

## 2 Ways to find appPackage and appActivity name of your App

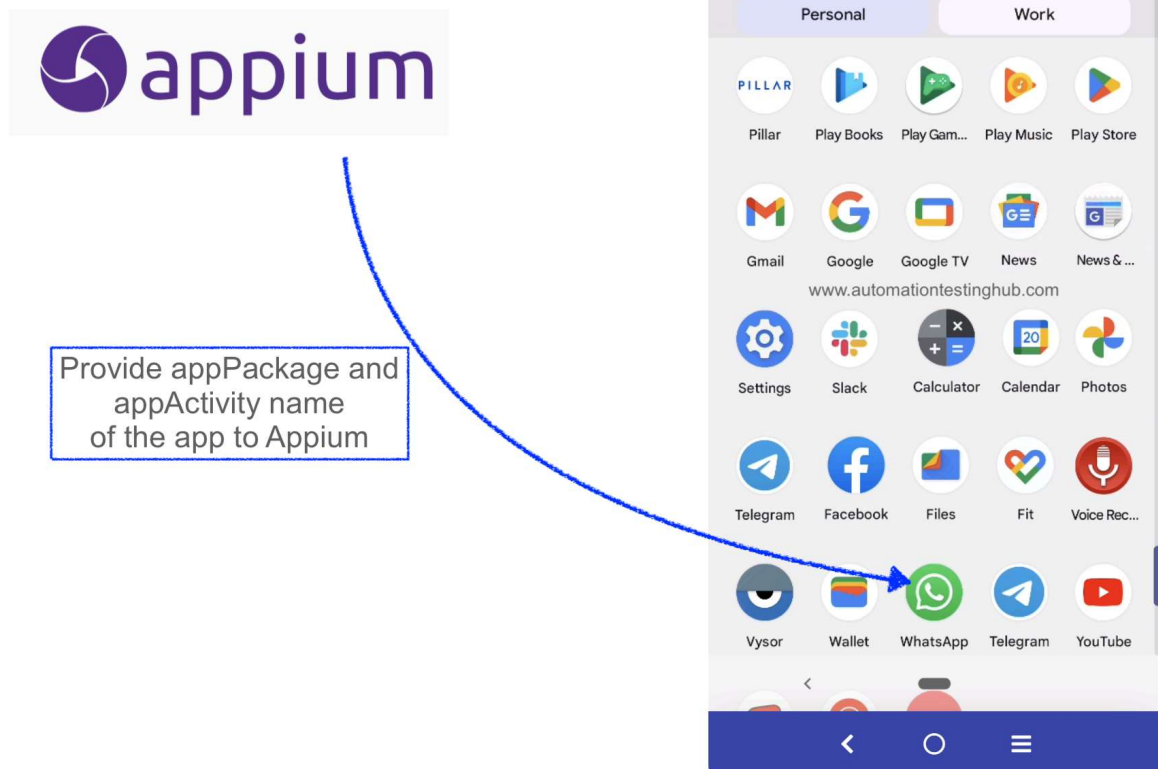


When you write test scripts with Appium, it usually involves launching an app first and then performing some actions on it. For example, let us suppose that you are testing WhatsApp. And your test case is to verify that you can send and receive messages. To test this, your code would first launch WhatsApp and then it would verify your test case.

But how do you launch WhatsApp with Appium? Or to put this question in a different way, how would Appium recognize which app on your phone is WhatsApp? You might have tens of apps installed on your mobile device. Hence, Appium needs to identify the correct app and then open it.

Let us first see how you would do this manually? You would scroll through the list of apps installed on your phone, and then you will open the app which has the name 'WhatsApp'. Well, Appium also follows a similar process. When you write your code, you would need to provide the name of the app, and Appium would then launch that app.

Things get a little technical here. You just can't provide the app name as 'WhatsApp' in your Appium code. Internally, all the mobile apps use a different technical name. You would need to provide this technical name (also known as package name). Together with this package name, you will also need to provide the activity name of the app.



**This article lists down 2 different methods using which you can find appPackage and appActivity names of your app under test.** You can use any of these methods to find out the package and activity names of your app. Before we start with these 2 methods, let's first get some more detail about appPackage and appActivity.

## What is appPackage and appActivity name ?

### appPackage:

**In very basic terms, appPackage is the technical name of the app provided by its developers.** It's actually the top level package under which all the code for the app resides.

**For example,** appPackage for 'YouTube' for Android is 'com.google.android.youtube'. For Facebook, this name is 'com.facebook.katana' and for WhatsApp, the appPackage is – 'com.whatsapp'. So if you want to launch Facebook from Appium, you would need to provide its name as 'com.facebook.katana' in Appium.

### appActivity:

**Again, speaking in very basic terms appActivity refers to the different functionalities that are provided by the app.**

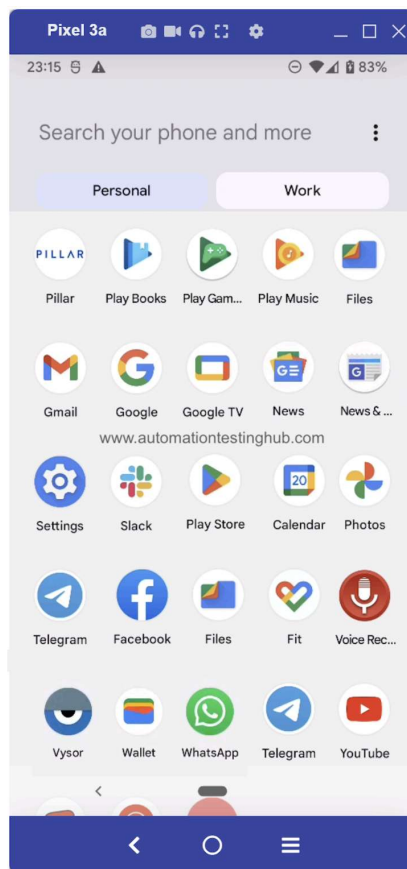
**For example,** WhatsApp provides multiple functionalities such as chats, status, calls, setting profile photo, etc. Each of these functionalities are represented by an appActivity.

Together with these activities, every app has a main activity which is sort of the main screen you see when you launch the app. For WhatsApp, it is the Chats window, and for Facebook it would be the Wall. When you launch the app with Appium, it needs to know which activity has to be launched. And you would need to provide the main activity name (the activity which represents the app's main screen)

With this basic and important understanding on about app package and app activity, let us start with the different methods with which you can identify this information of your app.

## Which app are we going to use in our Tutorial series?

With our [Appium Tutorial](#) series, we are trying to keep things very simple so that its easier for Appium beginners to follow our articles. Keeping this in mind, we use the **Google Play Store** and find its appPackage & appActivity name. The main reason for using this app is that it's available on all android mobile devices.



Let us now start with the first method on **how to identify the appPackage and appActivity name for Play Store app.**

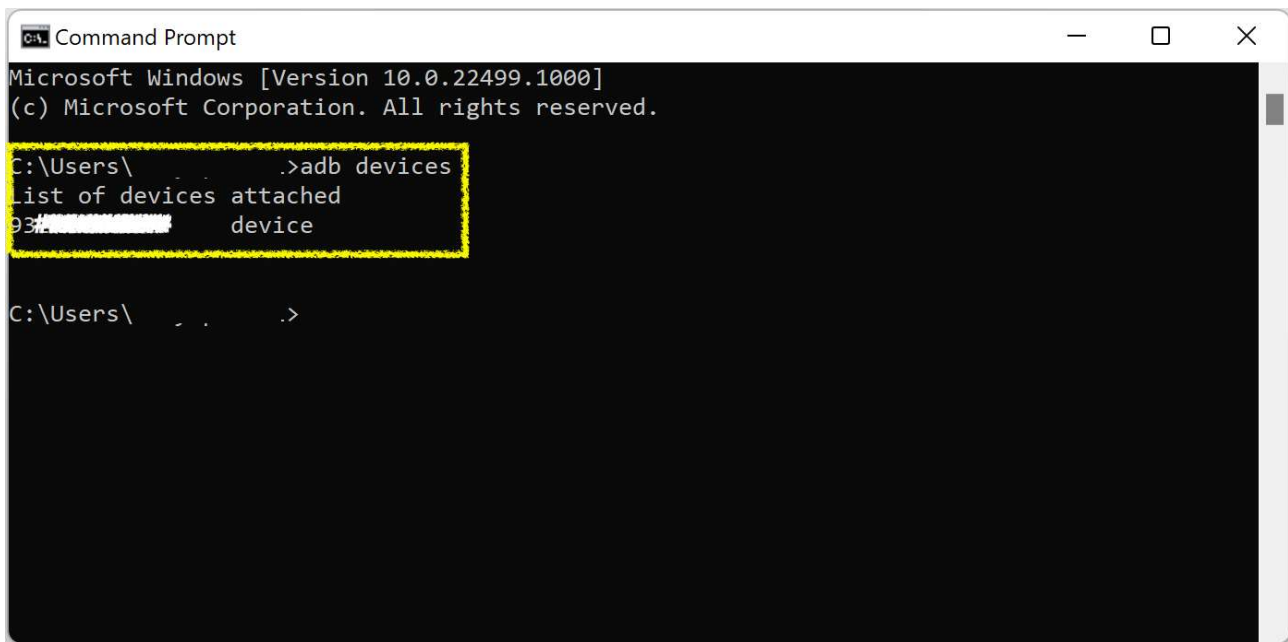
## Method 1: Using 'mCurrentFocus' or 'mFocusedApp' in Command Prompt

**Step 1:** Unlock your mobile device and connect it to your computer using USB cable

**Step 2:** Open Command Prompt and run '**adb devices**' command. We are running this command to just make sure that your mobile is properly connected.

**Step 3:** Once you run '**adb devices**' command, you should see that it displays the list of attached devices as shown in the below image (the actual device name that you see would be different based

on what mobile phone you use) –

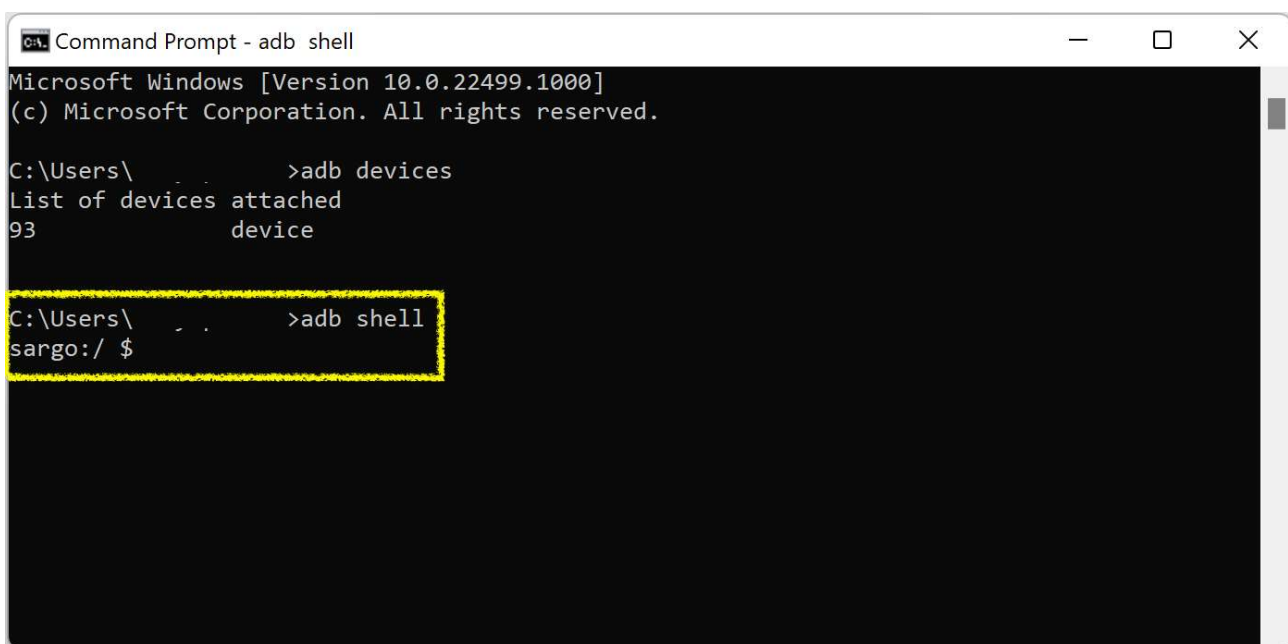


```
Command Prompt
Microsoft Windows [Version 10.0.22499.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\>adb devices
List of devices attached
93 device

C:\Users\>
```

**Step 4:** Run ‘adb shell’ command. After running this command, the command prompt should look something like this –

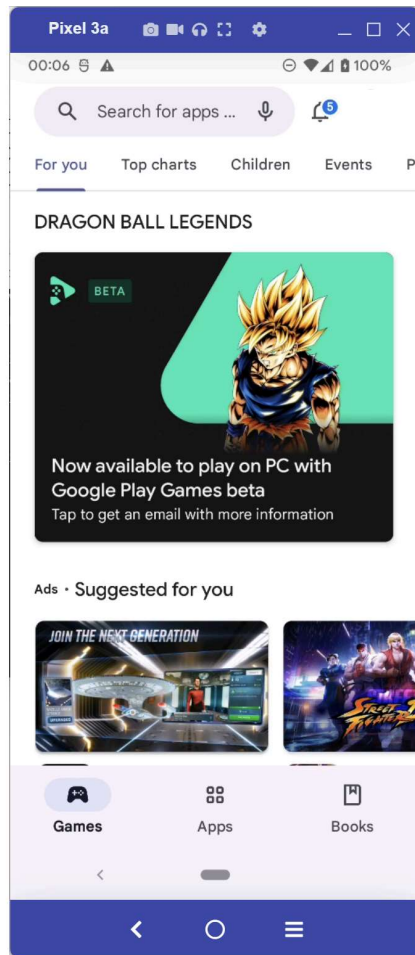


```
Command Prompt - adb shell
Microsoft Windows [Version 10.0.22499.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\>adb devices
List of devices attached
93 device

C:\Users\>adb shell
sargo:/ $
```

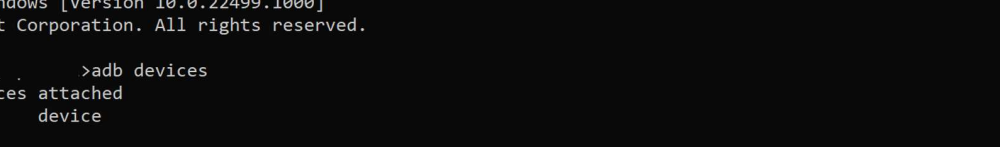
**Step 5:** Now in your mobile phone, open the app for which you want to find the appPackage and appActivity. Since we are doing this for Play Store, hence we will open “Play Store” on our mobile phone.



**Note:** Please make sure that you open the app before going to the next step, because command in the next step would provide the details only for the app which is currently in focus.

**Step 6:** Now run this command: **dumpsys window displays | grep -E 'mCurrentFocus'**

**Step 7:** The above command would display the details of the app which is currently in focus. From that, you can figure out the appPackage and appActivity name as per the below image –



The screenshot shows a Windows Command Prompt window titled "Select Command Prompt - adb shell". The window contains the following text:

```
Microsoft Windows [Version 10.0.22499.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\...>adb devices
List of devices attached
93          device

C:\Users\...>adb shell
sargo:/ $ dumsys window displays | grep -E 'mCurrentFocus'
mCurrentFocus=Window{90e37bb u0 com.android.vending/com.android.vending.AssetBrowserActivity}
sargo:/ $
```

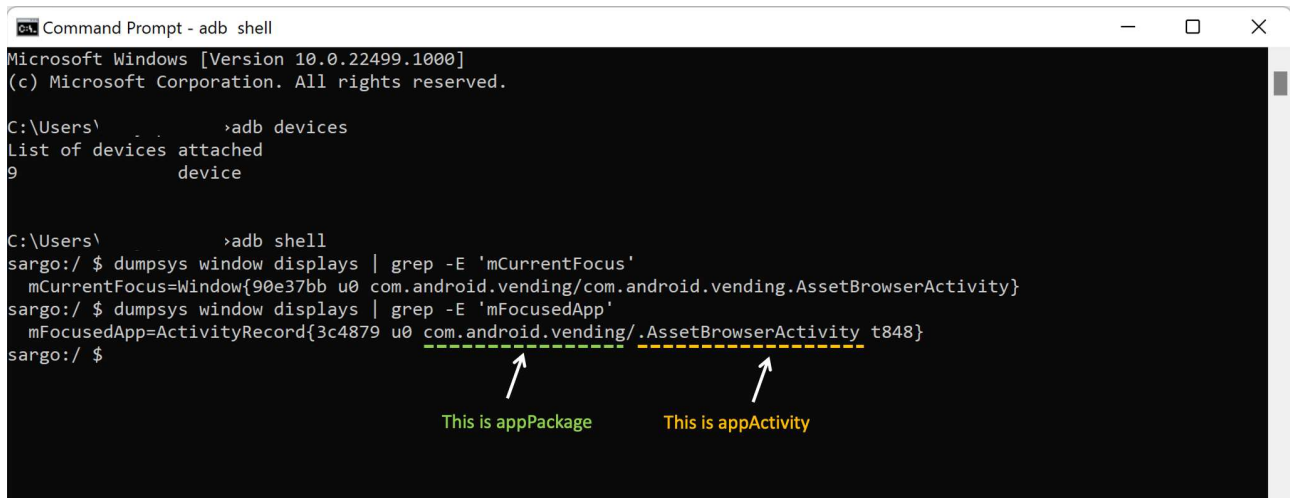
Two white arrows point to the package name and the class name in the output line. The first arrow points to `com.android.vending` and is labeled "This is appPackage" in green text. The second arrow points to `com.android.vending.AssetBrowserActivity` and is labeled "This is appActivity" in yellow text.

appPackage starts with com. and ends before backslash (/). So from the above image, appPackage name is – **com.android.vending**

appActivity starts after the backslash (/) and goes till the end. From the above image, appActivity name is – **com.android.vending.AssetBrowserActivity**



**Step 8:** There is one more similar command that provides the appPackage and appActivity name. This command adds some additional details before and after the package name & activity name, but you can still try it out just to verify that the results from the above command are same. This command is – **dumpsys window displays | grep -E 'mFocusedApp'** and the output of this command is shown below –



```

C:\Users\ >adb devices
List of devices attached
9 device

C:\Users\ >adb shell
sargo:/ $ dumpsys window displays | grep -E 'mCurrentFocus'
mCurrentFocus=Window{90e37bb u0 com.android.vending/com.android.vending.AssetBrowserActivity}
sargo:/ $ dumpsys window displays | grep -E 'mFocusedApp'
mFocusedApp=ActivityRecord{3c4879 u0 com.android.vending/.AssetBrowserActivity t848}
sargo:/ $
  
```

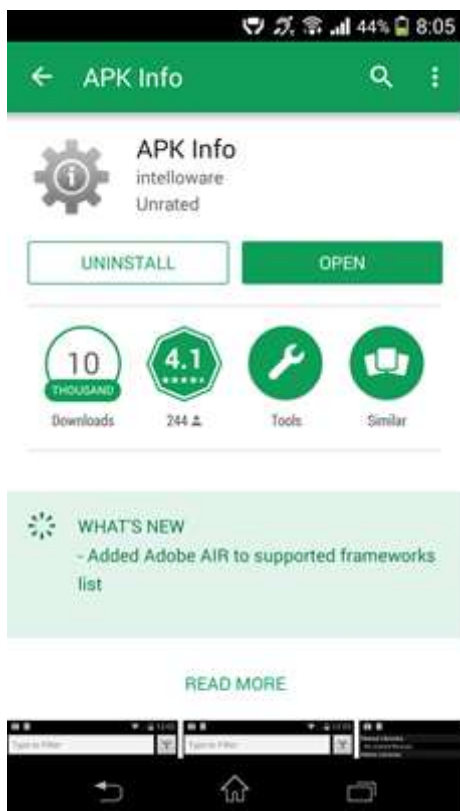
↑ This is appPackage
↑ This is appActivity

In this example, where we used mFocusedApp, appActivity name is shown as a relative name, i.e., it doesn't start with com. In such cases, you would need to add com.... at the beginning to get the complete activity name. So in our case, complete activity name for **.AssetBrowserActivity** would be **com.android.vending.AssetBrowserActivity**

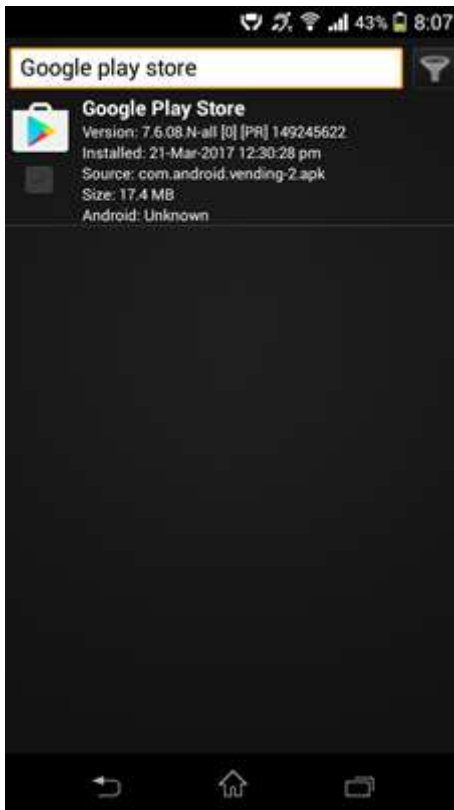
## Method 2: Using APK Info app

APK Info is an app which you can download from Play Store, and it will provide the appPackage and appActivity name of any app which is installed on your mobile device.

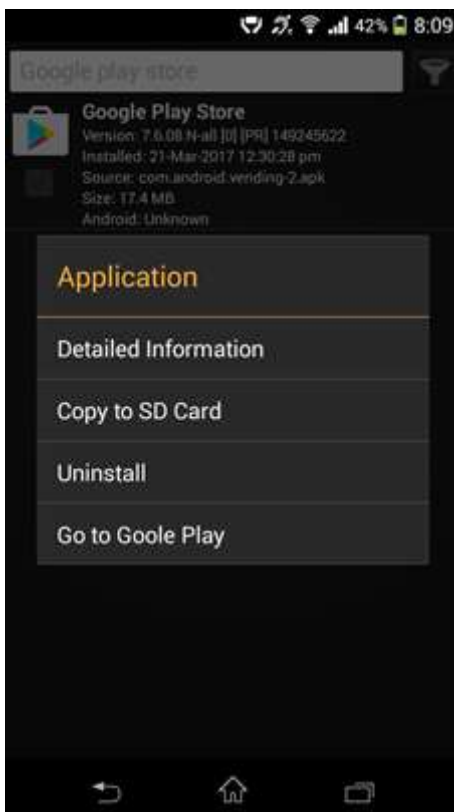
**Step 1:** Download “APK Info” app from Google Play Store on your android mobile.



**Step 2:** Once you have successfully installed APK Info app, open it and check that it lists down all the apps that you have on your phone. Then search for “Google Play Store” in the search pane as shown below

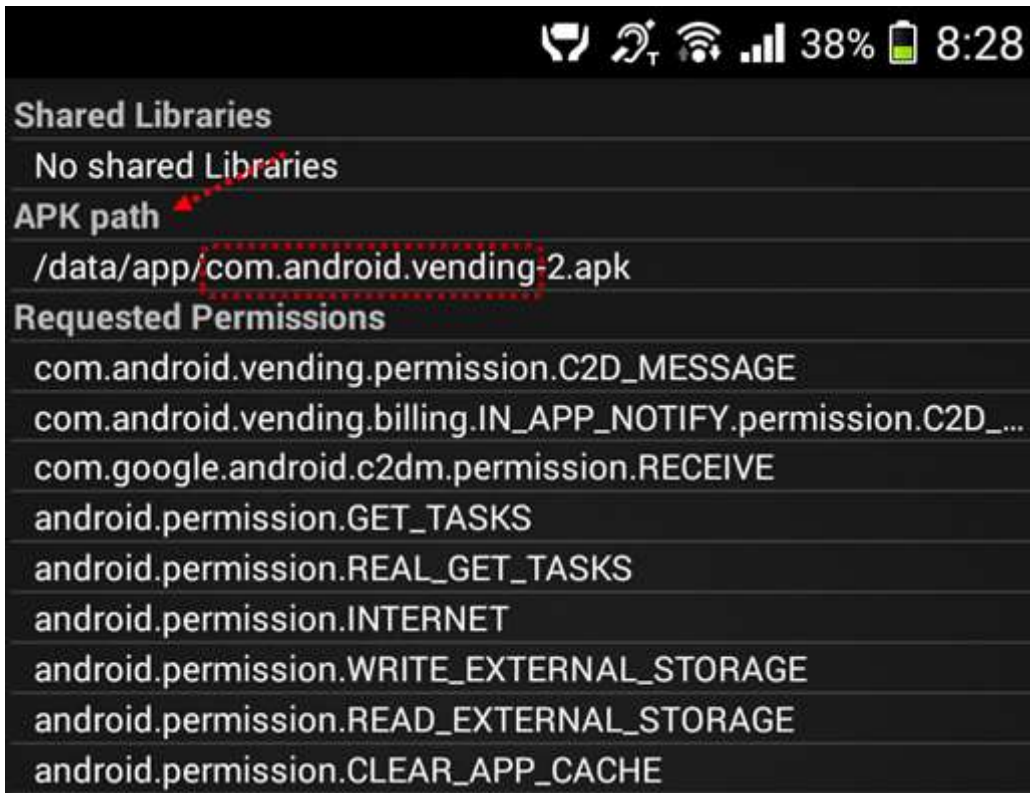


**Step 3:** Long press on the “Google Play Store” application icon inside the APK Info app till it displays the list of options as shown below –



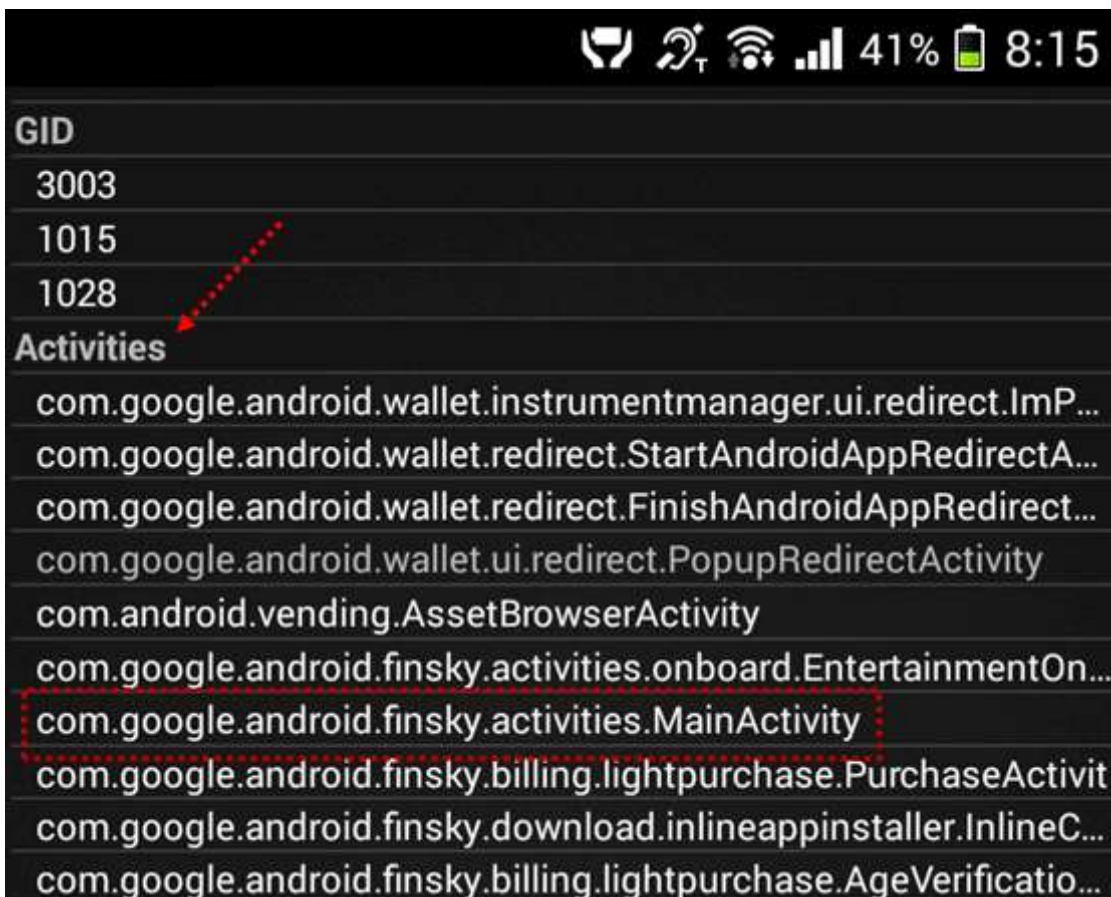
**Step 4:** Click the option “Detailed Information” option. It would show the detailed log for the app.

Here, check the **APK path** section. This sections displays the “appPackage” name as highlighted in red block in the below image –



**Note:** Skip any number at the postfix of the name (eg: here its “-2”). So, the appPackage name in this case is – **com.android.vending**

**Step 5:** Then to find the appActivity name of the app, scroll down to the sub-section “Activities”. This sub-section displays all the activities that are available for the app. From this list, you have to look for the activity which has “MainActivity” or “Main” or “Login” in the activity name.



Here “**com.google.android.finsky.activities.MainActivity**” is the appActivity name for the Play Store app.



Since Play Store is a full fledged app, so it contains a lot of activities. However, if you are testing a small app or some app which is in development phase, then it would not contain these many activities. So it would be easier to identify the main activity in that case. If you still find it difficult to identify the main activity, then you can always check back with your developers or use the first method that we have provided in this article.

With this, we complete our article on identifying appPackage and appActivity name for the app you want to test. Let us know if you face any issues while identifying these properties of any particular app with these methods. We would also love to hear from you, if you have any feedback for us, or if you have any other way which can help identify these properties.

We are continuously adding more articles to our tutorial series. You can check it out here – [Appium Tutorials](#)