iOS configuration parameters for the Gradle Plugin

When running your XCTest or XCUITest tests with the Perfecto Gradle Plugin, you can add configuration information for the test execution at the following levels:

- Configuration file (recommended): A JSON-formatted file listing parameter assignments (see below).
- ***build.gradle*** file: In the ***perfectoGradleSettings*** clause, include Authentication parameters and indicate the location of the configuration file.
- Command-line: Include any of the parameters in the following format: `-P<paramName>="<paramvalue>"`

The configuration parameters are grouped into the following categories:

On this page:

Perfecto lab authentication parameters

Parameter	Possible Values	Meaning
cloudURL		URL of the Perfecto Lab to connect to. For example, *mobilecloud.perfectomobile.com*
securityToken	Security Token	Tester's personal security token for the Perfecto Lab.

These parameters may be set:

- In the configuration file in the following format:

```
"cloudURL": "mobilecloud.perfectomobile.com"
"securityToken": "AAABNg0ODAoPeNqtkT1PwzAQhnf/CkssM...
JxQ3HEI8NsX02ff"
```

- In the `build.gradle` file in the following format:

```
perfectoGradleSettings {
    cloudURL "mobilecloud.perfectomobile.com"
    securityToken "AAABNg0ODAoPeNqtkT1PwzAQhnf/CkssM...
JxQ3HEI8NsX02ff"
}
```

- In the command-line in the following format:

```
> gradle perfecto-xctest -PcloudURL="mobilecloud.perfectomobile.com" -PsecurityToken="
AAABNg0ODAoPeNqtkT1PwzAQhnf/CkssM...JxQ3HEI8NsX02ff"
```

Device selection parameters

You can select specific devices or specify a number of random devices.

Select specific devices

You can use the following parameters to specify a list of devices:

Obtain the values for the following parameters based on the information in the **Manual Testing** view, as listed for the specific devices to select.

Parameter	Possible Values	Meaning
deviceName		The device ID
description		The device description as defined in the Perfecto Lab.
location		Physical location of the device.

manufacturer		For mobile devices manufacturer of the model. For example: Apple.
model		Device model - for example: iPhone-7
network		Device network, For example AT&T, Verizon
platformName	iOS	Device operating system
platformVersion		Operating system version - for example 11.0.3
resolution		Screen resolution of device

Note: If no device selection parameters are set, the plugin selects a random device from the devices connected to the Perfecto Lab.

These parameters may be set in the configuration file, in the following format:

- Single device selection:

```
"devices": {
        "platformVersion": "10.*"
        "manufacturer" : "Apple"
}
```

- Multiple device selection (in the following example, four devices are selected):

```
"devices" : [
        {"deviceName" : "FA6BR0304878"},
        {
                "platformVersion": "11.*",
                "manufacturer" : "Apple"
        },
        {
                "model": "iPad 4",
                "location": "US-MA-BOS",
                "network": "AT&T, Verizon"
        },
        {}
]
```

Note: The fourth device in this example (`"{}"`) is chosen randomly from all available iOS devices.

Specify the number of random devices

Instead of the ***devices*** clause, you can use the following parameter to indicate that the specified number of iOS devices should be selected and the tests run on the devices in parallel.

Parameter	Possible Values	Meaning
numOfDevices	number in range [1,100]	Number of iOS devices to select. If *numOfDevices* is outside the range, the value will default to select a single device.

This parameters may be set in either the configuration file, the ***perfectoGradleSettings*** clause of the `build.gradle` file, or on the command line, in the following format:

```
"numOfDevices" : 45
```

Selection retry parameters

If the plugin was unsuccessful in allocating a device, you can use the following parameters to configure a retry mechanism built into the plugin:

Parameter	Possible Values	Meaning

acquireDeviceRetryNumber	default = 0	Number of retries that plugin should attempt to allocate the device.
acquireDeviceRetryInterval	default = 30	Number of seconds to wait between each retry attempt

These parameters may be set in either the configuration file, the ***perfectoGradleSettings*** clause of the `build.gradle` file, or on the command line, in the following format:

Configuration File format

```
"acquireDeviceRetryNumber" : 5
"acquireDeviceRetryInterval" : 45
```

build.gradle file

```
perfectoGradleSettings {
        ...
    acquireDeviceRetryNumber  5
        acquireDeviceRetryInterval  45
}
```

Command Line format

```
gradle perfecto ... -PacquireDeviceRetryNumber=5 -PacquireDeviceRetryInterval=45
```

Network virtualization parameters

The Gradle plugin supports configuration of a virtual network environment to execute the XCUITest/XCTest tests.

Use the **networkVirtualization** clause in your *configuration file* or *build.gradle* file to define the parameters to define the characteristics of the emulated network. You can configure the virtual network using the same parameters supported by the Perfecto Network virtualization start command. The following table lists the parameters supported.

Sub-Parameter	Possible Values	Meaning
latency	In the range 0-8000 ms.	Latency applied on packets in the Network.
packetLoss	In the range 0-100%	Network packet loss. A reasonable packet loss value should not exceed 5%.
bandwidthIn	In range 3-100,000 Kbps or unlimited.	Limitation on the allowed download network bandwidth into the device.
bandwidthOut	In range 3-100,000 Kbps or unlimited.	Limitation on the allowed upload network bandwidth from the device.
packetCorruption	In the range 0-100%	Network packet corruption. A reasonable packet loss value should not exceed 5%
packetReordering	In the range 0-100%	The percentage of network packets sent immediately without any delay. Used alongside the **latency** parameter. Moreover, the packets sent immediately will arrive earlier than the packets that were delayed by the defined latency value, essentially creating a packet reordering. A reasonable packet reordering value should not exceed 10%.
packetDuplication	In the range 0-100%	Network packets duplicated. A reasonable packet duplication value should not exceed 3%.
delayJitter	In the range 0-8000 ms.	Random latency variation; the actual latency between latency +- jitter. Used alongside the Latency parameter. For example, if the latency is defined to be 100 ms and the jitter is 10 ms, this causes the added delay to be 100ms - 10ms.

correlation	In the range 0-100%	Network packet value correlation, affecting the Latency, Corruption, Reordering and Duplication. For Latency, Corruption, Reordering and Duplication, the current packet n value will be correlated by this % to previous packet n-1 value. In other words, the current packet value is correlated to the previous packet value. For example, if the correlation is defined to be 25%, the packet loss is 5%, and the previous packet was lost, then the updated packet loss value (for the current packet) would be 75% * 5% + 25%, which equals 28.75%. However, if the previous packet was not lost, then the packet loss value would be 75% * 5%, which equals 3.75%.
blockedDestinations	List of Strings	Network packet block, to specific destinations, defined by domain name, IP address, and IP range destinations in IP Prefix (Slash) notation.
blockedPorts	List of Strings	Network packet block, to specific ports. For example, to block http, define port number 80. To unblock, prefix the value with a '-'. For example, -80.
generateHarFile	true | false	Indicates if a HTTP Archive (HAR) file should be generated for each test to analyze the traffic of the virtual network. See **Note** below.
profile		Suggested network virtualization profiles. See the Network Conditions for supported values

Note: To use the ***generateHarFile*** parameter, first install the certificate and configure the device using Perfecto Automation. This feature is currently in limited release. For details, contact Perfecto Support.

These parameters may be set:

- In the configuration file, in the following format:

```
"networkVirtualization": {
    "latency": 89,
    "packetLoss": 7,
    "bandwidthIn": 5,
    "bandwidthOut": 40
}
```

- In the ***perfectoGradleSettings*** clause of the `build.gradle` file:

```
perfectoGradleSettings {
    ...
    networkVirtualization {
        packetLoss 5
        profile    4g_lte_good
    }
    ...
}
```

Device/application vitals parameters

The plugin supports collecting data regarding the underlying performance of either the application or the device. The vitals information may be used to identify extreme behavior of the application or device. Use the following parameters to indicate the vitals data to collect for the test run.

Parameter	Possible Values	Meaning
vitals	see list in Vitals start command description	The Vitals to collect.
interval	**Recommended:** between 10 and 30.	The data collection frequency, in seconds. Default: 1 sec

These parameters may be set:

- In the configuration file, in the following format:

```
"vitalsMonitoring": {
    "vitals": ["outputs.monitors.memory.used", "outputs.monitors.cpu.total"],
    "interval": 12
}
```

- In the ***perfectoGradleSettings*** clause of the build.gradle file:

```
perfectoGradleSettings {
        ...
        vitalsMonitoring {
                vitals   "all"
                interval 30
        }
        ...
}
```

Application parameters

Parameter	Possible Values	Meaning
instrumentationArgs		Custom arguments to pass to the Instrumentation Runner. format: • Configuration file: **"instrumentationArgs" : ["value","value",...]** • Command line: **-PinstrumentationArgs="value;value;..."**
instrumentationEnvVars		Environment Variable values to pass to the Instrumentation Runner. format: • Configuration file: **"instrumentationEnvVars" : ["key=value","key=value",...]** • Command line: **-PinstrumentationEnvVars="key=value;key=value;..."**
failBuildOnFailure	true | false	Fail the Gradle build if any test fails or there is a device error. Default is **false**.
debug		Run the task in debug mode to output more verbose messages.
appPath		Path to the *ipa* or *app* file for the application and unit tests. If the parameter links to an *.app* file (XCode output), the plugin will first convert this to an *.ipa* file before installing on the device. For details, see XCUITest.
testAppPath		Path to the UI test application runner *ipa* or *app* file. If the parameter links to an *.app* file (XCode output), the plugin will first convert this to an *.ipa* file before installing on the device. For details, see XCUITest.

Note: The ***instrumentationArgs*** indicates that the test methods use an external framework and the associated screenshot tool. For example:

- With the KIF framework, use: **"instrumentationArgs" : ["KIF"]**
- With the EarlGrey tool, use: **"instrumentationArgs" : ["EARLGREY"]**

Format of apkPath/testApkPath values

The path parameters indicate the location of the *.ipa* files of the application and the test application. These may be located in either the local workstation storage or in the Perfecto Lab Repository. To differentiate between these locations, use the following format to indicate a location in the Repository:

repository://[PUBLIC | PRIVATE]**:/**<path to item>

Repository item must have ***.ipa*** type indicator.

The "**repository://**" prefix is mandatory for Repository locations.

Selective testing parameters

In addition, there are **filter** parameters that limit the test scenarios executed during the run, as listed in the following table.

Parameter	Possible Values	Meaning
testClassNames		Array of class names that you wish to run. If not specified, all the classes will run. format: • Configuration file - "testClassNames":["Class","Class",...] • Command line - -PtestClassNames="Class;Class;..."
testMethodNames		Array of method names that you wish to run. If not specified, all methods will run. format: • Configuration file - "testMethodNames":["Class#Method","Class#Method",...] • Command line - -PtestMethodNames="Class#Method;Class#Method;..."
runUITests	true | false	Indicates if the XCUITest runner application should be installed and executed. If the value is false, the XCUITests will not be run. Default is **true** **Note**: Related to XCUITest-based tests only.
runUnitTests	true | false	Indicates if the XCTest methods should be executed. Default is **true**
testTimeout	number of milliseconds	An amount of time allocated to each test case. If a test case exceeds the timeout, it will be reported as "failed". The full test set will continue with the next test case.

When activating the plugin, with ***runUITests*** and ***runUnitTests*** set to **true**, supply both the **appPath** and **testAppPath** parameters.

These parameters may be set:

- In the configuration file, in the following format:

```
"testAppPath": "Users/usern/AppProjects/Playground/app/build/outputs/app-Runner.ipa"
"appPath": "Users/usern/AppProjects/Playground/app/build/outputs/app.ipa"
"testMethodNames": ["DemoClass#demoMethod","DemoClass#demoMethod2"]
```

- In the ***perfectoGradleSettings*** clause of the build.gradle file:

```
perfectoGradleSettings {
        ...
    testAppPath "Users/usern/AppProjects/Playground/app/build/outputs/app-Runner.app"
        appPath "Users/usern/AppProjects/Playground/app/build/outputs/app.ipa"
}
```

- In the command line, in the following format:

```
gradle perfecto-xctest ... -PappPath="Users/usern/AppProjects/Playground/app/build/outputs/app-target.
ipa" -PrunUnitTests="false"
```

KIF/EarlGrey tests

When running test methods from external frameworks, for example *Earl Grey* or *KIF*, based on the XCTest framework, the plugin considers these to be unit tests, even though they may in fact perform UI automation and testing. Therefore:

- Configure the ***runUnitTests*** parameter value to **true.**
- If not including any XCUITest test methods, set ***runUITests*** parameter value to **false**.
- Configure the ***instrumentationArgs*** parameter to identify the framework (see note above).
- Configure the framework to save the screenshots as described here.

Pre-execution and post-execution parameters

The following parameters control whether the application and test applications are uninstalled from the devices, either prior to installing the latest version or at the end of the tests:

Parameter	Sub-parameter	Possible Values	Meaning
installationDetails			The parameters in this Clause affect the installation phase of the test run.
-	**preCleanUp**	true | false	If the parameter is set (***true***) then the *ipa* files of the application and test-application are uninstalled before installing the new version (supplied by above parameters)
	resign	true | false	If the parameter is set (***true***) then the application will be signed with the Perfecto developer signature. If the parameter is ***false***, developer must sign the application with a developer certificate and supply the UDID of the Perfecto lab device. Learn more about signing the iOS applications. Default is ***true***.
preExecution			The parameters in this Clause affect the device used for the test run.
-	**reboot**	true | false	If the parameter is set (***true***) then the device will perform a reboot operation prior to installing the *ipa* files. Take into account that this is a time-consuming action that may delay the execution of the tests.
postExecution			The parameters in this Clause affect the post-execution phase of the test run.
-	**uninstall**	true | false	If the parameter is set (***true***) then the *ipa* files of the application and test-runner are uninstalled after the test has completed execution.

These parameters may be configured:

- In the configuration file, in the following format:

```
"installationDetails" : {
                                                "preCleanUp" : true
                                                "resign" : true
                                                }
"postExecution" : {"uninstall" : false}
"preExecution" : {"reboot" : "true"}
```

- In the ***perfectoGradleSettings*** clause of the module's build.gradle file:

```
perfectoGradleSettings {
        ...
    postExecution {
                uninstall false
        }
        installationDetails {
                preCleanUp true
        }
 }
```

Screenshot parameters

Screenshots are an important tool for analyzing the test actions. The following parameters affect XCUITest test executions.

Parameter	Possible Values	Meaning
takeScreenshotOnTestEnd	true | false	Save screenshot of device at end of the test executions, regardless of the test result status.
takeScreenshotOnTestFailure	true | false	Save screenshot of device at end of the test executions, if the test result status is a failure status.
takeScreenshotOnTestStep	true | false	Save screenshot of device at every logical step in your tests automatically.

These parameters may be set in the configuration file, build.gradle file, or the command-line.

Note: These configuration parameters are applicable **only** to XCUITest test methods. For XCTest unit tests, these may be snapped by the test code, using third-party tools and retrieved by the Gradle plugin for the execution report.

Reporting parameters

The following parameters are used by Smart Reporting to classify the execution report and make it easy to identify the reports for the execution.

Parameter	Possible Values	Meaning
tags		Set of tags to associate with the execution
jobName		CI identification of the build, used for classification of the report in the CI Dashboard.
jobNumber		CI Job Number of the build
projectName		Name of the project - for classification
projectVersion		Version number assigned to the project for this build
branch		Branch name. like additional tag.

These parameters may be set:

- In the configuration file, in the following format:

```
"tags" : ["plugin", "unit-test", "demo"]
"projectName": "playground"
"projectVersion": "1.5"
"jobName": "newFeature"
"jobNumber": "45"
```

- In the ***perfectoGradleSettings*** clause of the build.gradle file:

```
perfectoGradleSettings {
        ...
    projectName "playground"
        projectVersion "1.5"
}
```

- In the command line, in the following format:

```
gradle perfecto-xctest ... -Ptags="plugin;uiTest;demo" -PjobName="newFeature"
```

Export the test execution report

The Perfecto toolset generates a test execution report that can be exported for analysis. It is recommended to:

1. Use these configuration parameters to define ***tags***, ***jobName***, and ***jobNumber*** for the test execution.
2. Export the report data using these information items with the Smart Reporting Public API.

Location of the configuration file

The plugin execution reads the configuration file whose location is indicated by the **configFileLocation** parameter that may be supplied:

- In the ***perfectoGradleSettings*** clause of the **build.gradle** file.

```
perfectoGradleSettings {
        ...
    configFileLocation "C:\temp\XcuiTest\ConfigFile.json"
}
```

- Or in the command line

```
gradle perfecto-xctest ... -PconfigFileLocation="C:\temp\XcuiTest\ConfigFile.json"
```

If the parameter is not supplied:

- Tests will run on a randomly selected available device.
- The application and runner identification parameters (*appPath* and *testAppPath*) must be supplied in either the ***build.gradle*** file or on the command line.