# New architecture for Appium testing on Android

Perfecto now supports a new architecture for testing on Android devices that is fully aligned with local Appium testing. This change applies to Native, Web, and Hybrid testing.

**On this page:**

## Benefits of using the new architecture

The new Appium architecture comes with the following benefits:

- Full compatibility with Appium standard
- Bug fixes and enhancements
- Support of Android deep links
  Deep links are URLs that take users directly to specific content in your app. For example:
  **`getDriver().get("theapp://login/darlene/testing123");`**
  When a clicked link or programmatic request invokes a web URI intent, the Android system tries each of the following actions, in sequential order, until the request succeeds:
  1. Open the user's preferred app that can handle the URI, if one is designated.
  2. Open the only available app that can handle the URI.
  3. Allow the user to select an app from a dialog.
- Unique Perfecto features, such as sensor injection, to complement the Appium feature set

## Supported Android devices

The new architecture works with any Android device that supports the UIAutomator2 (v1.25.0) automation framework.

## Prerequisites

The new architecture requires:

- Native: Android 5 and above.
- Web and Hybrid: Android 5 and above with Chrome 66.0.3359.0 and above.
- 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.

## Enable the new Appium architecture

During the transition phase, you need to select the new architecture on a per-script basis using the following designated new capabilities. These capabilities control the architecture used for either native only, native + web, native + hybrid, or native + web + 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 these capabilities.

**New capability for native**

```
capabilities.setCapability("enableAppiumBehavior", true);
```

**New capabilities for web**

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

**New capabilities for hybrid**

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

### Quantum capabilities

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

**Quantum | New capability for native**

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

**Quantum | New capabilities for web**

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

**Quantum | New capability for hybrid**

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

Alternatively, use the following syntax.

**Quantum | Alternative syntax**

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

## Migrate existing Perfecto hybrid scripts to the new architecture

1. If your app uses webviews, set the **`setWebContentsDebuggingEnabled`** property to **`true`** when creating the webview object. For details, see Android remote debugging.
2. 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.
3. 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.
4. Add the following capabilities with a value of **`true`** (as described above):
    - **`useAppiumForHybrid`**
    - **`enableAppiumBehavior`**
5. Review and adjust the switch-to-webview logic in your code: The driver can now return the list with several webview contexts, and the best practice is switching to the last context in this list (provided that the tested webview is currently in the front).