**How to launch native installed app in Android**

1. First connect your real devices and check in terminal ‘adb devices’
2. Open your android installed app which you want to open
3. Open terminal and write the following command:

***adb shell dumpsys window windows | grep -E 'mCurrentFocus|mFocusedApp'***

// you will see this outputs:

mCurrentFocus=Window{a84a1c6 u0 com.android.chrome/com.google.android.apps.chrome.Main}

mFocusedApp=AppWindowToken{5ed34ea token=Token{bba54d5 ActivityRecord{cc7a58c u0 com.android.chrome/com.google.android.apps.chrome.Main t50}}}

***[NOTE: RED PART IS PACKAGE NAME AND YELLOW PART IS ACTIVITY NAME]***

1. Set your desired capabilities class:

**public** **static** AndroidDriver<AndroidElement> appConfig() **throws** MalformedURLException

{

DesiredCapabilities cap = **new** DesiredCapabilities();

cap.setCapability(MobileCapabilityType.***PLATFORM\_NAME***, "Android");

cap.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "Nexus\_5X\_API\_24");

cap.setCapability(AndroidMobileCapabilityType.***APP\_PACKAGE***, "com.android.chrome");

cap.setCapability(AndroidMobileCapabilityType.***APP\_ACTIVITY***, "com.google.android.apps.chrome.Main");

AndroidDriver<AndroidElement> driver=**new** AndroidDriver<>(**new** URL("http://127.0.0.1:4723/wd/hub"),cap);

**return** driver;

}

}

**Method 1: Using ‘mCurrentFocus’ or ‘mFocusedApp’ in Command Prompt**

You can run this command in command prompt, and **it will provide the appPackage and appActivity name of the app which is currently in focus.** Before you use this command, make sure that you complete the following pre-requisites –

**Note:** The pre-requisites are already covered in our [Appium Tutorial](http://www.automationtestinghub.com/appium-tutorial/) series. So you can skip these, if you have followed our Tutorial series from beginning. Otherwise, it would be a good idea that you complete the below pre-requisites first:

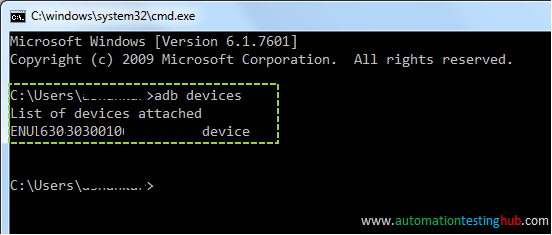
* You should have [downloaded and installed Android SDK](http://www.automationtestinghub.com/download-and-install-android/)
* You should have checked if you need to [install any additional SDK packages](http://www.automationtestinghub.com/install-additional-android-sdk-tools/)
* You should have [set up Android Environment Variables](http://www.automationtestinghub.com/setup-android-environment-variables/)

Let us now start with the steps that you need to follow to find appPackage and appActivity name using the first method.

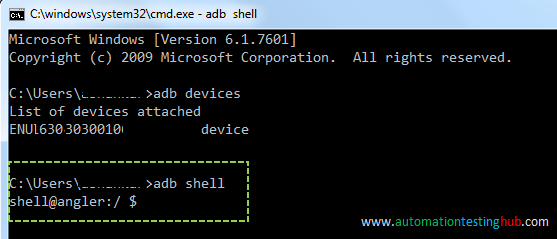
**Step 1:**Unlock your mobile device and connect it to your computer using USB cable

**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.

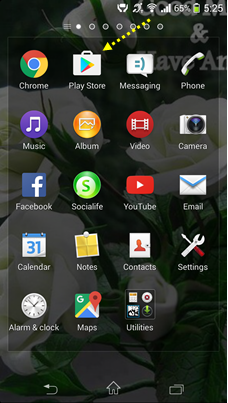
**Step 3:**Once you run **‘adb devices’** command, you should see that it displays the list of attached devices as shown in the below image (the actual device name that you see would be different based on what mobile phone you use) –



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



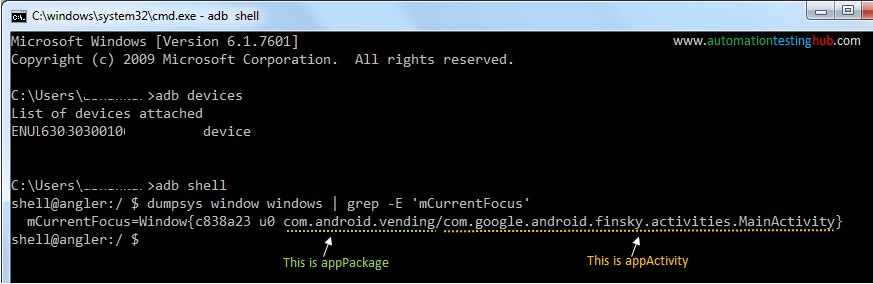
**Step 5: Now in your mobile phone, open the app for which you want to find the appPackage and appActivity.** Since we are doing this for Play Store, hence we will open **“Play Store”** on our mobile phone.



**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 windows | grep -E ‘mCurrentFocus’**

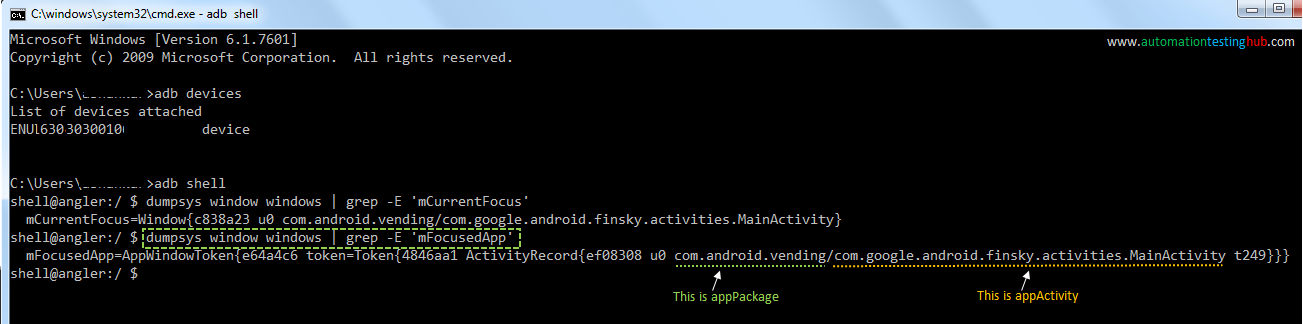
**Step 7:**The above command would display the details of the app which is currently in focus. From that, you can figure out the appPackage and appActivity name as per the below image –



appPackage starts with com. and ends before backshash (/). 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.google.android.finsky.activities.MainActivity**

**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 windows | grep -E ‘mFocusedApp’** and the output of this command is shown below –



This completes our first method of identifying appPackage and appActivity name. However, there is one important point which you should keep in mind. For some apps, the appActivity name would shown as relative name in command prompt, i.e., it would not start with com. In such cases, you would need to add com…. at the beginning on your own.

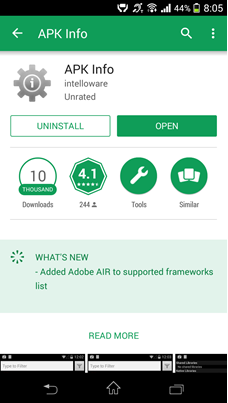
**For example,** consider some app which shows **com.myapp/.mainActivity** in command prompt when you run the above commands. In this case, you will notice that the appActivity starts with a dot (which is a relative name). So you would need to add com… at the beginning. After adding the complete appActivity name that you will use would be – **com.myapp.mainActivity**

**Method 2: Using APK Info app**

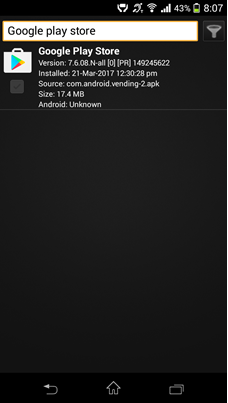
APK Info is an app which you can download from Play Store, and **it will provide the appPackage and appActivity name of any app which is installed on your mobile device.**

Let us now start with the steps that you need to follow to find appPackage and appActivity name using the second method.

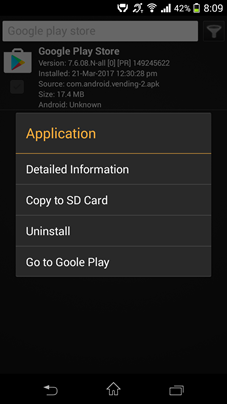
**Step 1:**Download “APK Info” app from Google Play Store on your android mobile.



**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

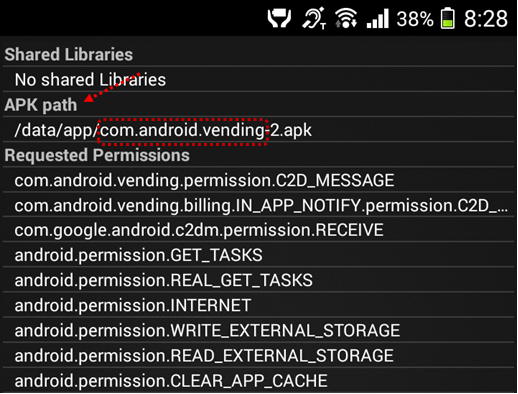


**Step 3:**Long press on the “Google Play Store” application icon inside the APK Info app till it displays the list of options as shown below –



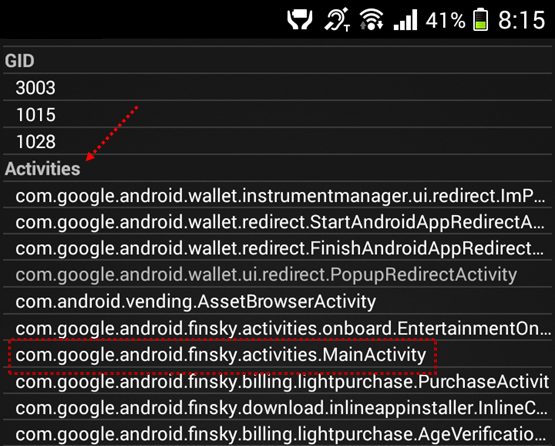
**Step 4:**Click the option “Detailed Information” option. It would show the detailed log for the app.

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.