Testing using Calabash and Xamarin Test Cloud in Ruby

Overview

In this tutorial, we will create a mobile test using Calabash and, optionally, the Xamarin Test Cloud.

In this case, the test (specification) is initially created in Jira as a Cucumber Test and afterwards, it is exported using the UI or the REST API.

Please note

With the migration of Xamarin Test Cloud to Visual Studio App Center, some CLI tools were deprecated and/or removed; the same happened with some documentation.

Thus, this tutorial may need to be updated accordingly.

Android and iOS

For the purpose of this tutorial, we will use an Android app as basis. The only change relevant for iOS would be to use the "calabash-ios" command instead of "calabash-android".

Requirements

- Android SDK (or iOS SDK)
- install the Ruby gems: "calabash-android" (or "calabash-ios"), "calabash-cucumber", "xamarin-test-cloud", "rubyzip"

Description

The code for our test will be the code for the basic "Hello World!" application that Android Studio generates.

Create a Cucumber Test, of Cucumber Type "Scenario", in Jira. The test will validate the presence of the "Hello World!" on the device's screen.

hello.feature

```
Feature: App startup banner

  @ABC-131
  Scenario: Message banner after startup
    When I wait for 2 seconds
    Then I see "Hello World!"
```

"calabash-android" provides some steps that can be reused in order to write Cucumber Scenarios/Scenario Outlines; in other words, if you just reuse those steps, you don't have to write any code at all for your tests.

After creating the Test in Jira and associating it with requirements, etc., you can export the specification of the test to a Cucumber .feature file via the REST API or the **Export to Cucumber** UI action from within the Test Execution issue.

The created file will be similar to the one above, but will contain the references to the Test issue key and the covered requirement issue key.

Project setup

In the project base folder, run the following command which will create a basic .feature.

```
calabash-android gen
```

Note that the `ApplicationManifest.xml` must contain the "**android.permission.INTERNET**" permission. See example below:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.smsf.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

Running tests locally

You can run your tests locally using a device emulator or with a real connected device.

```
calabash-android run app\build\outputs\apk\app-debug.apk  -x json -o data.json
```

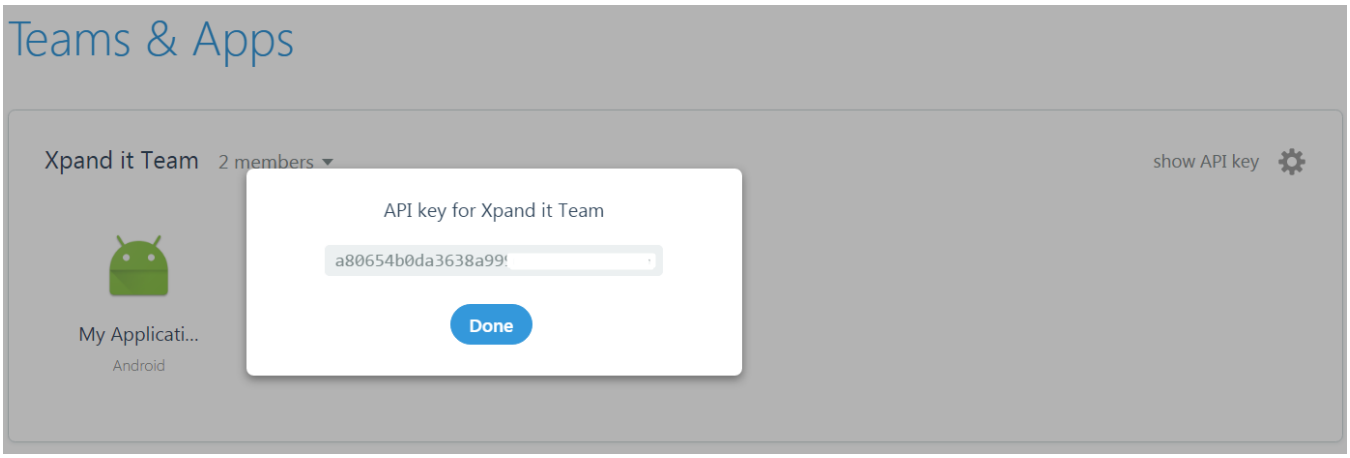
This command will generate a Cucumber JSON report file that can be imported to Xray in the same way you would generally do for Cucumber tests, i.e., via the REST API or the **Import Execution Results** action within the Test Execution.

Running tests in the Cloud with Xamarin Test Cloud

First, you must execute "calabash-android" with the "build" argument.

```
calabash-android build app\build\outputs\apk\app-debug.apk
```

Before submitting the application for testing, you must know the API key on the Test Cloud (obtainable under the team's section).



Afterwards, you can use the "test-cloud" utility to submit the packaged application (e.g., .apk) by identifying the following:

- API key
- Test Cloud's username (i.e., email)
- device's list ID
- series name

Cucumber config file used by the "test-cloud" utility:

config/cucumber.yml

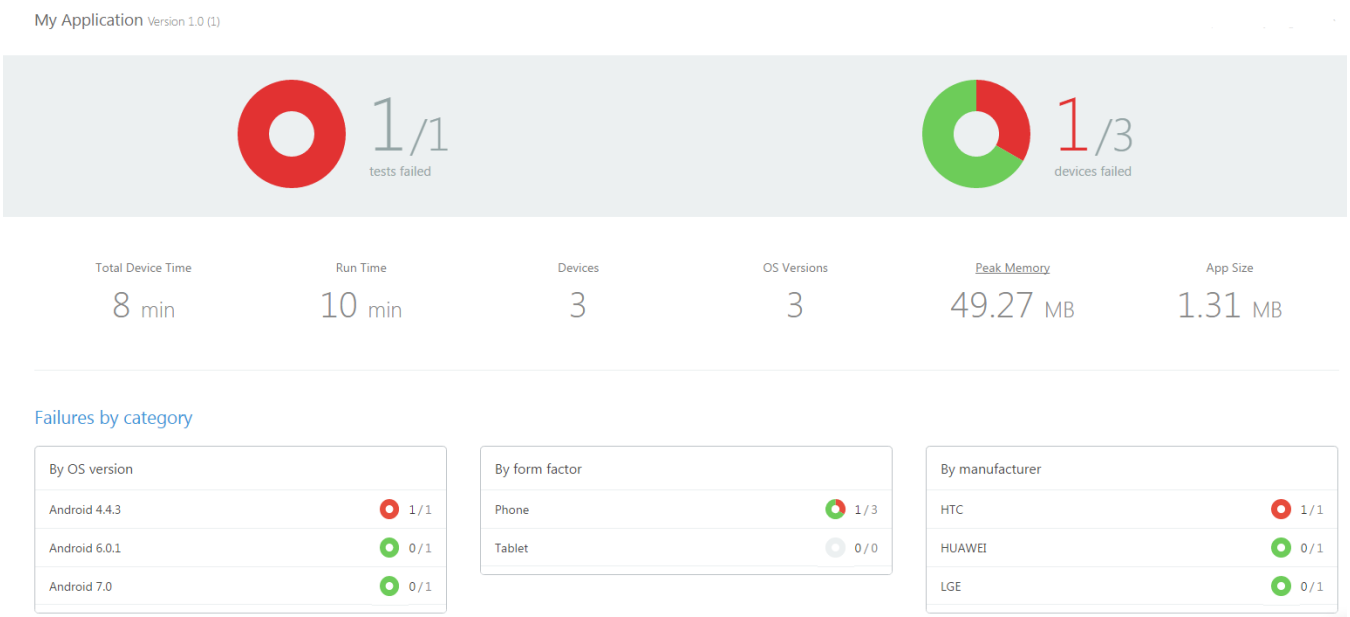
```
android: PLATFORM=android -f json
```

Example syntax or invoking "test-cloud" for submission of the app:

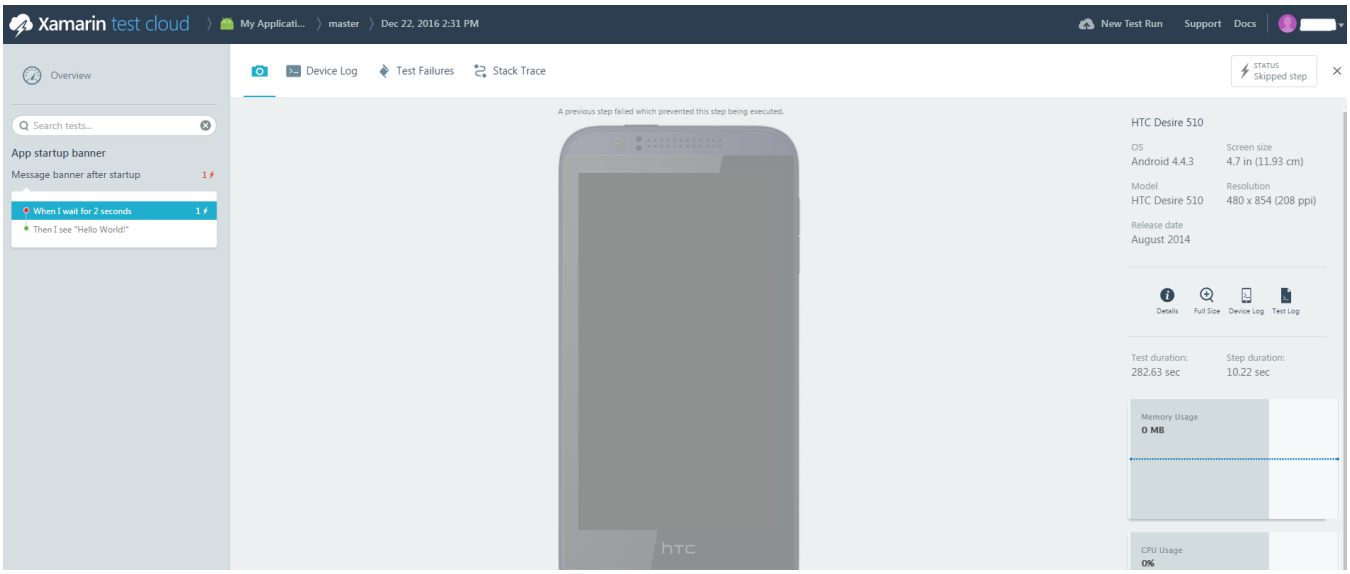
```
 test-cloud submit app\build\outputs\apk\app-debug.apk a80654b0da3638a999a235e7b31d1433  --user tester@example.
com  --devices 13684d00  --series "master" --locale "en_US" --async-json -p android --config config\cucumber.
yml > test-cloud.json
```

As soon as the test run finishes in Xamarin's Test Cloud, the detailed report will be available on the Test Cloud's page.

Some relevant numbers are displayed, such as the total devices and tests with failures.



You can analyze a given test failure: see what step failed, the device and test logs.



Importing results to Xray

Note that Xray supports the submission of a zip file containing multiple Cucumber JSON reports, one per each device.

In order to obtain Test Cloud's results in a machine-friendly way, you need to use Test Cloud's REST API since the "test-cloud" utility does not currently provide an immediate way of obtaining the results.

However, it's possible to use the preliminary Test Cloud Ruby client SDK and API in order to produce a ZIP file containing the Cucumber JSON reports.

Below is an example of an implementation where you pass the API key and the "test-cloud" output file as arguments, and that will produce a "results.zip" file.

obtain_tc_results.rb

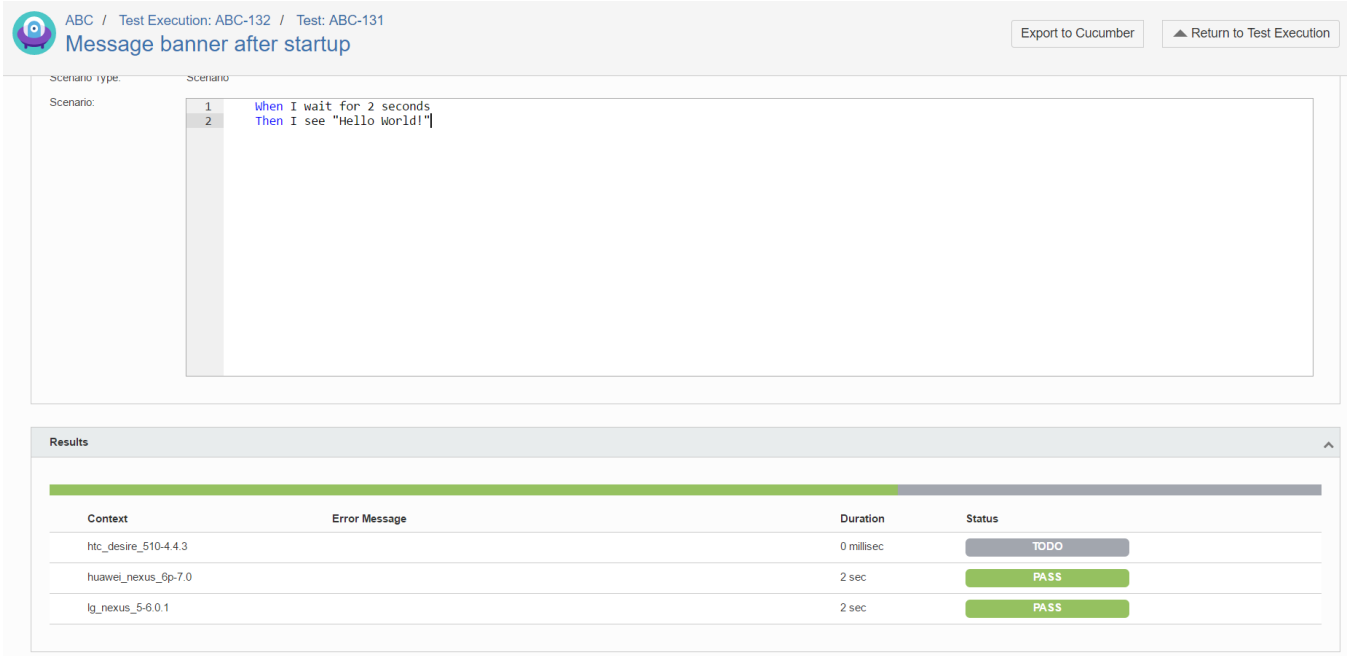
```
require 'tmpdir'
require 'open-uri'
require 'zip'
require './client.rb'

def download_file(url, dir, filename)
    open("#{dir}/#{filename}", 'wb+') do |file|
        file << open(url).read
    end
end
api_key = ARGV[0]
testcloud_json_file = ARGV[1]
#api_key = "xxx"
#testrun_id = "xx"

testrun_id = JSON.parse(File.readlines(testcloud_json_file).last)["test_run_id"]
zipfile_name = "results.zip"

client = Xamarin::TestCloud::Api::V0::Client.new(api_key)
while !client.test_runs.results(testrun_id).finished
 puts "waiting for results..."
 sleep 5
end
results = client.test_runs.results(testrun_id)

File.unlink zipfile_name
tmp_dir = Dir.mktmpdir
begin
  results.logs.devices.each do |log_device|
      download_file(log_device.cucumber_json, tmp_dir, "#{log_device.device_configuration_id}.json")
  end
    input_filenames =  Dir.entries("#{tmp_dir}").select {|f| !File.directory? f}
    Zip::File.open(zipfile_name, Zip::File::CREATE) do |zipfile|
        input_filenames.each do |filename|
            zipfile.add(filename, tmp_dir + '/' + filename)
        end
    end

ensure
  FileUtils.remove_entry tmp_dir
end
```

Creating a zip file with all Cucumber JSON reports:

```
 obtain_tc_results.rb  a80654b0da3638a999a235e7b31d1111 test-cloud.json
```

After importing the results using the REST API, the execution screen details will provide information on the test run result, grouped by device.

The **Context** section will be by filled out with the name of the original Cucumber JSON report; in this case, it's the name/id of the device.



Learn more

Please refer to Testing in BDD with Gherkin based frameworks (e.g. Cucumber) for an overview on how to use Cucumber Tests with Xray.

References

- https://developer.xamarin.com/guides/testcloud/calabash/
- https://developer.xamarin.com/guides/testcloud/introduction-to-test-cloud/
- https://github.com/calabash/calabash-android
- https://github.com/calabash/calabash-ios
- http://bitbar.com/how-to-setup-and-get-started-with-calabash/
- ~~https://github.com/xamarin/test-cloud-samples~~
- https://github.com/rubyzip/rubyzip
- Automated Tests (Import/Export)
- Exporting Cucumber Tests - REST