New architecture for Appium web and hybrid testing on iOS

Perfecto now supports a new architecture for hybrid and web testing on iOS devices that is fully aligned with local Appium testing.

On this page:

Prerequisites

The new architecture requires:

- iOS 10 and later (XCUITest only).
- XPATH 1.0. Regular expression matching with the **`matches()`** function is not supported.
- Appium client 6.1.0 or above. For more information and for a set of best practices on how to adjust your Appium Java code to work with this new version of the Appium client, see Upgrading to Appium Client 6.1.0.

Limitations

The following limitations apply:

- Cross-domain scripting is not supported.

Enable the new Appium architecture

During the transition phase, you need to select the new architecture explicitly on a per-script basis using the following designated new capabilities. These capabilities control the architecture used for both web and hybrid.
To use the new architecture on all your scripts, you can ask Perfecto Support to update your cloud configuration to use the new architecture by default. If the new architecture is the default, you can omit the new capabilities shown in red below.

New capabilities for web

```
capabilities.setCapability("useAppiumForWeb", true);
capabilities.setCapability("browserName","Safari");
```

New capabilities for hybrid

```
capabilities.setCapability("iOSResign", true);
capabilities.setCapability("useAppiumForHybrid", true);
```

Note: To allow hybrid testing on a cloud device, the app is resigned with a Perfecto development certificate. Apps using special entitlements that require the original author's certificate may not work as intended after the resigning process.

Quantum capabilities

When working with a Quantum project, add the following capabilities.

Quantum | New capabilities for web

```
<parameter name="perfecto.capabilities.useAppiumForWeb" value="true" />
<parameter name="perfecto.capabilities.browserName" value="Safari" />
```

Quantum | New capabilities for hybrid

```
<parameter name="perfecto.capabilities.useAppiumForHybrid" value="true" />
<parameter name="perfecto.capabilities.iOSResign" value="true" />
```

Note: To allow hybrid testing on a cloud device, the app is resigned with a Perfecto development certificate. Apps using special entitlements that require the original author's certificate may not work as intended after the resigning process.

Alternatively, use the following syntax.

Quantum | Alternative syntax

```
<parameter name="perfecto.additional.capabilities" value="{'useAppiumForWeb':true, 'useAppiumForHybrid':
true, 'iOSResign':true}" />
```

Migrate existing Perfecto Hybrid app automation scripts to the new architecture

How you migrate existing Perfecto Hybrid app automation scripts to the new architecture depends on whether you work with single webview or multiple webview apps.

Single webview apps

Note: Due to open tabs in Safari, the driver might identify several webview contexts. However, the implementation automatically uses the old webview names when looking for a single active webview, so the commands **`driver.context("WEBVIEW")`** and **`driver.context("WEBVIEW_1")`** both work as before. There is no need to search for the correct webview.

To migrate scripts for single webview apps:

1. Change the value of the **`automationName`** capability from **`PerfectoMobile`** to its default value, **`Appium`**. Alternatively, because **`Appium`** is the default value, you may remove this capability altogether.
2. Cancel the instrumentation (there is no need to instrument the tested app). Then fix object locators, if needed, because the tree seen by the script is slightly different.
3. Add the following capabilities with a value of **`true`** (as described above):
    - **`useAppiumForHybrid`**
    - **`iOSResign`**

Multiple webview apps

Working with multiple webview apps now requires the use of the Appium capability **`fullContextList`**. Getting a list of all webviews using the **`getContextHandles()`** command returns a list similar to the following:

`["NATIVE_APP", "WEBVIEW_2", "WEBVIEW_3"]`

where each webview is appended with a non-descriptive number and the order is not guaranteed to be the same with every test run. When setting the value of the desired capability **`fullContextList`** to **`true`**, the **`getContexts()`** command returns the contexts as a JSON string. When parsed, each context is an object that includes:

- The ID of the webview
- The URL that thewebview currently displays
- The title of the page

The **`fullContextList`** capability is only supported when testing on iOS devices usingthe new architecture.

To migrate scripts for multiple webview apps:

1. Add the **`fullContextList`** capability to the driver and set it to`true`.
2. Retrieve the contexts via the**`getContextHandles()`** command (in Java, you can then cast each context to **`Map<String,Object>`**).
3. Select the required context byURLortitlekey and execute a context switch, as shown in the following example:

Webview selection

```
Set<Map<String, Object>> contexts = driver().getContextHandles();
Optional<Map<String, Object>>webView;

webView=contexts.stream().filter(c -> !c.get("id").equals("NATIVE_APP"))
.filter(c ->c.get("title").equals(“EXPECTED_WEBVIEW_TITLE”))
.findFirst();

driver().context(webView.get().get("id").toString());
```