What is an Appium Client?

Appium Client is a software library or a set of tools that allows developers to write and execute test scripts for mobile applications using the Appium framework. It provides a way to interact with and control mobile applications (both native and hybrid) on different platforms like iOS and Android. The Appium Client communicates with the Appium server to send commands and receive responses, enabling automated testing of mobile apps across various devices and operating systems.

Automation commands are available that is up to the particular driver and plugins that you are using in any given session. A standard set of commands would include:

* Find Element
* Click Element
* Get Page Source
* Take Screenshot

If you look at these commands in the WebDriver specification, you'll notice that they are not defined in terms of any particular programming language. They are not Java commands, or JavaScript commands, or Python commands. Instead, they form part of an HTTP API which can be accessed from within *any* programming language.

**Example**: The Find Element command corresponds to an HTTP POST request sent to the HTTP endpoint /session/:sessionid/element (where in this case, :sessionid is a placeholder for the unique session ID generated by the server in a previous call to Create Session).

With the recommended different script despite being in different languages, does the same thing under the hood:

1. Call Find Element with a using parameter of xpath and a value parameter expressing the XPath query used to find an element. (If you're confused about these terms, you might find an introduction to Appium or Selenium useful)
2. Call Click Element with the ID of the element found in the previous call.
3. Call Get Element Text with the ID of the same element, and print it to the console.
4. Call Get Page Source to retrieve the page/app source and print it to the console.

Note: WebDriverIO primarily operates as a JavaScript-based framework, and it uses Node.js for executing test scripts. Since its ecosystem is largely centred around JavaScript, the most common language for writing test scripts in WebDriverIO is JavaScript and it also support other languages like (Java, Python, C#, PHP, Ruby and Golang).

Official Clients

1. These clients are currently maintained by the Appium team:
2. Appium Java Client (Client) Java (Language Use).
3. Appium Python Client (Client) Python (Language Use).
4. Appium Ruby Core Client (Recommended) Ruby (Client) Ruby (Language).
5. Appium .NET Client (Client) C# (Language Used).

Other Clients

These clients are not maintained by the Appium team and can be used with other languages:

1. WebdriverIO (Node.js).
2. Nightwatch.js (Node.js).
3. RobotFrameWork (DSL).

Types of Protocol Use by WebDriverIO?

1. **JSON Wire Protocol**:
   * This was the original protocol used by WebDriver to enable communication between clients and browsers.
   * It relies on HTTP requests, where clients send commands to the WebDriver server and receive responses.
   * It has been largely considered outdated and has been superseded by the W3C WebDriver Protocol.
   * While some existing tools may still reference or utilize JSON Wire Protocol, it is not recommended for new projects.
2. **W3C WebDriver Protocol**:
   * This is the current standard for WebDriver, which was developed by the W3C to provide a unified and standardized API for browser automation.
   * It implements a more structured set of capabilities and interactions compared to the JSON Wire Protocol.
   * The W3C WebDriver Protocol has become the de facto standard for browser automation across major browsers and is supported by WebDriverIO.

WebDriver Bidi Protocol

* **WebDriver Bidi (Bi-directional WebDriver Protocol)**:
  + This is a newer protocol designed to replace the traditional WebDriver protocol.
  + It introduces bi-directional communication, allowing clients to send commands and receive events or notifications seamlessly.
  + The bi-directionality opens up new possibilities for introspection and real-time control over browser automation tasks, such as tracking state changes or receiving updates on navigation events.
  + As of now, WebDriver Bidi is still under development, and while some browser vendors are working towards implementing it, you may not find full support in all tools immediately.

Note: The server (consisting of Appium itself along with any drivers or plugins you are using for automation) is connected to the devices under test, and is actually responsible for making automation happen on those devices. The client (driven by *you*, the Appium test author) is responsible for sending commands to the server over the network, and receiving responses from the server as a result. These responses can be used to tell whether automation commands are successful, or might contain information that you queried about the state of the application.

Note: **Appium**: As mentioned earlier, WebDriverIO uses the Appium driver for mobile testing.

Note: **WebdriverIO** essentially refers to a tool that uses the **WebDriver** protocol to perform **Input/Output** operations with a web browser for automated testing.

Note: Appium commands in five different programming languages, using the recommended Appium client binding for each language.

WebDriverIO

In a broader sense, you can say that **WebDriverIO is an Appium client**, as it uses Appium's capabilities to automate testing of mobile applications. WebDriverIO acts as a wrapper around the Appium driver, providing a simpler and more intuitive API for mobile testing. It leverages the underlying capabilities of Appium to control and automate mobile applications.

WebDriverIO can use in Appium Client

1. **Uses Appium driver**: Under the hood, WebDriverIO uses the Appium driver to interact with mobile applications. This driver is responsible for controlling the application and executing commands.
2. **Supports Appium features**: WebDriverIO provides access to various Appium features, such as gestures, file uploads, and text editing, making it an integral part of the Appium ecosystem.
3. **Automates mobile testing**: The primary purpose of WebDriverIO is to automate testing of mobile applications, which it achieves through its interaction with the Appium driver.

**Workflow Example:**

1. **You write test scripts** in WebdriverIO using JavaScript.
2. The scripts define actions like tapping on buttons, entering text, or navigating between screens in a mobile app.
3. WebdriverIO sends these commands to the Appium server, specifying the device and the actions to perform.
4. Appium communicates with the mobile device (or simulator/emulator) to execute these actions.
5. The results are sent back to WebdriverIO, and your test can assert behaviors (e.g., checking if an element is visible, validating UI states).