get the apps ready, to test them and to prepare your promotional materials. Below is a list of tasks that you'll need to complete before publishing your app on Google Play. he tasks are numbered to give you a rough idea of sequence but you can handle them in any sequence that works for you. Not all tasks will apply to every developer. Links to more detailed instructions are provided as well throughout.

1. **Get a Google Play Developer Account and add New application in Developer Console**
    a. Costs 25 dollars
    b. Register as a company rather than an individual if you are worried about any legal issues with IP
        i. Same process, but use your company's details when signing up
        ii. Register with a new google account, with company specific email
        iii. Use company credit card to pay
2. **Configure options for store listing**
    a. Game Title [<30 characters]
    b. Short Description [<80 characters]
    c. Full Description [<4000 characters]
        i. Shown only when user presses the read more option
    d. Categorization
        i. Select Application type
        ii. Select Category
        iii. Set your content rating [More on this later]
    e. Contact Details for your studio
        i. Website [Not needed]
        ii. Email [Required]
        iii. Phone [Not needed]
    f. Privacy Policy
        i. Optional
3. **Upload promotional materials and other graphic assets**
    a. Each asset has its own requirements that you can find on the store listing tab
    b. Hi-res App icon [Required]
        i. Design Guidelines
    c. Screenshots [Required]
        i. At least 2 overall are required
        ii. Max 8 screenshots for each supported device type
        iii. For app to be featured in designed for tablets section, need at least one 7-inch and one 10-inch screenshot
    d. Promo Video

        i. Add a youtube URL to a video showcasing features of your app
        ii. Make sure to use an individual video's YouTube URL, not a YouTube playlist or channel URL
        iii. Don't use an age-restricted video as your app's promo video
        iv. Use the full YouTube video link instead of a shortened link
    e. Featured Graphic [Required]
        i. Required in order to be featured anywhere within Google Play
        ii. Displayed at the top of store listing page in Play Store
        iii. A play button is shown over the feature graphic page if promo video exists
        iv. **Design Guidelines**
    f. Promo Graphic
        i. Used for promotions on older versions of Android OS (v4.0 and below)
    g. TV Banner [Required only for Android TV-enabled apps]
        i. App icon for Android TV
4. **Release Versioning for your app**
    a. Follow a common versioning system that consumers are used to
    b. MajorVersion.MinorVersion.Patch
5. **Determine Content Rating for your app**
    a. Fill out questionnaire to determine content rating for your app on Developer Console
    b. 4 levels : Everyone, Low , Medium and High Maturity
    c. Possible to get your ratings overwritten. If this happens, you can submit an appeal
6. **Determine Country Distribution**
    a. You can control which countries/territories your apps are distributed to
    b. Decide on which ones you want to exclude or include
    c. It would change localization needs and pricing settings
    d. **Supported locations for distribution**
7. **Decide if your app is free or priced**
    a. Free apps can be downloaded by any Android user in Google Play
    b. Paid apps can only be downloaded in countries that support paid downloads
    c. Free apps must remain free but paid apps can be changed to free apps
    d. For both free and paid apps, you can sell in-app products and subscriptions through In-app billing
    e. If your app is a paid app, or you are selling anything through it, need to set up a Google Payments Merchant account first
        i. Sign in to your Google Play Developer Console at **https://play.google.com/apps/publish**
        ii. Open financial reports on the side navigation
        iii. Click Setup a Merchant Account now
    f. Free apps automatically opt-in to **Android For Work**, but paid apps need to accept terms and conditions before they are distributed to Google Play for Work

8. **Consider using In-app billing or Android Pay**
    a. Google Play In-app Billing lets you sell digital content in your apps. Either one-time purchases or subscriptions
        i. **One-time Purchases Guide**
        ii. **Subscriptions Guide**
    b. Android Pay enables secure purchases of physical goods and services in app
9. **Set prices for your products**
    a. Can set prices for products in variety of currencies for different countries
    b. Different countries have different min and max values for price ranges
    c. All funds collected to currency of your merchant account using current exchange rates
    d. You receive 70% of payment, other 30% goes to Google
    e. Account for **Tax rate and Value-Added Tax (VAT)**
10. **Setting up Ads for your app**
    a. Specific whether your app contains ads or not. If your app contains ads, It will then have a “contains ads” label on store listing. Don’t lie, your app can get suspended from app store
    b. **Ad Policy**
11. **Distributing your app to special user programs**
    a. **Designed for Families**
    b. **Google Play For Work**
    c. **Google Play For Education**
        i. **Google for Education Guidelines**
12. **Start Localization**
    a. After deciding all the above, you can go about starting localization to ensure that your apps are internationalized. This requires a lot of time and effort so it is best to design for localization early on
    b. Localization involves not just translations but adding or removing features/content based on cultural differences
    c. Different ratings and legal issues in different countries
    d. Localizing strings, images and other resources in app
    e. Localizing your app’s store listing details on Google Play
    f. Localizing app’s graphic assets and promotional material
    g. **Detailed checklist**
13. **Confirm App Overall Size**
    a. Maximum size of APK is 100 MB
    b. Can add up to 2 APK **expansion files**, up to 2GB each for each APK
    c. If you have APK expansion files, what happens is the user downloads the APK first, and when they start the app for the first time, the expansion files are downloaded in order to run the app
14. **Confirm the App’s Platform and Screen Compatibility Ranges**
    a. **Device Dashboard** provide statistics of current device penetration of Android platform versions and screen sizes across all Android devices

b. Supporting Different Platform Versions
    i. Use stats from dashboard to decide what Android Platform versions you want to support
    ii. Good practice to support about 90% of active versions but target latest version
    iii. Specify Minimum and Target API Levels
c. Supporting Multiple Screens
    i. Use stats from dashboard to decide what screen sizes you want to support
    ii. Important to provide scalable support for multiple screen sizes

15. Make the APK

a. Build the APK, and sign it

16. Upload APK to play store by adding new Application

a. You can upload APK for Alpha or Beta testing, and send the link to alpha/beta testers
b. Click button for get license key (which you will need if it is a paid app, uses in-app billing or has APK expansion files)
c. If you are ready to release the app to the public, then upload it to production
d. Always build and test a Beta apk first just in case
    i. Beta ratings and reviews don't carry over to the production release

17. Final Checks

a. Review everything above
b. Make sure your app doesn't violate content policy guidelines
c. Click button near top right to publish your app
d. Takes a few hours to process

18. Updating or unpublishing your app

a. Updates
    i. You can upload a new APK, or make changes to store listing and click update button near top right
b. Unpublish
    i. Near the top of your app page in Developer Console, there is a option to unpublish app
c. Takes a few hours to process

19. Support Users after Launch

a. Check ratings and reviews frequently to be on the lookout for bugs or other issues
b. Make sure new platform versions don't break your app
c. Enter comprehensive summary of what each update contains

Steps to take in Unity before building your APK

If you are publishing your APK from Unity, then you will need to follow the steps below before building your APK and uploading it.

1. **Sign your APK before publishing it [Create keystore and key]**
    - **a.** A keystore is different from a key. A keystore can contain many keys. One good practice is create separate keystores for different categories of apps or different teams/groups. Then for each keystore, you can create multiple keys
    - **b.** Locate publishing settings under Player Settings for Android build
    - **c.** Tick the Create a new keystore
    - **d.** Type in a password and confirm it
    - **e.** Save your keystore by clicking the browse keystore button and save it somewhere you remember
    - **f.** Now select ‘create a new key’ under key alias
    - **g.** A new window opens. Enter all the necessary information
    - **h.** Alias can be any name you want to name it, usually the name of your app
    - **i.** Enter a password, and set validity to a large number of years, 50 is fine
    - **j.** Enter all other info as well and create the key
    - **k.** Every time you open the unity project, you need to select the keystore, and the key and enter the passwords before building the apk
    - **l.** After selecting the key, and building the APK, your new APK is now signed and ready to be published on Play Store

2. **Configuring the Player Settings for Android Build**
    a. Set the app icon to the same one you uploaded to Play Store
    b. Under “Other settings”, look at the “Identification” section
        i. Bundle Identifier
            1. Make it the same as the package name you chose on Play Services
        ii. Version*
            1. Specifies the build version number. Set this according to your versioning system for your APK. (1.xx , MajorVersion.MinorVersion.Patch is the recommended format)
        iii. Bundle Version Code
            1. An internal version number that isn’t shown to users. Only used to determine if one version is more recent than another. Each successive version you release should have a higher number
        iv. Minimum API Level
            1. Set the minimum API Level that phones need to support your app
    c. Under “Other settings”, look at the “Configuration” section
        i. Android Game

1. If enabled, built APK is marked as game rather than a regular app

ii. Install Location
    1. Specifies where your app installs to on the target device
    2. 3 Different settings: Automatic, Prefer External, Force Internal
        a. Automatic : Lets the OS decide, Users can still move app back and forth
        b. Prefer External: Install app to external SD card if possible, else to internal
        c. Force Internal: Always install app to internal memory, and users can't move it to external memory

3. **Remove debug info and logging info first**
4. **That's it, build your APK!**

Core App Quality

Android users expect high-quality apps, like the ones that are being provided to them by companies with a lot of resources. Below are basic criterias in various aspects that your app should meet. Before publishing your app, test it against the below to ensure they meet Android standards. Keep in mind, these should be the minimum level you strive to reach. There is a lot more that you could accomplish in each category to make your app better. Still, your app must meet the below standards at the very least, in order to be considered for promotional opportunities or to be featured in the Play Store.

1. **Visual Design and User Interaction**
    a. Standard Design
        i. App follows **Android Design** guidelines and uses **common UI Patterns and icons**
            1. App does not redefine the expected function of a system icon
            2. App does not replace a system icon with a completely different icon if it triggers the standard UI behavior
            3. If app provides a customized version of standard system icon, the icon strongly resembles the system icon and triggers standard system behavior
            4. App does not redefine Android UI patterns
    b. Navigation
        i. App supports standard system back button navigation and does not use any custom, on-screen back button prompts
        ii. All dialogues are dismissable using the back button
        iii. Pressing the home button at any point navigates to the Home screen of devices
    c. Notifications
        i. Multiple notifications are stacked into a single notification object
        ii. Notifications are persistent only if related to ongoing events (if music playing or phone call)
        iii. Notifications do not contain advertising or content unrelated to the core function of the app, unless user has opted in
    d. Action Bar
        i. Makes important actions prominent and accessible in a predictable way
        ii. Supports consistent navigation and view switching within apps
2. **Functionality**
    a. Permissions
        i. App requests only the absolute minimum permissions that it needs to support core functionality

ii. App does not request permissions to access sensitive data (such as Contacts or the System Log) or services that can cost the user money (such as the Dialer or SMS), unless related to a core capability of the app.

b. Install Location
   i. App functions normally if installed on SD card
   ii. Supporting installation to SD card is recommended for most large apps (10MB+). **App Install Location Guide** provides info about which types of apps should support installation to SD card

c. Audio
   i. Audio does not play when the screen is off, unless this is a core feature (for example, the app is a music player)
   ii. Audio does not play behind the lock screen, unless this is a core feature
   iii. Audio does not play on the home screen or over another app, unless this is a core feature
   iv. Audio resumes when the app returns to the foreground, or indicates to the user that playback is in a paused state

d. Screen Orientation
   i. App supports both landscape and portrait orientations (if possible)
   ii. Orientations expose largely the same features and actions and preserve functional parity. Minor changes in content or views are acceptable
   iii. App uses the whole screen in both orientations and does not letterbox to account for orientation changes
   iv. Minor letterboxing to compensate for small variations in screen geometry is acceptable
   v. App correctly handles rapid transitions between display orientations without rendering problems

e. User / App State
   i. App should not leave any services running when the app is in the background, unless related to a core capability of the app
   ii. App correctly preserves and restores user or app state
   iii. App preserves user or app state when leaving the foreground and prevents accidental data loss due to back-navigation and other state changes.
   iv. When returning to the foreground, the app must restore the preserved state and any significant stateful transaction that was pending, such as changes to editable fields, game progress, menus, videos, and other sections of the app or game
   v. When the app is resumed from the Recents app switcher, the app returns the user to the exact state in which it was last used
   vi. When the app is resumed after the device wakes from sleep (locked) state, the app returns the user to the exact state in which it was last used
   vii. When the app is relaunched from Home or All Apps, the app restores the app state as closely as possible to the previous state

viii. On Back keypresses, the app gives the user the option of saving any app or user state that would otherwise be lost on back-navigation

3. Performance and Stability

a. Stability
   i. App does not crash, force close, freeze, or otherwise function abnormally on any targeted device

b. Performance
   i. App loads quickly or provides onscreen feedback to the user (a progress indicator or similar cue) if the app takes longer than two seconds to load

c. Battery
   i. App supports power management features in Android 6.0+ (Doze and App Standby) properly.

d. Media
   i. Music and video playback is smooth, without crackle, stutter, or other artifacts, during normal app usage and load

e. Visual Quality
   i. App displays graphics, text, images, and other UI elements without noticeable distortion, blurring, or pixelation
   ii. App provides high-quality graphics for all targeted screen sizes and form factors, including for larger-screen devices such as tablets
   iii. No aliasing at the edges of menus, buttons, and other UI elements is visible
   iv. App displays text and text blocks in an acceptable manner
      1. Composition is acceptable in all supported form factors, including for larger-screen devices such as tablets
      2. No cut-off letters or words are visible
      3. No improper word wraps within buttons or icons are visible
      4. Sufficient spacing between text and surrounding elements

Reducing APK Size

Memory space on smartphones is competitive. As internal memory and size of SD cards increases, more people put all their music, videos and photos on their smartphones. So there is even less space for apps. Thus, the smaller your APK, the better it is for your users and also becomes a deciding factor as to whether to download your app, versus similar apps that take up less space.

1. **General Tips to Reduce APK Size**
    a. Reduce static footprint using name obfuscators like Proguard
    b. Remove debug information
    c. Use recommended media formats
        i. Images : PNG or JPEGs
        ii. Audio : AAC audio is recommended as better compression at same quality compared to mp3 or Ogg Vorbis
        iii. Video : Use H264 AVC
    d. Optimize PNG sizes without losing quality using online programs like PNGCrush
    e. Removed unused resources from project folder before building
    f. Avoid duplication
2. **Reducing APK size in Unity**
    a. Use Editor Log after build to determine space hungry assets
        i. select Open Editor Log from the small panel menu in the top right of the Console window
        ii. Provides a summary of assets broken down by type and lists them in order of size contribution
        iii. Typically, textures, music, and videos take up most space while scripts, levels and shaders are usually negligible
    b. Compress and/or reduce size of textures
    c. Compress meshes and animations
    d. Remove unused assets in Resources folder
        i. Unity automatically strips most unused assets during the build
        ii. However, unused scripts and unused assets inside a Resources folder are not removed
    e. Don’t worry about media formats
        i. Unity re-codes imported assets into its own internal formats so choice of source asset type is not relevant
        ii. Use format that is most convenient for you during development

Play Store Visibility

Your number one aim should be to give your app as much visibility on the play store as possible. With or without advertising for your app, this is essential for success.

1. **First 24 hour after release is**
    a. Must attain at least 100 - 200 downloads
    b. Time frame where you make it or break it
    c. If you don't hit this amount, chances are very low that Play Store shows your app to users when they browse similar categories
    d. Try and ask friends, families, use public groups on facebook and reddit in order to try and get downloads
    e. **Indie Game Promo Facebook Group** and **Android Gaming Subreddit**
2. **Need a unique, easily searchable name for your game**
    a. If it shares too many common words with other games, your game will likely filter to bottom of the list, unless you do really well on day one, or already have a lot of downloads
3. **Getting featured by Play Store**
    a. Must meet core app quality first
    b. If you have an inside contact whom you can get to review your app, great
4. **Getting a large wave of downloads a while after release**
    a. Maybe your game blows up on social media and gets a lot of attention
    b. Even if had very little downloads before that, getting a large wave of downloads in a small time will push your app up the search lists
    c. Users will be more likely to see your app when browsing similar apps or categories

Google Play Game Services Overview

Google Play Game Services provides cloud services for developers for both Android and iOS. It allows developers to add achievements, leaderboards, quests, events, saved game logs, and real-time/turn-based multiplayer functionality to your game. Players have to sign in with their google play account in order to use these services. It adds a lot of functionality to your games, so try and use it if possible.

1. **General Steps to configure game services for your game**
    a. Create an entry for your game and fill up all details needed
    b. Provide necessary credentials to authorize and authenticate your game, in order to access Google APIs
    c. Once your entry is created, link it to Android, iOS, and web versions of your game, so that players on these platforms see the same game details and share the same game services configurations
    d. Use an unique package name, and a good one as you can't change it
2. **Publish game services before publishing your APK, or else your APK can't access game services**
3. **Allow for Alpha and Beta testers so that you can test your APK first**
4. **Quota and Rate Limits**
    a. Review your application's daily quota by visiting it in Developer Console
    b. To view or change usage limits for your project, or to request an increase to your quota, **open the API Library** in the Developers Console
    c. You can set a max number of calls a user can make per second to prevent abuse of your app
    d. Requesting more quota probably won't be granted unless your game is experiencing truly exceptional usage and already taking steps to limit usage
5. **Leaderboards**
    a. Shows +/- range of scores around each player's score
    b. Daily, weekly, monthly, all-time scores shown
    c. Can format scores for different games
    d. Add min and max score limits
    e. Can give each leaderboard its own icon
    f. UI for leaderboards can be customized
6. **Achievements**
    a. Need at least 5 achievements to publish game services entry
    b. Every achievement needs an app icon before you can publish
    c. Each achievement a player gets gives points to player's Google Play account
    d. You can only assign a max of 1000 points total to all achievements per game
7. **Real-time Multiplayer**
    a. Create rooms where up to 8 players can join to play together online

    b. Either auto-match players, or let players host games and invite friends/random people
8. **Turn-based Multiplayer**
    a. Players share a single state of game, and only one player can has permission to modify shared state at any time
    b. 8 players per match, and can either auto-match or let players host games and invite people
9. **Events and Quests**
    a. Events allow you to collect data from player's behaviors and actions in-game and store them on the cloud
    b. You can code for what data you collect
    c. Use data gathered from events to help make quests for players
    d. Quests are basically in-game challenges that player has to achieve in time limit
    e. Events keep track of player progress in quests
10. **Saved Games**
    a. Save a player's progression to Google's servers
    b. Synchronizes saved data across multiple devices, between phones and tablets
    c. Allows player to start from where they left off
    d. Saved data isn't counted as part of developer's quota, but as part of player's Google Drive quota
    e. Developers only use their Google Drive API quota
    f. Saved files' binary data has size limit of 3 MB
11. **Quality Checklist**
    a. Each checklist item has one of 3 levels of importance
        i. **Required**. Minimum requirements that must be implemented for your game to be considered Google Play games services-compatible
        ii. **Best practices**. Strongly recommended implementation guidelines
        iii. **Good-to-have**. Suggested guidelines to help you create a distinctive player experience
    b. Follow at least best practices and required before advertising your game as Google Play games services-compatible

Google Play Game Services Unity Plugin

A plugin that allows you to access Google Play Games API from Unity's Social Interface system. The features available include achievements, leaderboards, signing in, saving data to the cloud, events, quests, nearby connects and turn-based/real-time Multiplayer. There is cross-platform support for all features except nearby connections (only Android).

Detailed Instructions on how to use the plugin can be found on the **github page**.

1. **System Requirements**
    a. Unity 5 or above
    b. To deploy on Android
        i. Android SDK
        ii. Android v4.0 or higher
        iii. Google Play Services library V8.4 or above
    c. To deploy on iOS
        i. XCode 6 or above
        ii. Cocoapods
2. **Configure your game services entry in Google Play Developer Console**
    a. Follow steps **here** to create an entry for your game and configure it
        i. Remember to use a good, unique package game. You can't change it after you set it
        ii. Remember to activate real-time or turn-based multiplayer if you want to use it
        iii. To get your SHA1 key, you need to run keytool on your keystore
            1. The SHA1 fingerprint used to create the linked Android app is from the keystore used to sign the Unity application
            2. Need to run in command line: keytool -exportcert \
            -alias *<your-key-name>* \
            -keystore *<path-to-production-keystore>* \
            -list -v
            3. Keytool.exe is found in your Java installation folder, for the JDK inside the bin folder. For me in windows it was : C:\Program Files\Java\jdk1.7.0_79\bin
            4. Either go to this folder in cmd and run it from there or add this path to Environment variable Path
3. **Add tester email addresses to the testing section of your game on Play Games Console**
4. **Installation of Plugin**
    a. Download the git folder, find the unity package in it, and import it into your Unity project
    b. Also switch build platform to Android in build settings

    c. Then follow the setup instructions for Android or iOS on the github page.
5. **Copy game resources from developer console to use them**
    a. For achievements, leaderboards, events and quests
    b. Instructions found on the **github page**