Katalon Studio

Katalon Studio is a comprehensive toolset for web, mobile app, API, and desktop automation testing that helps overcome common challenges in web UI test automation, for example, pop-ups, iFrame, and wait time. It can seamlessly integrate your automated tests into your CI/CD pipeline.

On this page:

1 | Set up Katalon

1. Download your OS-specific Katalon Studio: https://www.katalon.com
2. Install the latest Katalon Studio on your machine.

> If you work with a Mac, add Katalon Studio to **Applications** and then launch it from **Applications** only.

2 | Install the Perfecto plugin

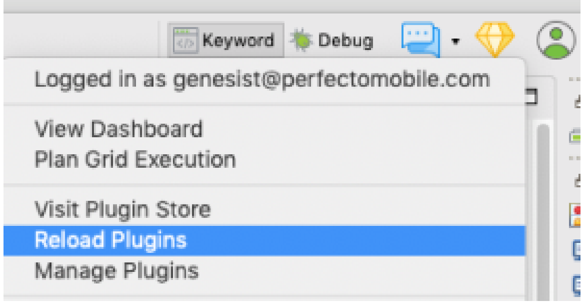
The Perfecto plugin for Katalon offers the following features:

- Automatic generation of custom capabilities that allow the remote driver to connect to Perfecto Cloud devices and automate web, mobile web, and mobile native/hybrid apps.
- Ability to update the test name, tags, result status, failure reason, reportium assertions, project name, and steps in Perfecto Smart Reporting.
- Ability to integrate with the CI Dashboard in Perfecto Smart Reporting.
- Ability to run multiple test cases, test suites, and test suite collections in parallel or sequentially.

You can install the plugin directly or customize it.

To install the plugin directly

1. Sign in to the Katalon store and install the Perfecto plugin directly from: https://store.katalon.com/product/158/Perfecto-Integration#overview-content
2. Open Katalon Studio and sign in with the same credentials.
3. In Katalon Studio, click the **Account** button and select **Reload Plugins**.



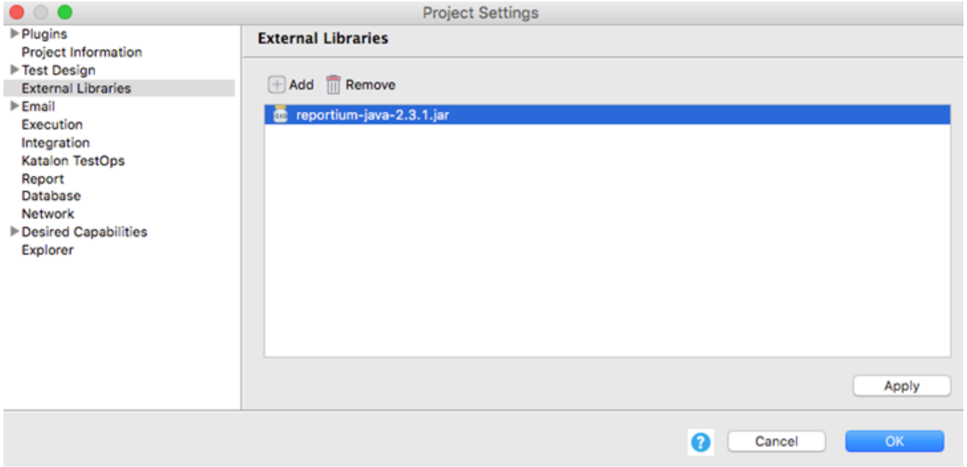
4. Download the latest `reportium-java.jar` from here: https://repo1.perfectomobile.com/public/repositories/maven/com/perfecto/reporting-sdk/reportium-java/
5. Go to **Project > Settings**, select **External Libraries** in the left pane, add the downloaded `reportium-java.jar`, click **Apply**, and then click **OK** to close the **Settings** form.

To customize the plugin

1. Clone or download our Perfecto + Katalon plugin project from GitHub: https://github.com/PerfectoMobileSA/perfecto-katalon-studio-plugin
2. Open the project in Katalon Studio.
3. Build the project by running the following command:

```
gradle clean build katalonPluginPackage
```

4. Copy `build/libs/katalon-studio-perfecto-plugin.jar` and paste it into the **Plugins** folder of your Katalon Studio project to use our plugin.



reportium-java-2.3.1.jar

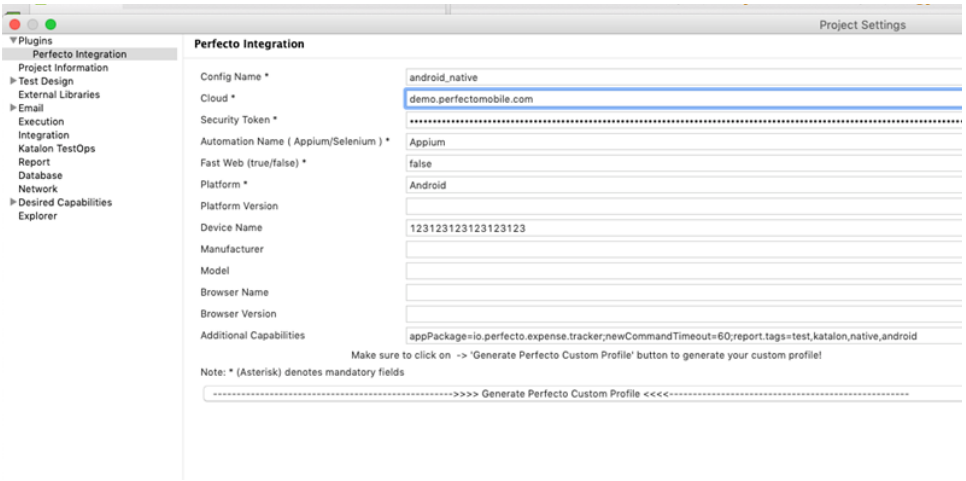
5. Download the latest `reportium-java.jar` from here: https://repo1.perfectomobile.com/public/repositories/maven/com/perfecto/reporting-sdk/reportium-java/
6. Go to **Project > Settings**, select **External Libraries** in the left pane, add the downloaded `reportium-java.jar`, click **Apply**, and then click **OK** to close the **Settings** form.

3 | Create a custom Perfecto profile

This section explains how to create a new custom Perfecto profile. To edit an existing profile, in the **Config Name** field, enter the name of the profile you want to edit; then provide the required capabilities and generate the profile. This overrides the previous capabilities.

To create a custom Perfecto profile:

1. In Katalan Studio, select **Project > Settings**.
2. In the left pane, Expand the **Plugins** section and select **Perfecto Integration**. The **Perfecto Integration** form opens on the right.
3. In the **Perfecto Integration** form, provide information in all the mandatory fields (marked with an asterisk).
   The value in the **Config Name** field must be unique.
   For Appium-related configurations, make sure `enableAppiumBehavior` is set to `true.`



Perfecto Integration

Config Name * android_native
Cloud * demo.perfectomobile.com
Security Token * ••
Automation Name (Appium/Selenium) * Appium
Fast Web (true/false) * false
Platform * Android
Platform Version
Device Name 123123123123123123
Manufacturer
Model
Browser Name
Browser Version
Additional Capabilities appPackage=io.perfecto.expense.tracker;newCommandTimeout=60;report.tags=test,katalon,native,android

Make sure to click on -> 'Generate Perfecto Custom Profile' button to generate your custom profile!

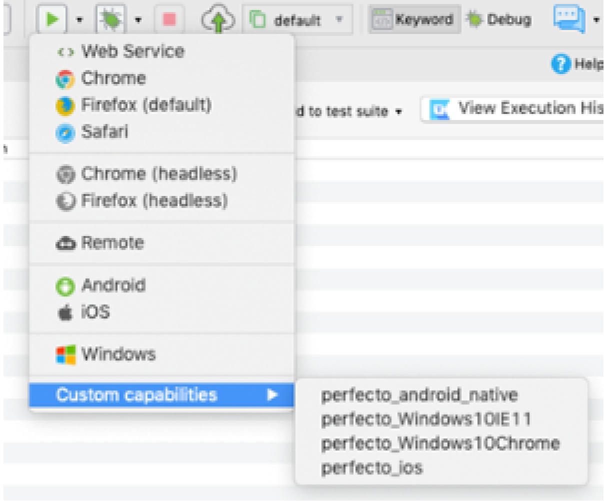
Note: * (Asterisk) denotes mandatory fields

-->>>> Generate Perfecto Custom Profile <<<<--

4. (Optional) To provide additional capabilities in the **Additional Capabilities** field, use a semicolon (`;`) as a separator. For example: `resolution=1024x768;location=US East;report.tags=smoke,regression`
5. To generate the profile, click **Generate Perfecto Custom Profile**.

> **Note:** To save changes, you need to click **Generate Perfecto Custom Profile** each time you edit a new or existing profile.

When generated, each custom profile created based on the unique **Config Name** field is displayed under **Custom Capabilities**. By default, each profile created with the Perfecto plugin has a prefix of `perfecto_`. This helps uniquely identify Perfecto profiles.



Add global variables

You pass project configurations such as application path, job name, and job number as global variables. The Perfecto plugin for Katalon requires global variables to integrate with the CI Dashboard in the Perfecto **Test Analysis** tab (Smart Reporting), as specified in the following table.

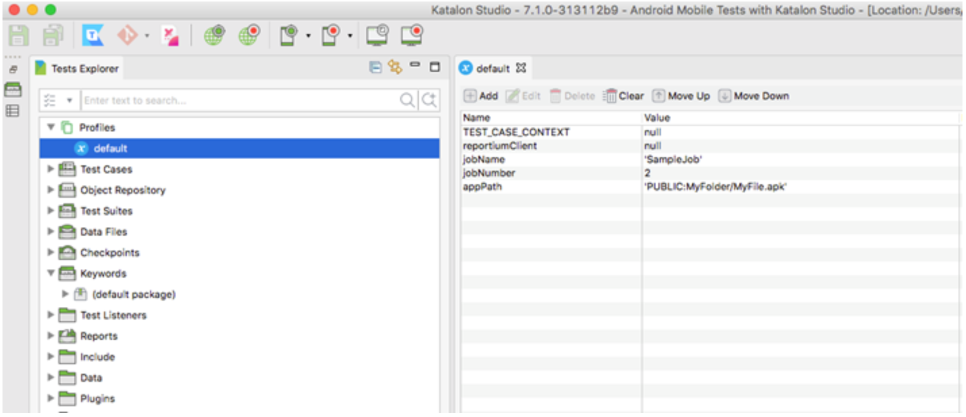
Variable name	Value type	Value
`TEST_CASE_CONTEXT`	Null	Null
`reportiumClient`	Null	Null
`jobName`	String	Name of the job
`jobNumber`	Number	Job number
`projectName`*	String	Project
`projectVersion`*	String	1.0

* Optional variable to group results in Perfecto Smart Reporting

The `appPath` variable is only required for specifying the location of a native or hybrid app in the Perfecto Media repository.

To add global variables:

1. In Katalon Studio, in the **Test Explorer** pane, under **Profiles**, open the default file and click **Add**.
2. Enter the name, select the default type, and enter the default value.
3. Click **OK**.
4. Save the profile.



Name	Value
TEST_CASE_CONTEXT	null
reportiumClient	null
jobName	'SampleJob'
jobNumber	2
appPath	'PUBLIC:MyFolder/MyFile.apk'

To access global variables in the code, use the following syntax:

```
- GlobalVariable.variableName
```

For example:

```
- GlobalVariable.appPath
```

Pre-conditions

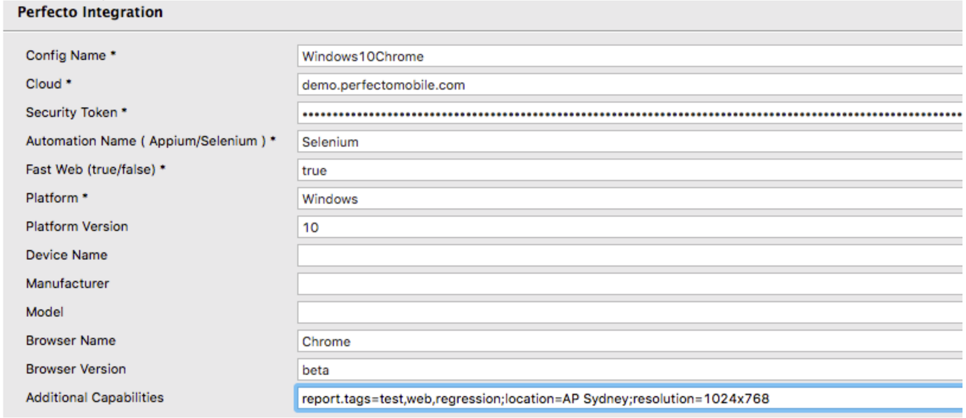
Make sure to adhere to the pre-conditions specified in the following table for web and native/hybrid testing.

	Change	To	Remove occurrence at end of test
Web	`WebUI.openBrowser(“URL”)`	`PerfectoKeywords.openBrowser(“URL”)`	`WebUI.closeBrowser()`
Native /Hybrid	`Mobile.startApplication(“app-path”)`	`PerfectoKeywords.startApplication(“app-path”)`	`Mobile.closeApplication()`

Note: Import `com.kms.katalon.keyword.perfecto.PerfectoKeywords` in **Script** view.

The following images show example configurations.

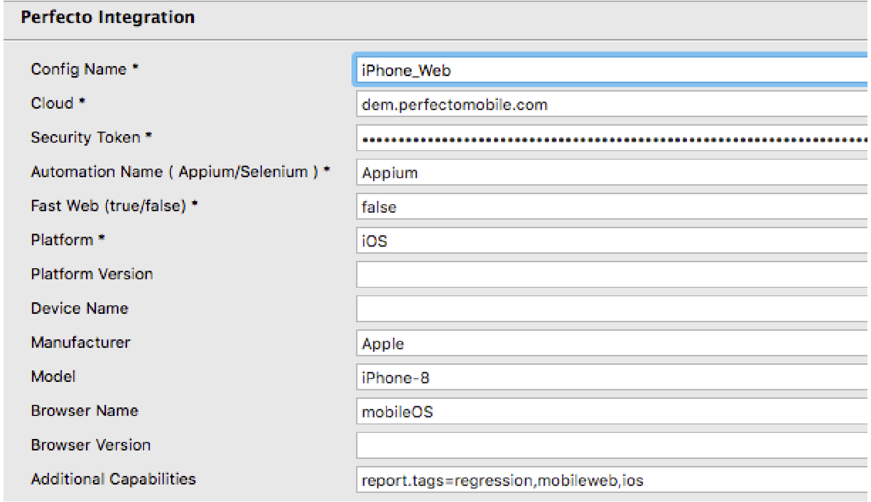
- **Web configuration:**



Perfecto Integration

Config Name *	Windows10Chrome
Cloud *	demo.perfectomobile.com
Security Token *	••
Automation Name (Appium/Selenium) *	Selenium
Fast Web (true/false) *	true
Platform *	Windows
Platform Version	10
Device Name	
Manufacturer	
Model	
Browser Name	Chrome
Browser Version	beta
Additional Capabilities	report.tags=test,web,regression;location=AP Sydney;resolution=1024x768

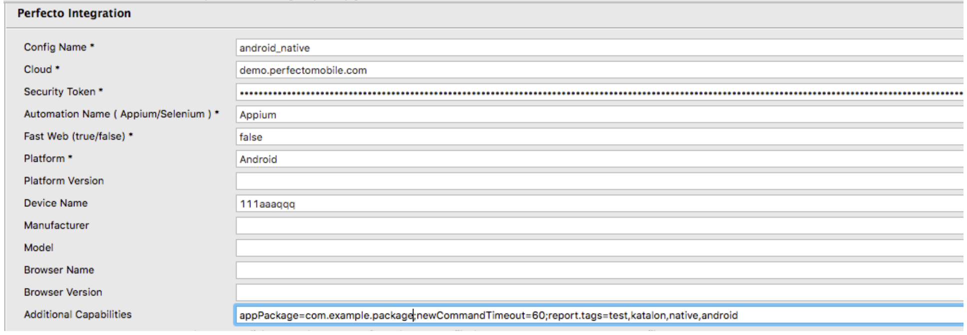
- **Mobile web configuration:**



Perfecto Integration

Config Name *	iPhone_Web
Cloud *	dem.perfectomobile.com
Security Token *	••
Automation Name (Appium/Selenium) *	Appium
Fast Web (true/false) *	false
Platform *	iOS
Platform Version	
Device Name	
Manufacturer	Apple
Model	iPhone-8
Browser Name	mobileOS
Browser Version	
Additional Capabilities	report.tags=regression,mobileweb,ios

- **Perfecto native configuration (legacy):**



Perfecto Integration

Config Name *	android_native
Cloud *	demo.perfectomobile.com
Security Token *	••
Automation Name (Appium/Selenium) *	Appium
Fast Web (true/false) *	false
Platform *	Android
Platform Version	
Device Name	111aaaqqq
Manufacturer	
Model	
Browser Name	
Browser Version	
Additional Capabilities	appPackage=com.example.package;newCommandTimeout=60;report.tags=test,katalon,native,android

Perfecto keywords

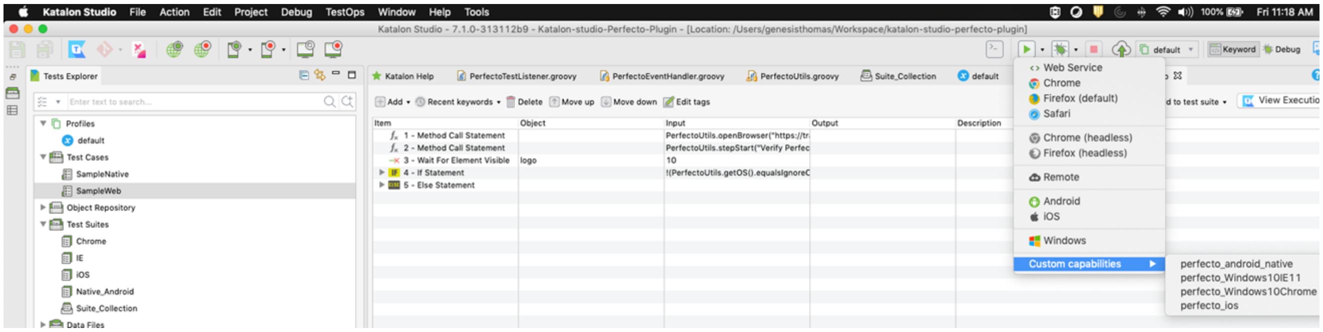
Functionality to implement	Syntax
Perfecto Smart Reporting	`PerfectoKeywords.stepStart("description")`
Perfecto Smart Reporting's `reportiumAssert`	`PerfectoKeywords.reportiumAssert("description", true/false)`
Media upload to the Perfecto Media repository	`PerfectoKeywords.uploadMedia('<<PATH TO LOCAL APK/IPA>>', <<PATH TO PERFECTO MEDIA REPOSITORY>>")`
Perfecto's OCR Click	`PerfectoKeywords.ocrClick('label')`
Perfecto's OCR Find	`PerfectoKeywords.ocrFind ('content', timeout)`

4 | Execute tests

This section walks you through executing a single test case, a test case suite, and a test case suite collection.

To execute a test case:

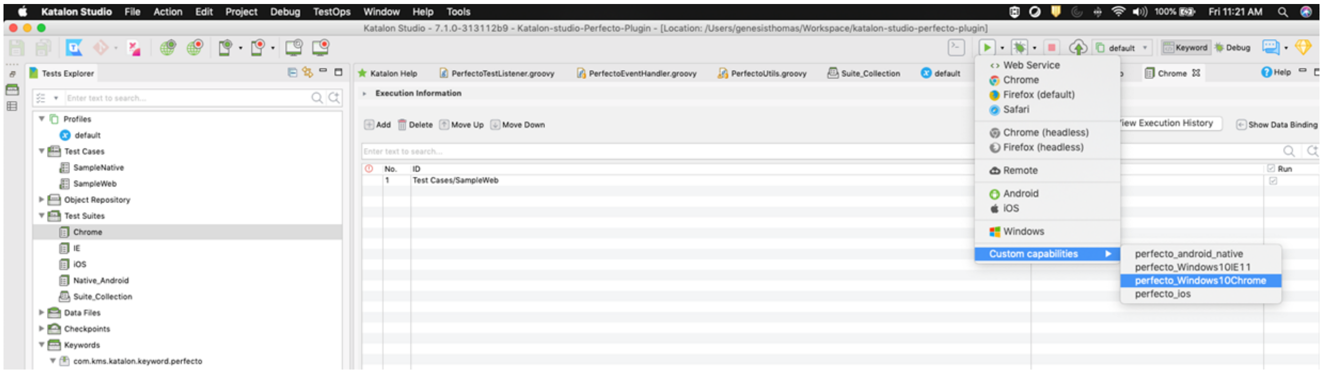
1. Open a test case.
2. Click the run button.
3. From the menu that opens, select **Custom Capabilities > perfecto_*<profile_name>***.



Item	Object	Input	Output	Description
1 - Method Call Statement		PerfectoUtils.openBrowser("https://tr		
2 - Method Call Statement		PerfectoUtils.stepStart("Verify Perfec		
3 - Wait For Element Visible	logo	10		
4 - If Statement		!(PerfectoUtils.getOS().equalsIgnoreC		
5 - Else Statement				

To execute a test suite:

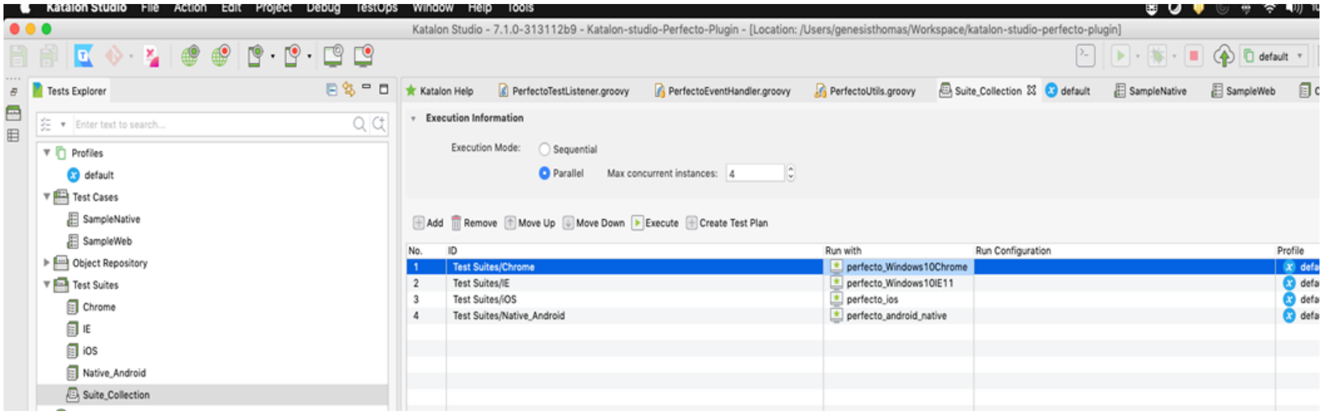
1. Open a test suite.
2. Click the run button.
3. From the menu that opens, select **Custom Capabilities > perfecto_*<profile_name>***.



No.	ID
1	Test Cases/SampleWeb

To execute a test suite collection:

1. Open a test suite collection.
2. In the **Execution Information** form, do the following:
   a. Add a profile for each test suite.
   b. Select the **Execution Mode** as **Sequential** or **Parallel**.
   c. Set the max concurrent instance as applicable.
   d. Click **Execute** to run the entire suite collection.



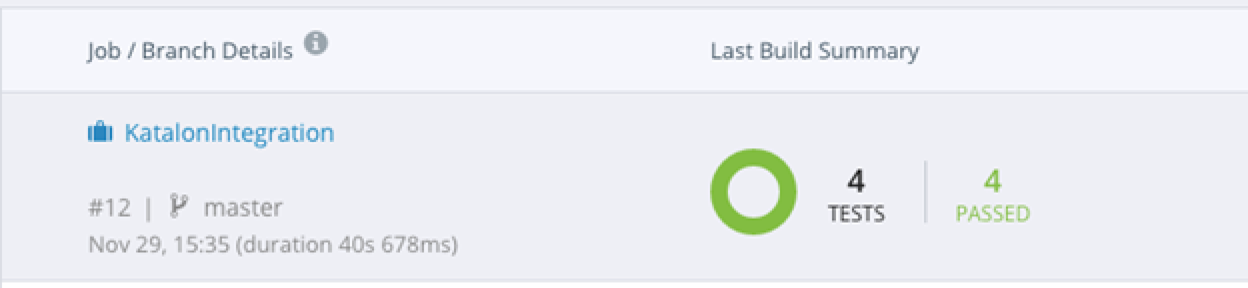
Execution Mode: ○ Sequential ● Parallel Max concurrent instances: 4

No.	ID	Run with	Run Configuration	Profile
1	Test Suites/Chrome	perfecto_Windows10Chrome		defa
2	Test Suites/IE	perfecto_Windows10IE11		defa
3	Test Suites/iOS	perfecto_ios		defa
4	Test Suites/Native_Android	perfecto_android_native		defa

CI Dashboard integration

The following images illustrate how Perfecto Smart Reporting displays the results of the Katalon test runs in the **Test Analysis** view in the Perfecto cloud:

- A summary of how many tests have passed vs. failed
- The **Report Library**, which is a list of all reports, one for each test run, along with status, history information, platform details, and so on
- The **Single Test Report** (STR), a single report with detailed steps and screen captures



Job / Branch Details

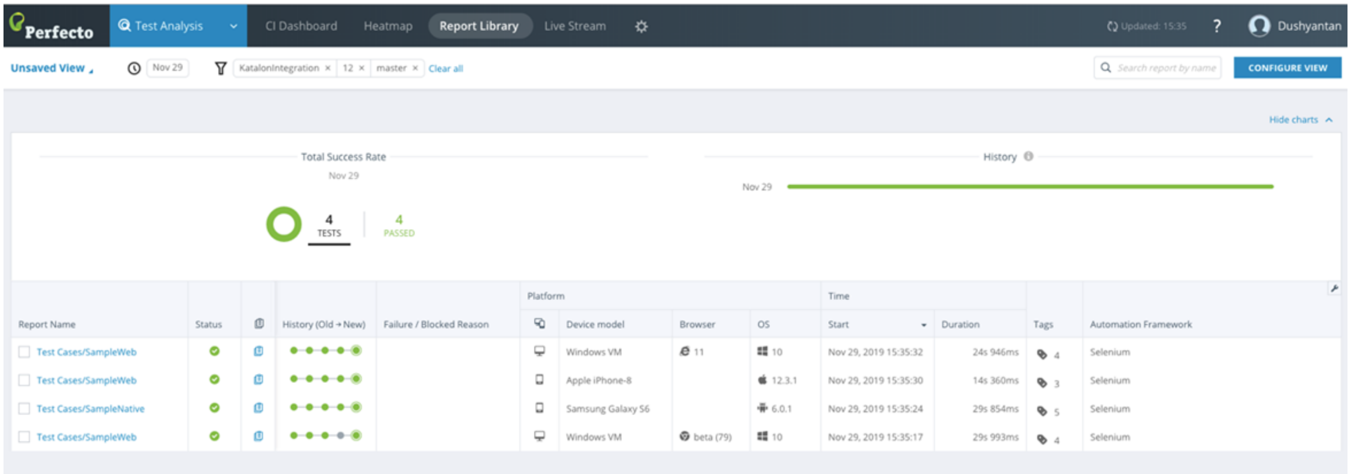
Last Build Summary

KatalonIntegration

#12 | master

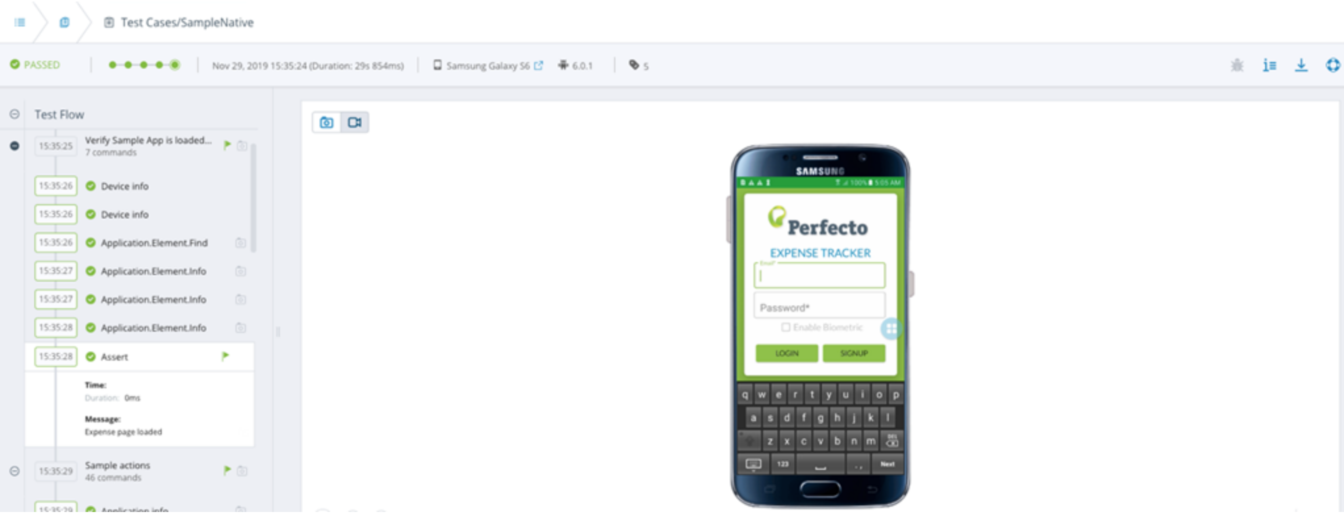
Nov 29, 15:35 (duration 40s 678ms)

4 TESTS | 4 PASSED



4 TESTS | 4 PASSED

					Platform				Time			
Report Name	Status		History (Old → New)	Failure / Blocked Reason		Device model	Browser	OS	Start	Duration	Tags	Automation Framework
□ Test Cases/SampleWeb	✓		●-●-●-●-●			Windows VM	11	10	Nov 29, 2019 15:35:32	24s 946ms	4	Selenium
□ Test Cases/SampleWeb	✓		●-●-●-●-●			Apple iPhone-8		12.3.1	Nov 29, 2019 15:35:30	14s 360ms	3	Selenium
□ Test Cases/SampleNative	✓		●-●-●-●-●			Samsung Galaxy S6		6.0.1	Nov 29, 2019 15:35:24	29s 854ms	5	Selenium
□ Test Cases/SampleWeb	✓		●-●-●-●-●			Windows VM	beta (79)	10	Nov 29, 2019 15:35:17	29s 993ms	4	Selenium



Test Cases/SampleNative

PASSED | Nov 29, 2019 15:35:24 (Duration: 29s 854ms) | Samsung Galaxy S6 | 6.0.1 | 5

Test Flow

15:35:25 Verify Sample App is loaded...
7 commands

15:35:26 Device info

15:35:26 Device info

15:35:26 Application.Element.Find

15:35:27 Application.Element.Info

15:35:27 Application.Element.Info

15:35:28 Application.Element.Info

15:35:28 Assert

Time:
Duration: 0ms

Message:
Expense page loaded

15:35:29 Sample actions
46 commands

15:35:29 Application info