USING APPIUM FOR MOBILE TEST AUTOMATION
Mobile phones have made everyone’s life easier, with Internet access and life-enhancing apps always within reach. But with this wider use, the pressure to quickly launch has skyrocketed, and so the need to adequately test these apps before they reach market has never been more important.

Keeping up with QA and testing needs for the mobile platform obviously cannot be accomplished through manual testing alone. Additionally, there’s a platform proliferation problem — who has all those platforms in-house?

To efficiently address mobile application testing needs, it is critical to find a standardized, widely accepted automated testing platform.

**WHY APPIUM?**

Appium is an open-source test framework that covers automation testing across all three types of mobile applications: native, web and hybrid. It is also cross-platform and can reuse code between iOS, Android, and Windows test suites with the same API.

As a popular open-source tool, there is ample support from an active developer community that is continually refining it. We like to use Appium because it satisfies the requirements that we test engineers demand for a mobile test platform:

- Commonly used programming APIs can be integrated.
- Can run an app through Appium server without the need to modify the app.
- No source code required (unlike Robotium).
- Continuous-integration compatibility with Jenkins.
- Runs on Selenium Grid.
- Works with real devices and simulators, both locally and via cloud services.
- Doesn’t require an Android Package Kit (APK).
During the last 10 years in providing mobile testing solutions, we’ve tried many different mobile testing tools and technologies. Along the way, we’ve accumulated deep expertise in using Appium as an all-purpose mobile test automation solution. Typically, we use it for the following types of tasks:

- Testing against web applications.
- Native app testing.
- Local and remote testing.

Over the years, we’ve found that while Appium may have great community support and substantial documentation, filtering through the configuration and operational details is more difficult for some situations than others.

Some of the more common Appium issues we have experienced include:

- **Problem 1:** iOS testing not supported on Windows.
- **Problem 2:** Xcode 8 no longer supports UIAutomation.
- **Problem 3:** Complicated management of devices and tests.
- **Problem 4:** Unable to run simultaneous iOS simulators.

This white paper steps through each of these problems, why they are important, and how we approached solving them.
Though Appium supports both Android and iOS, to perform iOS application testing also requires Xcode in macOS. This option is simply not available in Windows.

If the engineers were previously using Windows to do the coding and testing, or if they want to support Apple’s iOS applications, they have to switch to a Mac to debug and run the tests (Mac emulation on Windows, due to inconsistencies that can occur between Apple hardware and software operating on non-native platforms, is not recommended).

The standard solution is to simply use an Apple computer with macOS by:

1) installing the latest Xcode,
2) installing the latest Android SDK, and
3) setting up Appium for both iOS and Android, step by step.

Upon successful Appium setup, a final confirmation message will be generated (see Figure 1).

![Figure 1: Appium Configuration Confirmation Screen](image-url)
And with Appium GUI mode, you can quickly switch between iOS and Android to edit each of the settings (see Figure 2).

The decision to run Appium tests under macOS provides the flexibility to drive iOS devices or simulators, as well as the ability to execute tests with the Android operating system.

The ability to move between macOS and Appium allows simplified coding, debugging and script execution all on the same machine.
This is a significant issue that may be encountered when a mobile automation tester moves from previous versions of iOS (9.2 and below) to iOS 10. The issue, clearly defined within Appium GitHub notes, states, “For iOS automation, Appium relies on system frameworks provided by Apple.”

For iOS 9.2 and below, Apple's only automation technology was called UIAutomation, and it ran in the context of a process called ‘Instruments’. As of iOS 10, Apple has completely removed the UIAutomation instrument, thus making it impossible for Appium to work the way it used to. Fortunately, Apple introduced a new automation technology, called XCUITest, beginning with iOS 9.3. For iOS 10 and up, this will be the only supported automation framework from Apple.” (see Figure 3)

As Xcode 8 no longer supports UIAutomation, tests created previously will likely not work correctly. Specifically, the UIAutomation issue occurs when you update the Xcode to version 8 or above when you have iOS tests created previously on Xcode 7 with the UIAutomation framework. In those cases, you have to either keep using an old version of Xcode and Appium or migrate the tests to comply with XCUITest.

<table>
<thead>
<tr>
<th>Xcode version</th>
<th>Build</th>
<th>Release date</th>
<th>min OS X to run[67]</th>
<th>OS X SDK(s)[68]</th>
<th>iOS SDK included[69]</th>
<th>Automation support</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3</td>
<td>7D175</td>
<td>March 21, 2016</td>
<td>10.11</td>
<td>10.11.4 (15E60)</td>
<td>iOS 9.3 (13E230)</td>
<td>UIAutomation and XCUITest</td>
</tr>
<tr>
<td>7.3.1</td>
<td>7D1012</td>
<td>May 3, 2016</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7D1014</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>8A218a</td>
<td>September 13, 2016</td>
<td></td>
<td>10.12 (16A300)</td>
<td>iOS 10 (14A345)</td>
<td></td>
</tr>
<tr>
<td>8.1</td>
<td>8B62</td>
<td>October 27, 2016</td>
<td>10.11.5</td>
<td>10.12.1 (16B2649)</td>
<td>iOS 10.1 (14B72)</td>
<td>XCUITest</td>
</tr>
<tr>
<td>8.2</td>
<td>8C38</td>
<td>December 12, 2016</td>
<td>10.12 (16C58)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.2.1</td>
<td>8C1002</td>
<td>December 19, 2016</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.3</td>
<td>8E162</td>
<td>March 27, 2017</td>
<td>10.12</td>
<td>10.12.4 (16E185)</td>
<td>iOS 10.3 (14E269)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Xcode Version Compatibility Matrix

---

As XCUItest is a new technology, a number of things in previously developed automation scripts may be affected. And while Appium Development has taken care to limit the impact of the change for already developed scripts, there still may be issues. For example, the application hierarchy reported to Appium is not necessarily identical within XCUItest versus UIAutomation. Class names may be different as well.

There are some approaches to help alleviate test automation issues when moving to iOS 10.

- Update Appium to 1.6.3 or above to support XCUItest, though there is no GUI mode yet (Command line only), and then install all the XCUItest components required.
- Map the classes and locators as much as possible following Appium’s instructions.\(^3\)
- Replace XPath location, as it is not natively supported by Apple’s XCTest framework.

When moving forward with iOS 10, testers have little choice but to adhere to the new testing guidelines in support of Xcode’s new XCUItest technology. It is not highly complex, but care must be taken to ensure proper operation of new scripts, as well as continuity for previously developed scripts.

---

\(^3\) “Migrating to XCUItest — appium.”
Appium’s ability to deal with a wide variety of devices, real and virtual, is a key benefit of the test platform. But this variability comes at a price, the ongoing management challenge of device configurations as well as test case diversity and associated management.

For mobile test environments, you will likely need to manage and maintain multiple real devices or simulators. Furthermore, different test scripts may be tailored to different devices, adding a layer of complexity. Configuration and management of these mobile environments can be time consuming, including organizing a wide range of preconditions that must be defined (connections, desired capabilities, browsers, Appium server ports, etc.) before the tests are executed.

There are ways to simplify the mobile test configuration management (MTCM) challenge:

- **Use WiFi connections** to avoid having to physically connect and disconnect devices.
- **Use variables**. Never hard code any device-specific values in the automation scripts.
- **Use testng.xml to pass parameters**. Examples include what tests to run on which device, configuration information, device name, UDID (see Figure 4).
- **Keep devices connected** to assist in ensuring mobile configurations will remain unchanged (e.g., assigned IP address).
- **Maintain script organization**. Create multiple XML files as needed to maintain proper organization.
- **Configure script initialization and termination**. Code the setup and teardown process for each test within the script (e.g., kill and reset Appium server) for each test to ensure the environment is always valid.
- **Configure continuous integration**. Once the environment and device pool is stabilized, Jenkins can be configured for CI.
Applying the above-noted recommendations will create more efficient test execution workflows by significantly reducing the time required to set up, modify and execute key factors associated with a given mobile test. In addition, a well-organized mobile test configuration management system can ease the burden of project personnel transitions, for both junior level as well as experienced engineers entering a project, significantly reducing the time and effort required between project introduction and test execution.

A key advantage in any test automation is that you can typically set up tests to execute in parallel, where multiple devices can be simultaneously run against a test application server. Running multiple tests from different devices (or simulators) at the same time can dramatically improve the number of overall tests that can be executed within a given time period, especially important in situations where there is a limited test window.

However, unlike Android, running local parallel iOS tests using Appium isn’t currently possible. The issue is not an Appium limitation. Only one session can be active at a time due to Xcode/iOS restrictions. Specifically, according to Apple’s design, only one simulator session can be active at a time on Mac (see Figure 5a).  

---

Figure 5a: Appium Multiple Simulator Configuration: Windows vs. macOS

---

Some ways to alleviate this limitation are as follows:

- **Use real devices rather than simulators.** This can be accomplished by launching multiple Appium server instances, and each server connects a device. Each instance will have its own set of parameters to establish appropriate connections to each device (see Figure 5b). Note that separate temporary folders for each device will still need to be created.

![Figure 5b: Appium Parallel Test Configuration using Real iOS Devices on macOS](image-url)
- **Use Selenium grid** — To drive multiple Selenium client machines in a hub/node configuration (see Figure 5c).

- **Use a third-party cloud service.** BrowserStack, SauceLabs, AWS Device Farm, and similar offerings can support uploading of your Appium tests, iOS and/or Android applications directly to their cloud services. The services can be configured to support as many parallel iOS or Android tests as your account allows.

Parallel testing is an essential tool for the software tester. With the complexities inherent to mobile application software testing, having methods for maximizing test configuration flexibility, including working around limitations such as iOS-related parallel testing, is key to ensuring efficient test execution.

Understanding the large number of viable, available configuration offerings, both local and remote cloud services, will ensure you can select the best configuration for your specific testing needs.
SUMMARY

With many companies now directly taking a mobile-first or mobile-only strategy, no one can afford to release a mobile app that is sub-par. One error can make or break a company and its image. But who has the time to test everything manually, or all the devices needed for testing?

Using Appium can provide many benefits and be a powerful way to conduct test automation for mobile platforms to increase coverage and relieve your manual testing headaches.

However, as we have documented, sometimes it’s not so easy. In this white paper, we’ve shown some of the problems we often encounter, and provided some practical approaches to solving them.

Of course, we have many more techniques and approaches that can certainly alleviate these challenges and support your Appium efforts.

Give us a call. We’re here to help.
XBOSoft is a software testing and quality assurance company that offers services in web, mobile and desktop applications, as well as broad domain experience with extended expertise in healthcare, finance, and software security. XBOSoft’s quality process assessments and test methodologies speed products to market and improve clients’ software quality and performance throughout the software’s life cycle.

With a proven track record working for Fortune 100 companies to small independent developers, XBOSoft today has over 100 employees in San Francisco and Beijing offices serving clients in the US and Europe. As the firm surpasses its tenth year of continuous growth, XBOSoft proudly offers its full range of testing and quality assurance for both on and off-shore services.