# Generating a Signed Release APK File in Android Studio

Once the development work on an Android application is complete and it has been tested on a wide range of Android devices, the next step is to prepare the application for submission to the Google Play App Store. Before submission can take place, however, the application must be packaged for release and signed with a private key. This chapter will work through the steps involved in obtaining a private key and preparing the application package for release.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Contents**  [[hide](javascript:toggleToc())]   * [1 The Release Preparation Process](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#The_Release_Preparation_Process) * [2 Changing the Build Variant](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#Changing_the_Build_Variant) * [3 Creating a Keystore File](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#Creating_a_Keystore_File) * [4 Generating a Private Key](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#Generating_a_Private_Key) * [5 Creating the Application APK File](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#Creating_the_Application_APK_File) * [6 Register for a Google Play Developer Console Account](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#Register_for_a_Google_Play_Developer_Console_Account) * [7 Uploading New APK Versions to the Google Play Developer Console](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#Uploading_New_APK_Versions_to_the_Google_Play_Developer_Console) * [8 Summary](http://www.techotopia.com/index.php/Generating_a_Signed_Release_APK_File_in_Android_Studio#Summary) | |  |  |
|  | | |

**The Release Preparation Process**

Up until this point in the book we have been building application projects in a mode suitable for testing and debugging. Building an application package for release to customers via the Google Play store, on the other hand, requires that some additional steps be taken. The first requirement is that the application be compiled in release mode instead of debug mode. Secondly, the application must be signed with a private key that uniquely identifies you as the application’s developer. Finally, the application package must be aligned. This is simply a process by which some data files in the application package are formatted with a certain byte alignment to improve performance.

Whilst each of these tasks can be performed outside of the Android Studio environment, the procedures can more easily be performed using the Android Studio build mechanism as outlined in the remainder of this chapter.

**Changing the Build Variant**

The first step in the process of generating a signed application APK file involves changing the build variant for the project from debug to release. This is achieved using the Build Variants tool window which can be accessed from the tool window quick access menu (located in the bottom left hand corner of the Android Studio main window as shown in Figure 54-1).

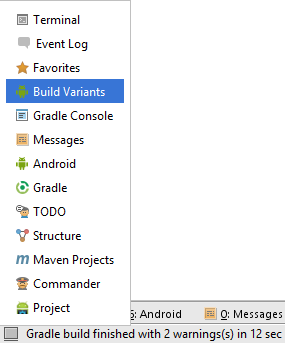
[](http://www.techotopia.com/index.php/Image:Android_studio_build_variants_menu.png)

Figure 54-1

Once the Build Variants tool window is displayed, change the Build Variant settings for all the modules listed from debug to release:

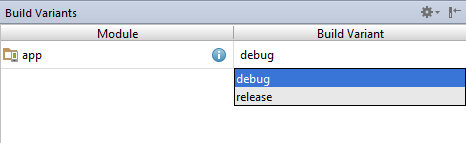
[](http://www.techotopia.com/index.php/Image:Android_studio_set_build_variant2.png)

Figure 54-2

The project is now configured to build in release mode. The next step is to configure signing key information for use when generating the signed application package.

**Creating a Keystore File**

To create a keystore file, select the Build -> Generate Signed APK… menu option to display the Generate Signed APK Wizard dialog as shown in Figure 54-3:

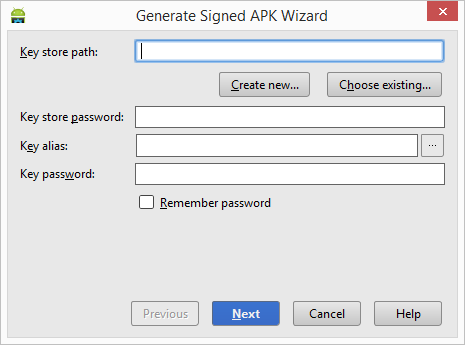
[](http://www.techotopia.com/index.php/Image:Android_studio_apk_build_keystore.png)

Figure 54-3

In the event that you have an existing release keystore file, click on the Choose existing… button and navigate to and select the file. In the event that you have yet to create a keystore file, click on the Create new… button to display the New Key Store dialog (Figure 54-4). Click on the button to the right of the Key store path field and navigate to a suitable location on your file system, enter a name for the keystore file (for example release.keystore.jks) and click on the OK button.

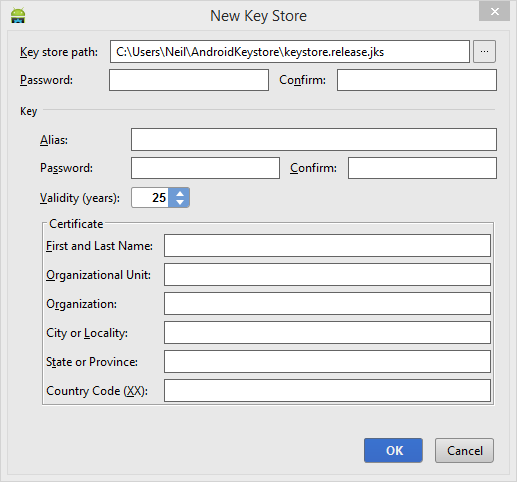
[](http://www.techotopia.com/index.php/Image:Android_studio_configure_release_key.png)

Figure 54-4

The New Key Store dialog is now divided into two sections. The top section relates to the keystore file. In this section, enter a strong password with which to protect the keystore file into both the Password and Confirm fields. The lower section of the dialog relates to the release key that will be stored in the key store file.

**Generating a Private Key**

The next step is to generate a new private key which will be used to sign the application package. Within the Key section of the New Key Store dialog, enter the following details:

* An alias by which the key will be referenced. This can be any sequence of characters, though only the first 8 are used by the system.
* A suitably strong password to protect the key.
* The number of years for which the key is to be valid (Google recommends a duration in excess of 27 years).

In addition, information must be provided for at least one of the remaining fields (for example your first and last name or organization name).

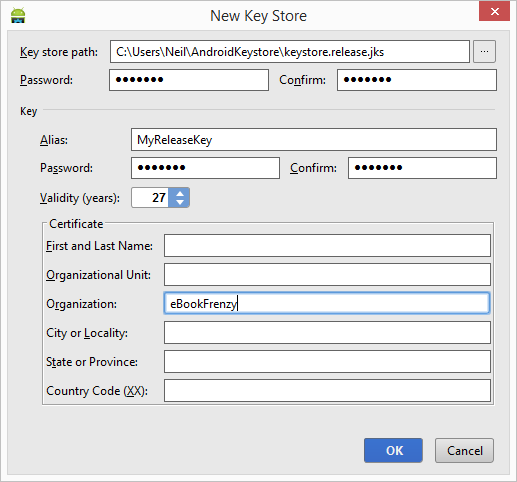
[](http://www.techotopia.com/index.php/Image:Android_studio_release_key_configured.png)

Figure 54-5

Once the information has been entered, click on the Next button to proceed with the package creation.

**Creating the Application APK File**

The next task to be performed is to instruct Android Studio to build the application APK package file in release mode and then sign it with the newly created private key. At this point the Generate Signed APK Wizard dialog should still be displayed with the keystore path, passwords and key alias fields populated with information:

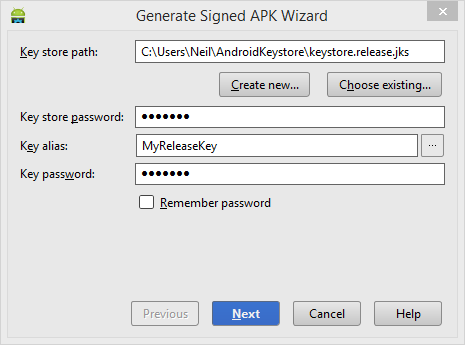
[](http://www.techotopia.com/index.php/Image:Android_studio_apk_generation_wizard.png)

Figure 54-6

Assuming that the settings are correct, click on the Next button to proceed to the APK generation screen (Figure 54-7). Within this screen, review the Destination APK path: setting to verify that the location into which the APK file will be generated is acceptable. In the event that another location is preferred, click on the button to the right of the text field and navigate to the desired file system location.

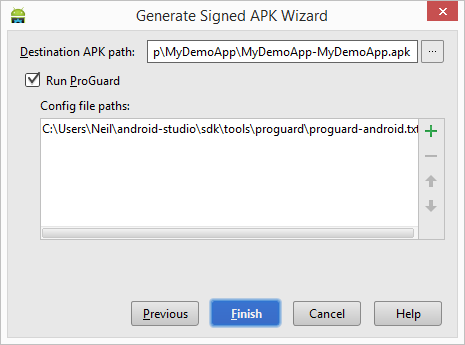
[](http://www.techotopia.com/index.php/Image:Android_studio_generating_apk_file.png)

Figure 54-7

Enable the Run ProGuard checkbox (ProGuard performs a series of optimization and verification tasks that result in smaller and more efficient byte code) before clicking on the Finish button. At this point the Gradle system will compile the application in release mode. Once the build is complete, a dialog will appear providing the option to open the folder containing the APK file in an explorer window:

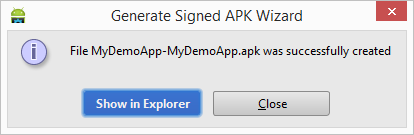
[](http://www.techotopia.com/index.php/Image:Android_studio_apk_built.png)

Figure 54-8

At this point the application is ready to be submitted to the Google Play store.

The private key generated as part of this process should be used when signing and releasing future applications and, as such, should be kept in a safe place and securely backed up.

The final step in the process of bringing an Android application to market involves submitting it to the Google Play Developer Console. Once submitted, the application will be available for download from the Google Play App Store.

|  |  |  |
| --- | --- | --- |
| [http://www.techotopia.com/cover_images/android_studio_front_cover_150x120.png](http://www.ebookfrenzy.com/ebookpages/android_studio_ebook.html)  [eBookFrenzy.com](http://www.ebookfrenzy.com) | **Purchase the full edition of this Android Studio Development Essentials publication in eBook ($9.99) or Print ($35.99) format** Android Studio Development Essentials Print and eBook (ePub/PDF/Kindle) editions contain 58 chapters. | [Buy eBook](https://www.e-junkie.com/ecom/gb.php?c=cart&i=PLDM-37&cl=56328&ejc=2) [Buy Print](https://www.createspace.com/4917513) |

**Register for a Google Play Developer Console Account**

The first step in the application submission process is to create a Google Play Developer Console account. To do so, navigate to <https://play.google.com/apps/publish/signup/> and follow the instructions to complete the registration process. Note that there is a one-time $25 fee to register. Once an application goes on sale, Google will keep 30% of all revenues associated with the application.

Once the account has been created, the next step is to gather together information about the application. In order to bring your application to market, the following information will be required:

* **Title – The title of the application.**
* **Description – Up to 4000 words describing the application.**
* **Screenshots – Up to 8 screenshots of your application running (a minimum of two is required). Google recommends submitted screenshots of the application running on a 7” or 10” tablet.**
* **Language – The language of the application (the default is US English).**
* **Promotional Text – The text that will be used when your application appears in special promotional features within the Google Play environment.**
* **Application Type – Whether your application is considered to be a game or an application.**
* **Category – The category that best describes your application (for example finance, health and fitness, education, sports etc.).**
* **Locations – The geographical locations into which you wish your application to be made available for purchase.**
* **Contact Information – Methods by which users may contact you for support relating to the application. Options include web, email and phone.**
* **Pricing & Distribution – Information about the price of the application and the geographical locations where it is to be marketed and sold.**

Having collected the above information and prepared the application package file for release, simply follow the steps in the Google Play Developer Console to submit the application for testing and sale.

**Uploading New APK Versions to the Google Play Developer Console**

The first APK file uploaded for your application will invariably have a version code of 1. If an attempt is made to upload another APK file with the same version code number, the console will reject the file with the following error:

You need to use a different version code for your APK because you already have one with version code 1.

To resolve this problem, the version code embedded into the APK needs to be increased. This is performed in the module level build.gradle file of the project, shown highlighted in Figure 54-9. It is important to note that this is not the top level build.gradle file positioned lower in the project hierarchy listing:

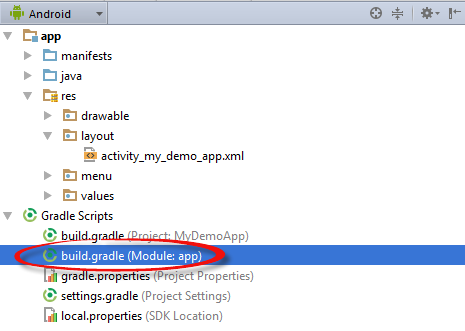
[](http://www.techotopia.com/index.php/Image:Android_studio_root_gradle_file2.png)

Figure 54-9

By default, this file will typically read as follows:

apply plugin: 'com.android.application'

android {

compileSdkVersion 21

buildToolsVersion "21.1.2"

defaultConfig {

applicationId "com.ebookfrenzy.mydemoapp"

minSdkVersion 21

targetSdkVersion 21

versionCode 1

versionName "1.0"

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

dependencies {

compile fileTree(dir: 'libs', include: ['\*.jar'])

To change the version code, simply change the number declared next to versionCode. To also change the version number displayed to users of your application, change the versionName string. For example:

versionCode 2

versionName "2.0"

Having made these changes, rebuild the APK file and perform the upload again.

**Summary**

Before an application can be submitted to the Google Play store, it must first be built in release mode, signed with a private certificate and the resulting APK package file subjected to a process referred to as alignment. As outlined in this chapter, all of these steps can be performed with relative ease through the use of the Android Studio build system.