Programming: Language: Python
Framework: PyTest and Appium

Introduction

For detailed information on PyTest, please refer to its documentation. In this guide, we will use Appium to automate Android devices. We will first cover the basics of Appium and then learn how to integrate it with LambdaTest. Appium is a tool that interacts with devices using UiAutomator, which helps to interact with the UI of devices.

Requirements

- Node.js: >=18 (latest version recommended)
- NPM: >=8 (for installing Appium)
- Java: >=7 (not in official documentation but recommended)
- Operating System: Windows, Linux, or macOS (Ubuntu is used in this tutorial)

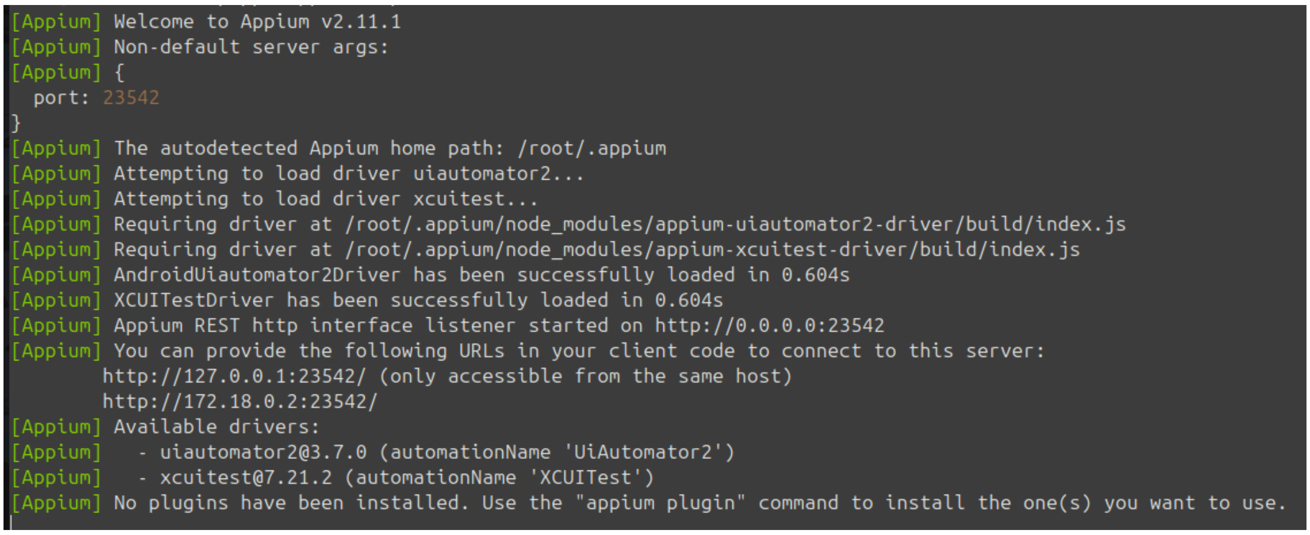
Installation

Assuming that you have already installed the required dependencies, you can follow these steps to set up Appium:

1. Install Appium globally using npm:
   -> npm i --location=global appium

2. To run Appium, execute the following command:
   -> appium

You should see an output similar to this:



```
[Appium] Welcome to Appium v2.11.1
[Appium] Non-default server args:
[Appium] {
  port: 23542
}
[Appium] The autodetected Appium home path: /root/.appium
[Appium] Attempting to load driver uiautomator2...
[Appium] Attempting to load driver xcuitest...
[Appium] Requiring driver at /root/.appium/node_modules/appium-uiautomator2-driver/build/index.js
[Appium] Requiring driver at /root/.appium/node_modules/appium-xcuitest-driver/build/index.js
[Appium] AndroidUiautomator2Driver has been successfully loaded in 0.604s
[Appium] XCUITestDriver has been successfully loaded in 0.604s
[Appium] Appium REST http interface listener started on http://0.0.0.0:23542
[Appium] You can provide the following URLs in your client code to connect to this server:
        http://127.0.0.1:23542/ (only accessible from the same host)
        http://172.18.0.2:23542/
[Appium] Available drivers:
[Appium]   - uiautomator2@3.7.0 (automationName 'UiAutomator2')
[Appium]   - xcuitest@7.21.2 (automationName 'XCUITest')
[Appium] No plugins have been installed. Use the "appium plugin" command to install the one(s) you want to use.
```

3. Install Appium Inspector to get the path or selector for UI elements. Visit the following URL: https://github.com/appium/appium-inspector/releases
   Appium Inspector Releases and download the version supported by your OS. If you are using Ubuntu, download the x86_64.AppImage.

Note: *If you are using Ubuntu 24 with a basic (Minimal Installation), install the following package to open the app with a click:*

```
-> sudo apt install libfuse2
```

4. You will need the Android SDK to run Appium Inspector. To simplify this process, install Android Studio, which includes everything you need.If you don't have adb, you can install it with:

   ```
   -> sudo apt install adb
   ```

   ADB helps to list all connected devices to the system. ADB is a command-line tool that facilitates communication with devices.

Getting UI Elements from Appium Inspector

You can use either an emulator or real devices. In this guide, we will use real devices, but the steps will be similar for emulators.

1. After connecting the device, run the following command to get a list of all connected devices along with their UUIDs:

   ```
   -> adb devices
   ```
2. Create device capabilities, which will contain some basic components like platformName, deviceName, platformVersion, automationName, etc. You can learn more about it here.

For Emulator:

```
{
   "platformName": "Android",
   "platformVersion": "11.0",
   "deviceName": "Pixel 7 API 30",
   "automationName": "UiAutomator2",
   "autoLaunch": false
}
```

For Real Device:

```
{
   "platformName": "Android",
   "appium:platformVersion": "11",
   "appium:deviceName": "POCO M2 Reloaded",
   "appium:automationName": "UiAutomator2",
   "appium:autoLaunch": false
}
```

Note: *If you change the Appium port, update it in the port section. Similarly, update the path section if needed; generally, the path will be /wd/hub.*

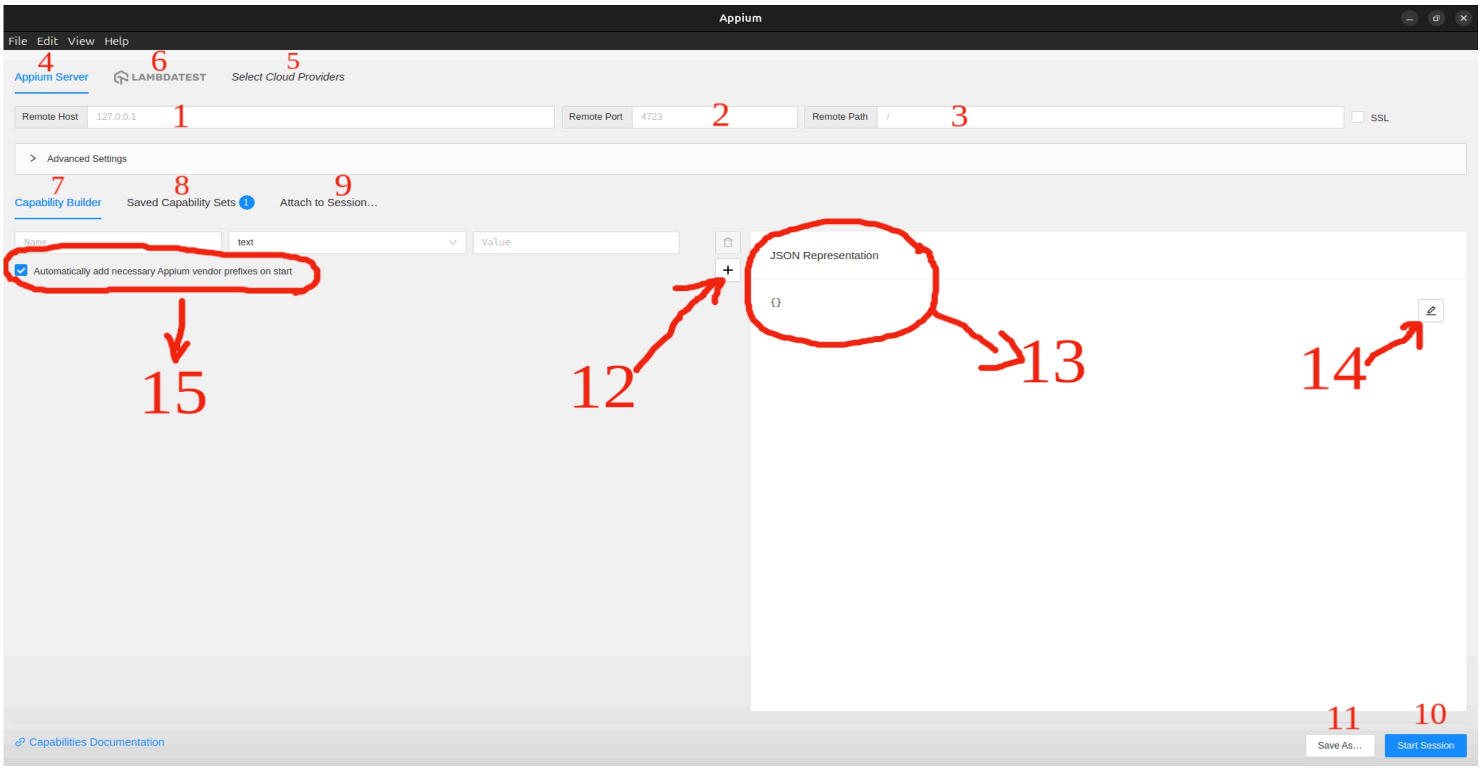
4. Click on 'Start Session'.

5 . Follow these steps to record and interact with UI elements:

- Click on the start recording icon (if you are running Appium on its default settings).

- In the 'Source' section, you will see 'App Source' and 'Selected Elements'.
- In 'App Source', you will get the UI elements source.
- In 'Selected Elements', you will find tools to record and interact with elements. You will also have properties like xpath, attribute, and index, etc. These properties will be used to connect to the device while running automation.

Appium Inspector:



In the above image, there are numbers related to each of the elements or parts of the Appium Inspector. Below are descriptions of these elements, corresponding to their numbers.

1. Remote Host: If you are just running appium, you can leave this blank (the default value will be automatically taken). If you want to use a specific address, write it here. You can change the address using the command --address address_name. The default address is 127.0.0.1.

2. Remote Port: Leave this blank for the default value. Appium uses port 4723 by default. If you want to use another port, you need to add --port port_no at the end of the command and write the port number here.

3. Remote Path: Again, leave this blank for the default value. It uses / as the path. To change the path, use --base-path=path_name. Generally, the path used is /wd/hub.

4. Appium Server: Select this option if you are using the Appium server, as we are doing.

5. Select Cloud Provider: Here, you can select a cloud provider who provides remote devices. Some options include LambdaTest, BrowserStack, and Sauce Labs, which provide cloud devices for automation. These services help in the production environment of automation. We will use LambdaTest in this tutorial. If you select it, you are good to go.

6. LambdaTest: We selected LambdaTest in the previous step, hence its components appear here.

7. Capability Builder: Here, you will add information about the device you select for further procedures (this is explained in the 'Getting UI Elements from Appium Inspector' section).

8. Saved Capability Sets: In the 13th section, you write your device capabilities. You can do so by clicking the button in the 14th section and then clicking the save button in the 11th section. After that, you will see your saved capability here.

9. Attach to Sessions: If you are using another device on the same Appium server, you can directly select it from here. It will be automatically listed.

10. Add Capabilities: (12th) You can add capabilities by clicking here and selecting the type of object you want to enter, e.g., string, boolean, etc.

11. Additional Files: (13th) This boolean value will add any other files that will be needed during the Appium run.

Automation Script

```
from appium.options.common.base import AppiumOptions
from appium import webdriver

options = AppiumOptions()
options.load_capabilities({
    "platformName": "Android",
    "appium:deviceName": "",
    "appium:platformVersion": "11",
    .
    .
    ...
})
driver = webdriver.Remote('<appium url>', options=options)
```

In the above code, we create a driver that will be used further. First, we create options with capabilities, then we add them to the WebDriver.

NOTE: *In place of <appium url>, you can use http://127.0.0.1:4723/ if you are using the default options for Appium.*

Now, we can write a simple automation script to log in to an application:

```
from appium.webdriver.common.appiumby import AppiumBy

 def test_login():
    # Click on the button
    driver.find_element(by=AppiumBy.ID, value="android:id/button1").click()
    # Enter email
    driver.find_element(by=AppiumBy.ANDROID_UIAUTOMATOR, value='new
 UiSelector().text("Enter Email Address")').send_keys('email@test.com')
    # Enter password
    driver.find_element(by=AppiumBy.ANDROID_UIAUTOMATOR, value='new
 UiSelector().text("Password")').send_keys('***********')
    # Click on login button
    driver.find_element(by=AppiumBy.ID, value="login").click()
```

Before running the script, make sure that Appium is running and you have the correct Appium URL.

The driver.find_element method will find/select elements on the screen using the given selector. For example, by=AppiumBy.ID uses the ID selector. Appium will search for elements with the ID android:id/button1 and then click on it. Similarly, it will enter the email and password, and then click on the login button.

How to Use LambdaTest for Automation

1. **Integrating LambdaTest with Appium**
   To integrate LambdaTest with Appium, you just need to add your LambdaTest Username and Access Key. The rest of the steps are similar to those described above.

Note: *For LambdaTest capabilities, you can use the LambdaTest Capabilities Generator.*

2. **Automation Test Script:**
   Change the Appium URL to:
   " https://<username>:<accessKey>@mobile-hub.lambdatest.com/wd/hub "
   Replace <username> and <accessKey> with your actual LambdaTest username and access key. After that, everything else remains the same, and you will be good to go.

3. **How to Get Username and Access Key**
   a) Login to LambdaTest and go to the dashboard.

b) **Method 1:**
In the dashboard, at the bottom right side, you will see a key icon. Click on it to get your credentials.

c) **Method 2:**
-Go to Settings > Account Settings > Password & Security. Here you will find your credentials.