Top of Form

Bottom of Form

**Appium Desktop App – Download and Install**

Appium Desktop in the new open-source GUI app that helps you access the Appium Server using a graphical interface. From an Appium Beginner’s perspective – if you want to write test scripts with Appium, you would need 2 things. First, you would need to start the Appium Server. And second, you would also need a mechanism using which you can identify the controls (buttons, text boxes etc) in the mobile app which you want to automate. Appium Desktop helps you perform both these tasks –

* Appium Desktop provides a GUI for the Appium Server. Using this GUI tool, you will be able to start and stop the Appium server, and also see the logs
* It also provides an Inspector using which you can get the properties of the elements in your mobile app

**Please Note:** Appium comes in both GUI and non-GUI versions (non-GUI versions can be operated from the command prompt). We will be working with the GUI version in our [Appium Tutorial](http://www.automationtestinghub.com/appium-tutorial/" \t "_blank) series, as its much easier to install and use as compared to the non-GUI version. We will encourage you to check out the non-GUI version as well. You can go to this link and read about its features and how its installed – [Download Appium (Non GUI version)](http://www.automationtestinghub.com/download-and-install-appium-1-6/)

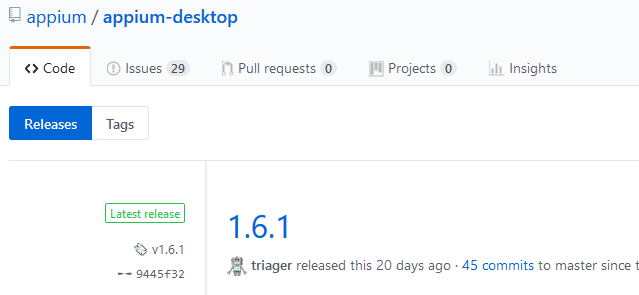
**Download Appium Desktop**

Follow the steps given below to download the latest version of Appium Desktop –

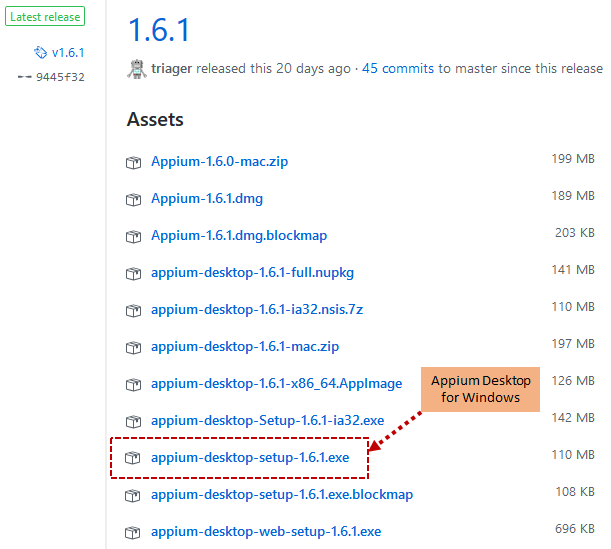
**1.**Open Appium Desktop’s GitHub page – <https://github.com/appium/appium-desktop/releases>

**2.**This page lists down all the versions of Appium Desktop. You will have to download the latest version which is shown at the top of the page. The latest version on the website is **v1.6.1** (as on 30 May 2018).

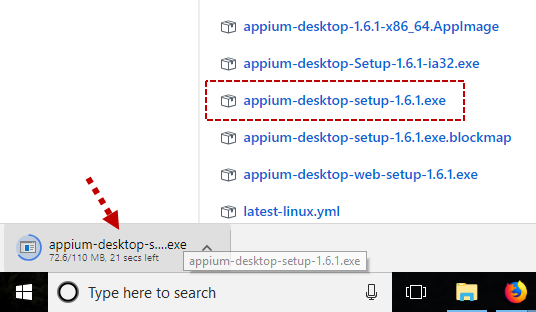
**Please Note:**New beta versions are released very often. Even if you see a newer version on Github, please go ahead and download that version. The steps given in this article would remain the same (only the screenshot contents might change)



**3.**In the Downloads section of the latest release, you will find different packages for Windows, Mac, Linux etc



**4.**Click on the windows version to download and save it on your machine



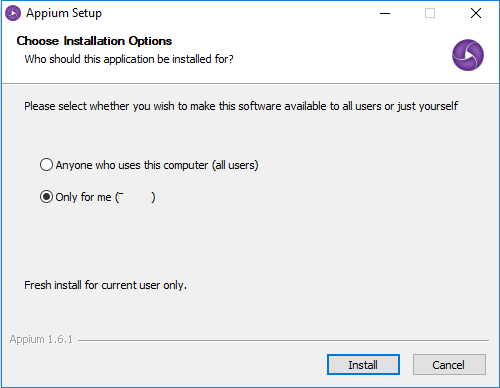
With this, we complete the download process for Appium Desktop. Let us now install it on our machine.

**Install Appium Desktop**

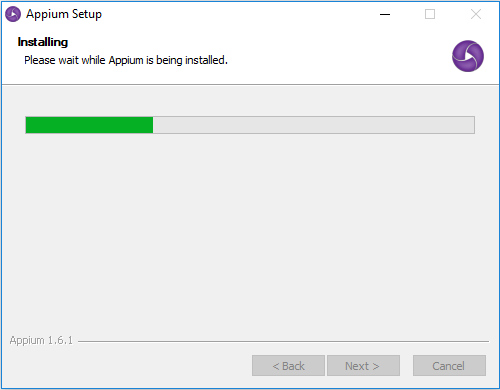
Use the steps given below to install Appium Desktop

**1.**Open the folder location where you have downloaded Appium Desktop and double click on the exe to start the installation process

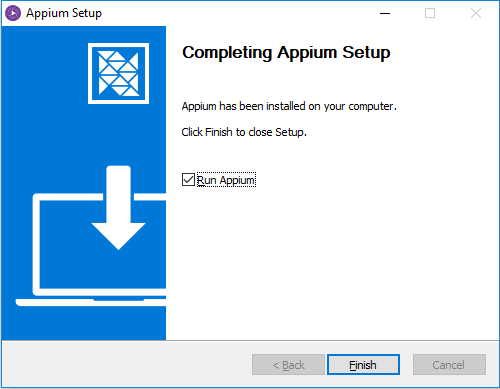
**2.**Appium will first ask you Installation Options. Leave the already selected option as it is and then click on Install button



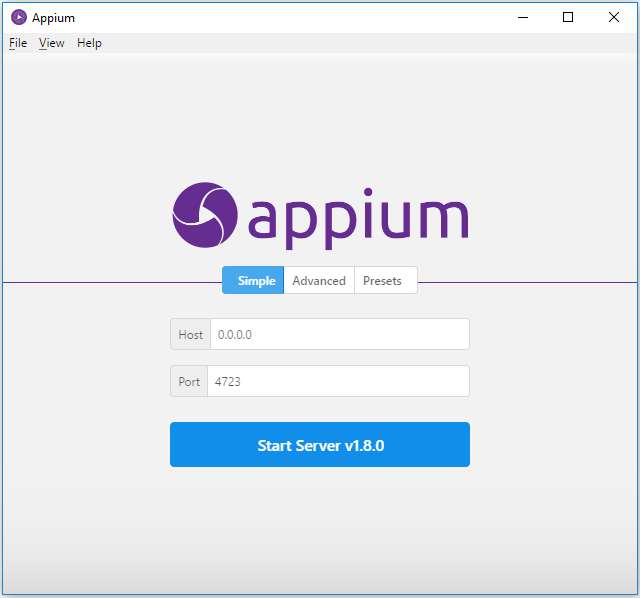
**3.**Once the installation process starts, you will see the Appium Setup window as displayed below



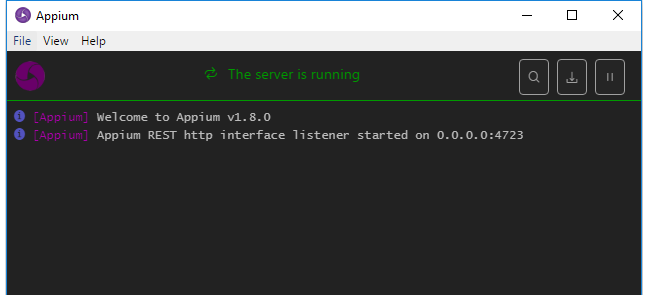
**4.**Once installation is complete, you will see the below Appium Desktop window



**5.**Now, leave the **Run Appium** checkbox selected and click on **Finish** button. Appium Desktop Start screen will be displayed

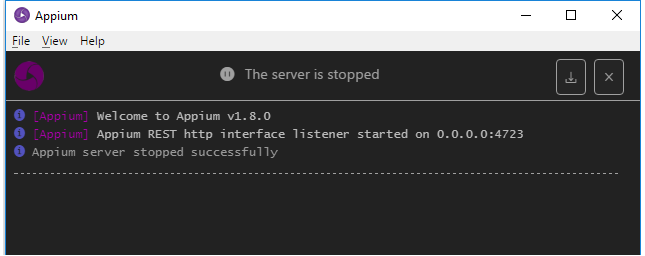


**6.**To verify that the installation and setup is successful, click on Start Server button. This would start the Appium server and you would see **‘The server is running’**message in Appium window



This shows that the Appium Desktop has been installed successfully.

**7.**Click on Stop Server button to stop the Appium server



After Appium server is stopped, you can close the Appium Desktop window. With this we complete the Appium Desktop download and installation process. Try it out and let us know if you face any issues while setting up Appium Desktop. You can also share your feedback with us with using the comments section. Your feedback will go a long way in helping us improve our articles and make it more useful to all our readers.

Top of Form



Bottom of Form

# First Appium Test Script to Launch App on Real Device

This article covers the below topics, which you can follow to run your test script

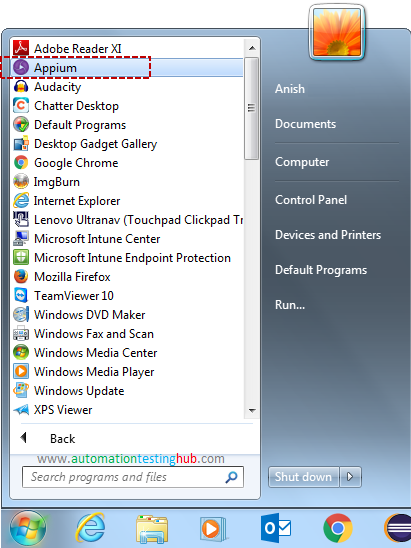
* Start Appium Server
* Get details of the mobile device
* Get details of the app which is to be launched
* Write Appium test script
* Run your test script and check that it launches the app

Let us now work with each of these topics one by one.

## Start Appium Server

We are going to use [Appium Desktop App](http://www.automationtestinghub.com/appium-desktop/" \t "_blank) to start the Appium server. Follow the steps given below to do this –

**1.**Go to **Start -> All Programs** and look for **Appium**



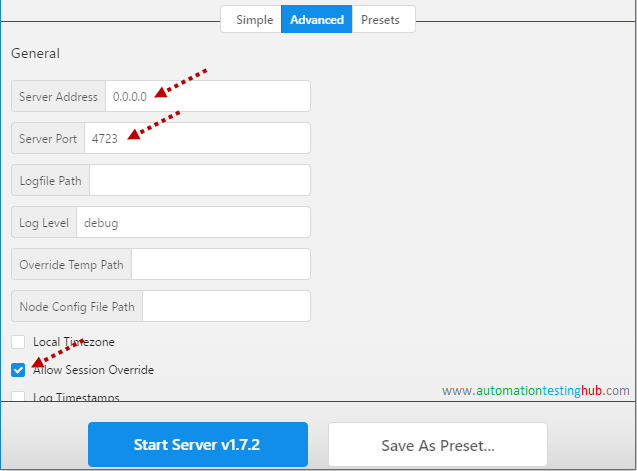
**2.**Click on Appium to open Appium Desktop (it might take 20-30 seconds for Appium Desktop to open)

**3.**Once Appium Desktop opens, navigate to the **Advanced** tab as shown below



**4.**Enter the following values in the fields as shown in the below image –

* **Server Address** – 0.0.0.0
* **Server Port** – 4723
* **Allow Session Override** – tick the checkbox

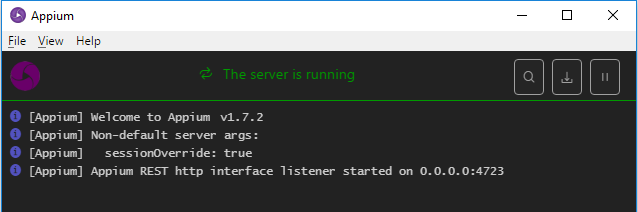


**Note:** Why did you tick “Allow Session Override” checkbox? When you run an Appium test script, it creates a new session on the Appium server. If this session doesn’t close properly, then your script would fail when you re-run it. This is because the previous session still exists and thus, Appium server is not able to create a new session.

And when you tick the checkbox, you provide Appium the capability to override the existing session when you re-run scripts. This would ensure that your scripts don’t fail due to session related issues.

**5.**Leave all other fields as it is, and then click on **Start Server v1.7.2** button

**6.**Appium server would start and you would see the screen as shown below



Let us now move over to the next section.

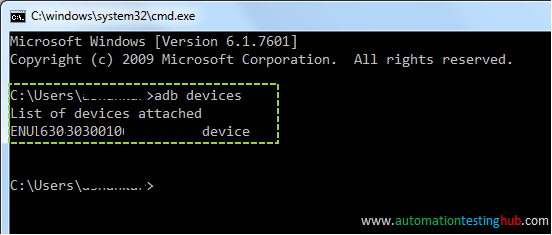
## Get your Mobile Phone’s Device ID

Appium identifies your mobile phone using its Device Id (also called as its UDID). For example, consider a scenario where you have multiple mobile phones connected to your machine using USB cables. Now when you run your Appium test script, Appium would detect that you have multiple devices connected. So it would need the Device ID, so that it can connect to the correct device and run the scripts on that device. Let us see how you can find the Device ID (or UDID) of your mobile phone –

**1.**Connect your mobile device to your computer using USB cable. (If your mobile device shows some popup after connecting, then accept that popup)

**2.**Now open command prompt and run this command: **adb devices**

**3.**Once you run this command, you would see the details of the device as shown in the below image (if you see some other response such as “daemon not running. daemon started successfully”, then run the command again)

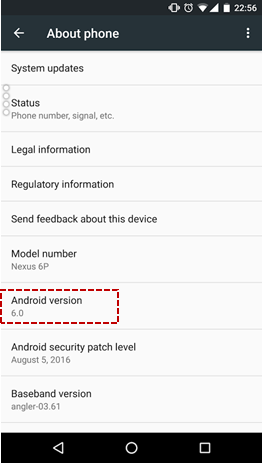


**4.**From the above screenshot the Device ID is – **ENUL6303030010**. Note down the device ID that you get for your phone, because you would need to provide that in your Appium test script.

**5.**Keep your mobile device connected to the USB cable. It needs to remain this way when you run your script.

## Get your Mobile Phone’s Android version

You would also need to provide your mobile phone’s Android version number in the Appium script. To find this, open your mobile phone and go to **Setting > About Phone**. In About Phone screen, you can see the Android version as shown below –



Please note that the exact process to check the Android version might differ from phone to phone. If you have difficulties finding it on your device, then please check for the steps on internet.

## Find out appPackage and appActivity names of the mobile app

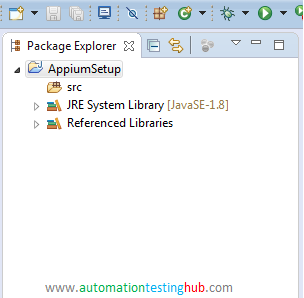
In our previous article on [finding appPackage and apActivity name](http://www.automationtestinghub.com/apppackage-and-appactivity-name/), we covered in detail the process of identifying this information. We had also mentioned there, that we will use Play Store app as an example in this article. So for Google Play Store app, we have

* appPackage name as – **com.android.vending**
* and appActivity name as – **com.google.android.finsky.activities.MainActivity**

## Write Appium Test Script to launch Play Store app

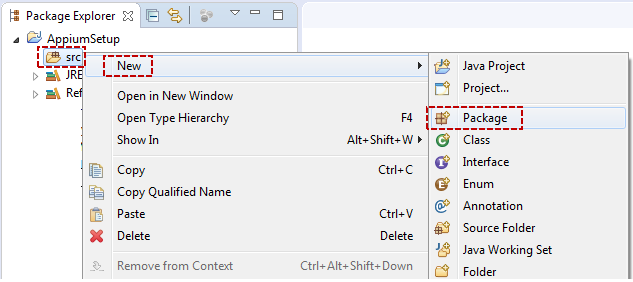
Now comes the important part where you write the Appium test script. Do this by following the steps given below –

**1.**Open the Appium project that we had created in this article – [Create new project in Eclipse](http://www.automationtestinghub.com/appium-project-in-eclipse/). It should look something like this –

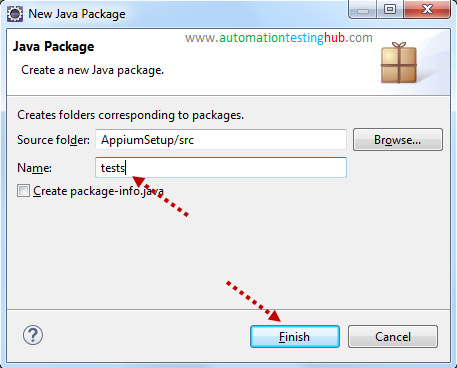


We will first create a new package in this project. Then we will then create a Java class inside the package, and finally we will write the Appium test script in the Java class. Let’s see how this is done.

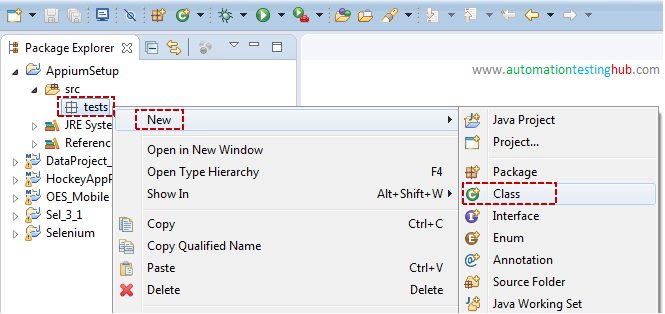
**2.**In the Appium project in Eclipse, right click on **src** folder. Then select **New > Package**



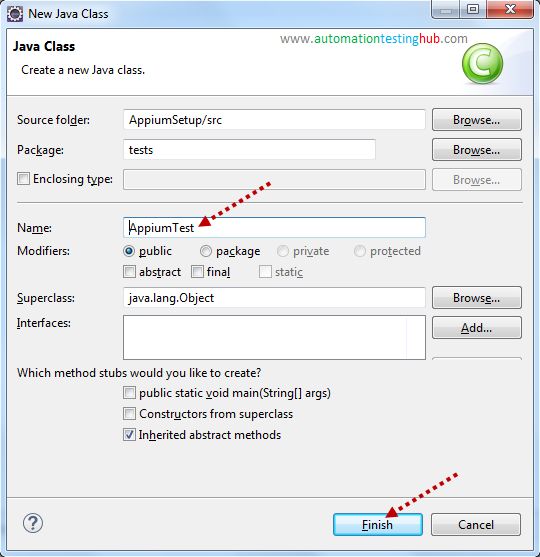
**3.**In the popup window, enter package name as **tests** and then click on **Finish** button



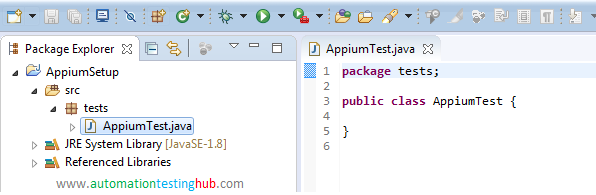
**4.**You can see that a new package called tests is created under src folder. Right click on this package and select **New > Class** option



**5.**In the popup window, enter the class name as **AppiumTest** and then click on **Finish**button



**6.**The new class would be created as shown below. This is the place where you will add your code.

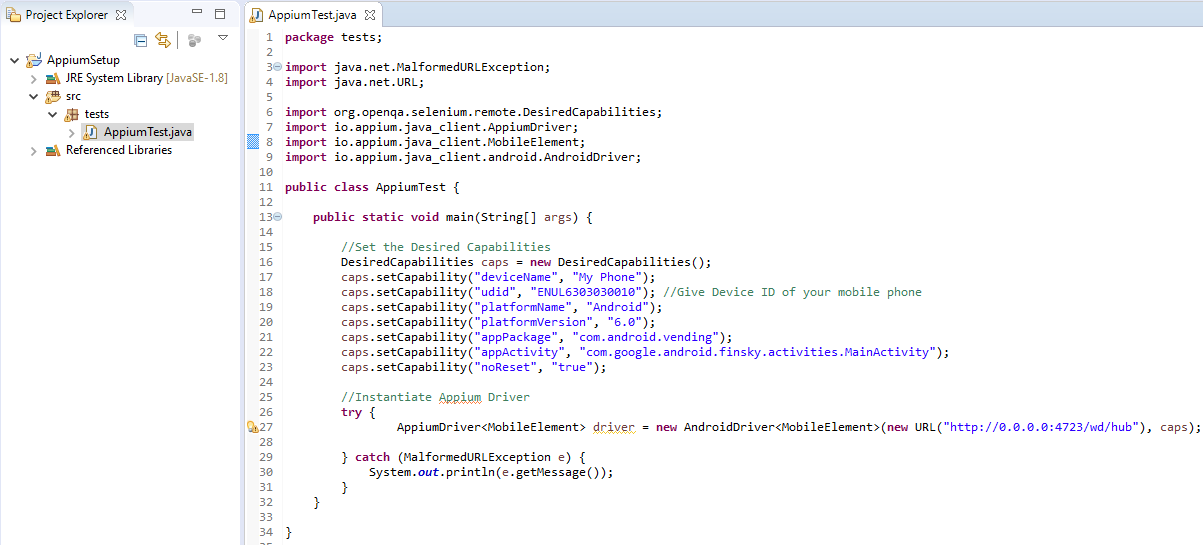


**7.**In this article, we will just provide you with the complete code that you would need to launch Play Store app. Don’t worry if you are not able to understand some of the details in the code. We will work on a separate article, where we will explain the code in detail.

**8.**For now, delete all the content in **AppiumTest.java** class and replace it with the code given below. **Also, please make sure that you replace the device id (in line number 18) given in this code with the device id of your phone.**



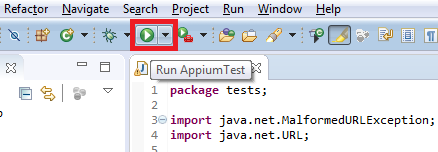
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | package tests;    import java.net.MalformedURLException;  import java.net.URL;    import org.openqa.selenium.remote.DesiredCapabilities;  import io.appium.java\_client.AppiumDriver;  import io.appium.java\_client.MobileElement;  import io.appium.java\_client.android.AndroidDriver;    public class AppiumTest {    public static void main(String[] args) {    //Set the Desired Capabilities  DesiredCapabilities caps = new DesiredCapabilities();  caps.setCapability("deviceName", "My Phone");  caps.setCapability("udid", "ENUL6303030010"); //Give Device ID of your mobile phone  caps.setCapability("platformName", "Android");  caps.setCapability("platformVersion", "6.0");  caps.setCapability("appPackage", "com.android.vending");  caps.setCapability("appActivity", "com.google.android.finsky.activities.MainActivity");  caps.setCapability("noReset", "true");    //Instantiate Appium Driver  try {  AppiumDriver<MobileElement> driver = new AndroidDriver<MobileElement>(new URL("<http://0.0.0.0:4723/wd/hub>"), caps);    } catch (MalformedURLException e) {  System.out.println(e.getMessage());  }  }    } |

**9.**Your code in Eclipse should look like this. Press **Ctrl + S** keys to save this Appium test script.  


**Note:**When you check the code, you would observe that all the values (such as udid, appPackage, appActivity etc) are the ones that you had fetched in the previous sections of this article.

## Run Appium Test Script

You have written your Appium test script. Its now time to run it. Please make sure that your mobile device is connected to your computer with USB and its unlocked. To run the script, click on the run icon from the toolbar.



Your script would start running and you would see that Play Store app would get launched on your mobile device. This entire process might take 10-15 seconds because Appium has to connect to your mobile device first. A good approach would be to run your script and then monitor the Appium Desktop screen. You would notice that Appium Desktop screen should start showing some logs. This way you will know that your Appium test script is running.

Congratulations!!! With this, you have completed the first step of Appium mobile automation – which is to launch your mobile app with Appium code. If you have faced any issue while running the script, or if your app does not launch, then please let us know using the comments section.

* [Skip to content](http://www.automationtestinghub.com/apppackage-and-appactivity-name/#genesis-content)
* [Skip to primary sidebar](http://www.automationtestinghub.com/apppackage-and-appactivity-name/#genesis-sidebar-primary)

[**AutomationTestingHub**](http://www.automationtestinghub.com/)

Selenium, Appium, Cucumber and more !!!

Top of Form



Bottom of Form

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

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 name of your app under test.** You can use any of these methods to find out the package and activity name 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 which is provided by its developers.** It’s actually a 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 it’s name as ‘com.facebook.katana’ in Appium.

#### appActivity:

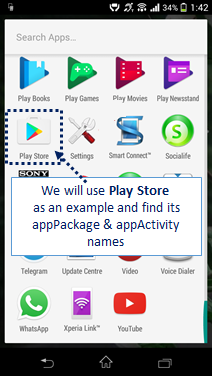
Again, **speaking in very basic terms appActivity refers to the different functionalities that are provided by the app.**  
**For example**, WhatsApp provides multiple functionalities such as conversations, profile information, setting profile photo, setting status, notifications and a lot of other things. All these functionalities are represented by different 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)

Without waiting any further, let us start with the different methods with which you can identify the appPackage and appActivity name of your app.

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

With our [Appium Tutorial](http://www.automationtestinghub.com/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

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/" \t "_blank) 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:

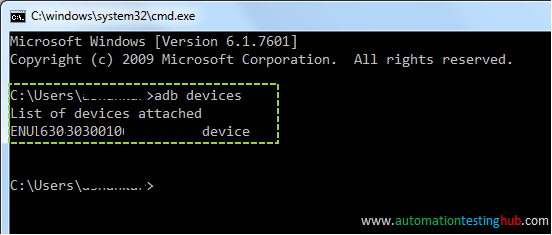
* You should have [downloaded and installed Android SDK](http://www.automationtestinghub.com/download-and-install-android/)
* You should have checked if you need to [install any additional SDK packages](http://www.automationtestinghub.com/install-additional-android-sdk-tools/)
* You should have [set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

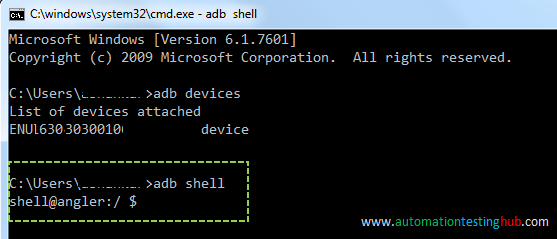
**Step 1:**Unlock your mobile device and connect it to your computer using USB cable

**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.

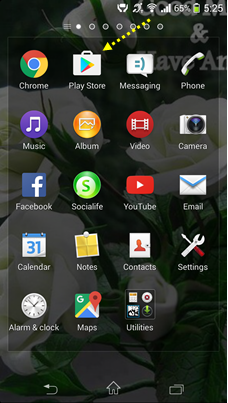
**Step 3:**Once you run **‘adb devices’** command, you should see that it displays the list of attached devices as shown in the below image (the actual device name that you see would be different based on what mobile phone you use) –



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



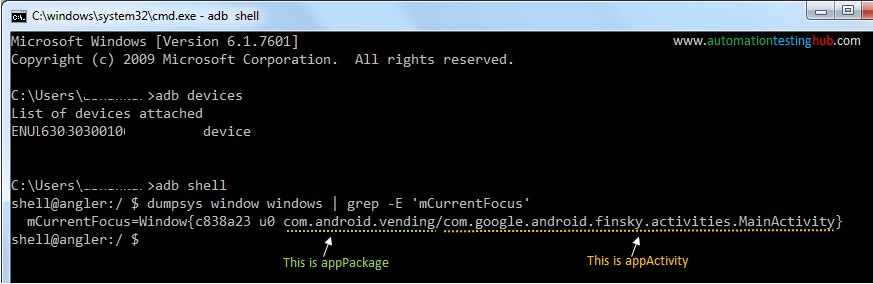
**Step 5: Now in your mobile phone, open the app for which you want to find the appPackage and appActivity.** Since we are doing this for Play Store, hence we will open **“Play Store”** on our mobile phone.



**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’**

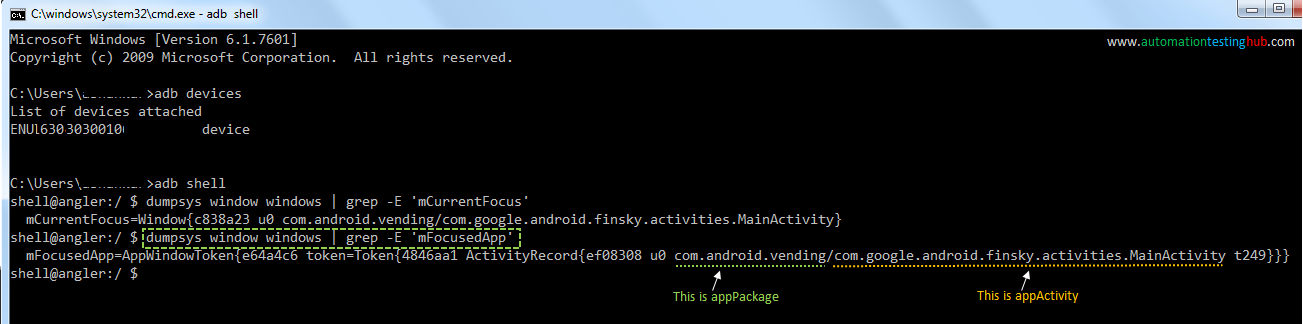
**Step 7:**The above command would display the details of the app which is currently in focus. From that, you can figure out the appPackage and appActivity name as per the below image –



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.

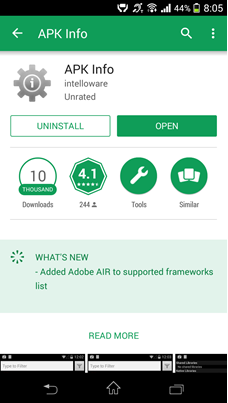
**For example,** consider some app which shows **com.myapp/.mainActivity** in command prompt when you run the above commands. In this case, you will notice that the appActivity starts with a dot (which is a relative name). So you would need to add com… at the beginning. After adding the complete appActivity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

## Method 2: Using APK Info app

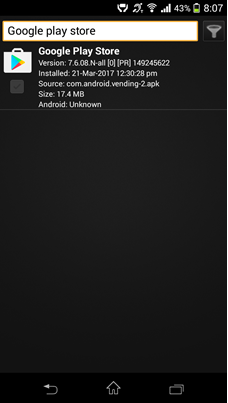
APK Info is an app which you can download from Play Store, and **it will provide the appPackage and appActivity name of any app which is installed on your mobile device.**

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

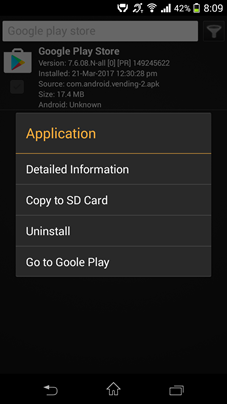
**Step 1:**Download “APK Info” app from Google Play Store on your android mobile.



**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

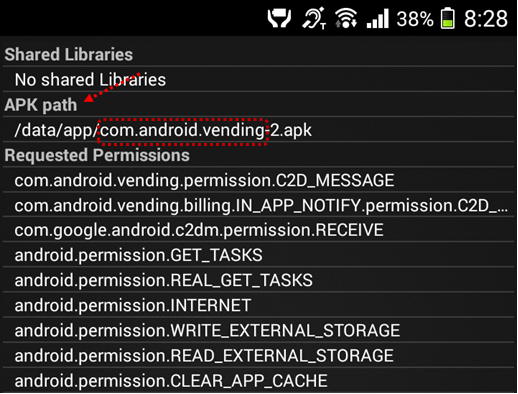


**Step 3:**Long press on the “Google Play Store” application icon inside the APK Info app till it displays the list of options as shown below –



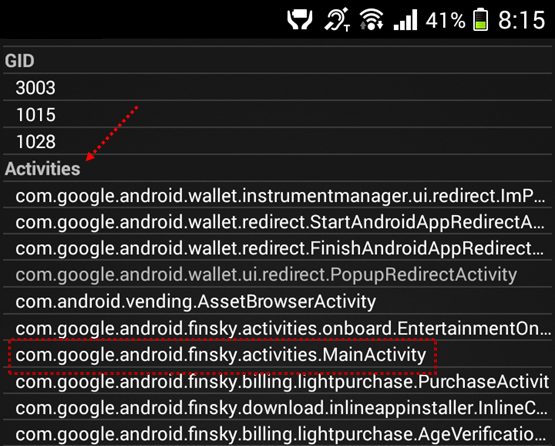
**Step 4:**Click the option “Detailed Information” option. It would show the detailed log for the app.

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.

We are continuously adding more articles to our tutorial series. You can check it out here – [Appium Tutorials](http://www.automationtestinghub.com/appium-tutorial/" \t "_blank)

Filed Under: [Appium](http://www.automationtestinghub.com/category/appium/)