[Parallel execution of Consumervenice Onboarding UITests on multiple simulator using Xcode 9 beta](https://engineering.paypalcorp.com/blog/parallel-execution-of-consumervenice-onboarding-uitests-on-multiple-simulator-using-xcode-9-beta/" \o "Parallel execution of Consumervenice Onboarding UITests on multiple simulator using Xcode 9 beta)

With many exciting changes revealed in WWDC this year, Parallel UI testing and multiple simulator supports are some of the most important and sought after features in testing world. Currently, UI Testing is being done using cucumber and frank integration in Consumervenice iOS application. And even though, it does the job, it doesn’t do it efficiently due to the sequential nature of UI Test run on single simulator, which is time consuming and inconvenient.  After two weeks of research and web crawling, what was taken up as an intern project “Parallelizing the UI Tests on **single host**” turned out to be impossible using current cucumber frank integration. ([https://engineering.paypalcorp.com/blog/parallel-ui-testing-of-ios-application-on-multiple-simulator](https://engineering.paypalcorp.com/blog/parallel-ui-testing-of-ios-application-on-multiple-simulator/)). Luckily, Apple came up with the official support for multiple simulators in WWDC 17 and **XCTests** became the most feasible option for the task considering some of the criteria below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **XCtests** | **Cucumber** | **Appium** |
| **Natively supported** | ✅ | ❌ | ❌ |
| Supports latest iOS | ✅ | ❌ | ? |
| **Parallel execution** | ✅ | ❌ | ❌ |
| Can reuse Android tests | ❌ | ✅ | ✅ |
| **Test recording** | ✅ | ❌ | ❌ |
| Grouping test flows | ✅ | ❌ | ❌ |
| Supports CI | ✅ | ✅ | ? |
| **Easy to start for iOS Devs** | ✅ | ❌ | ❌ |
| Applicable for RC builds | ? | ✅ | ? |

Although one important thing to note here is that even though apple claimed to support parallel testing on multiple simulators, for the most part, I only found support for **running the same test suits on multiple simulators using multiple –destination tags** in xcodebuild. And although useful, that is **not what I wanted for Consumervenice**. I wanted to run different test suits in parallel in order to reduce the time taken to complete the tests. Initially I started with dummy app and achieved the same using multiple simultaneous xcode build commands run on parallel on multiple simulator ( <https://engineering.paypalcorp.com/blog/step-by-step-guide-for-running-ui-automation-tests-in-parallel-using-xcode-9-beta)>

However, when the same strategy was used for Consumervenice, I encountered countless errors in the build process. After careful consideration, I found out that Consumervenice iOS app uses so many external dependencies and simultaneous build commands fail because of that.

Solution: Leverage the **xcodebuild build-for-testing** feature introduced last year at WWDC. This will in turn build the application only for testing, and generate **xctestrun**file which can be used for testing the application without building using **xcodebuild test-without-building.**But wait, how to specify the test targets to test? Using **-only-testing** tags. Specify the test target and destination pair along with test-without-building command and run the same command with different targets and destinations.

<https://www.youtube.com/watch?v=DSaZREfIuGk&feature=youtu.be>

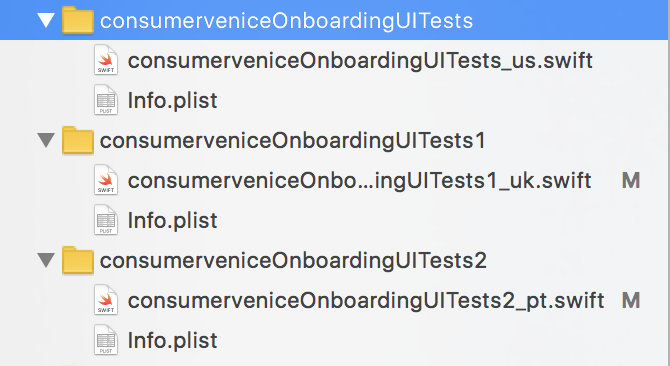
Limitations:

* Xcode 9 beta 3 is used for the current implementation, which is very unstable for now. It requires a lot of configuration and simulator hacks to run the XCTests for Consumervenice app. We must wait for official Xcode 9 release (September 2017) which hopefully will fix those issues.
* Whether we can run Xcode UI tests on RC build from CI is an area with uncertainties.
* Parallel test run execution is not working as expected yet, it has a delay in switching between running simulators. According to Apple, it should not have looked like this. Thus, probably, it will be improved in official Xcode 9 release. For now, the highly expected reduction of time is not achieved and time taken is more in line with the sequential run as each individual simulator gets slower when run in parallel. [Any ideas or thoughts on this is highly appreciated]
* We must find a way to remove an application from Simulator between tests execution in order to properly test push notifications allowance popup. But there is no elegant way to do that as of now. Although we managed to wipe out the user data and always start the tests from homepage using some hacks (Step 4 of article mentioned:  <https://engineering.paypalcorp.com/blog/configuring-xcode-9-to-run-consumervenice-ios-app/> )
* Integration of new Xcode build instruments with CI is still in question
* I have successfully migrated onboarding tests for all the counties except Switzerland. Unfortnately, in case of Switzerland, simulator can’t open the optionchooser for state field in testing mode (option chooser acts a bit weird in testing mode and degrades the simulator performance by huge factor. Luckily, the hacks mentioned above provides a workaround for that, except for Switzerland)

How to do it?

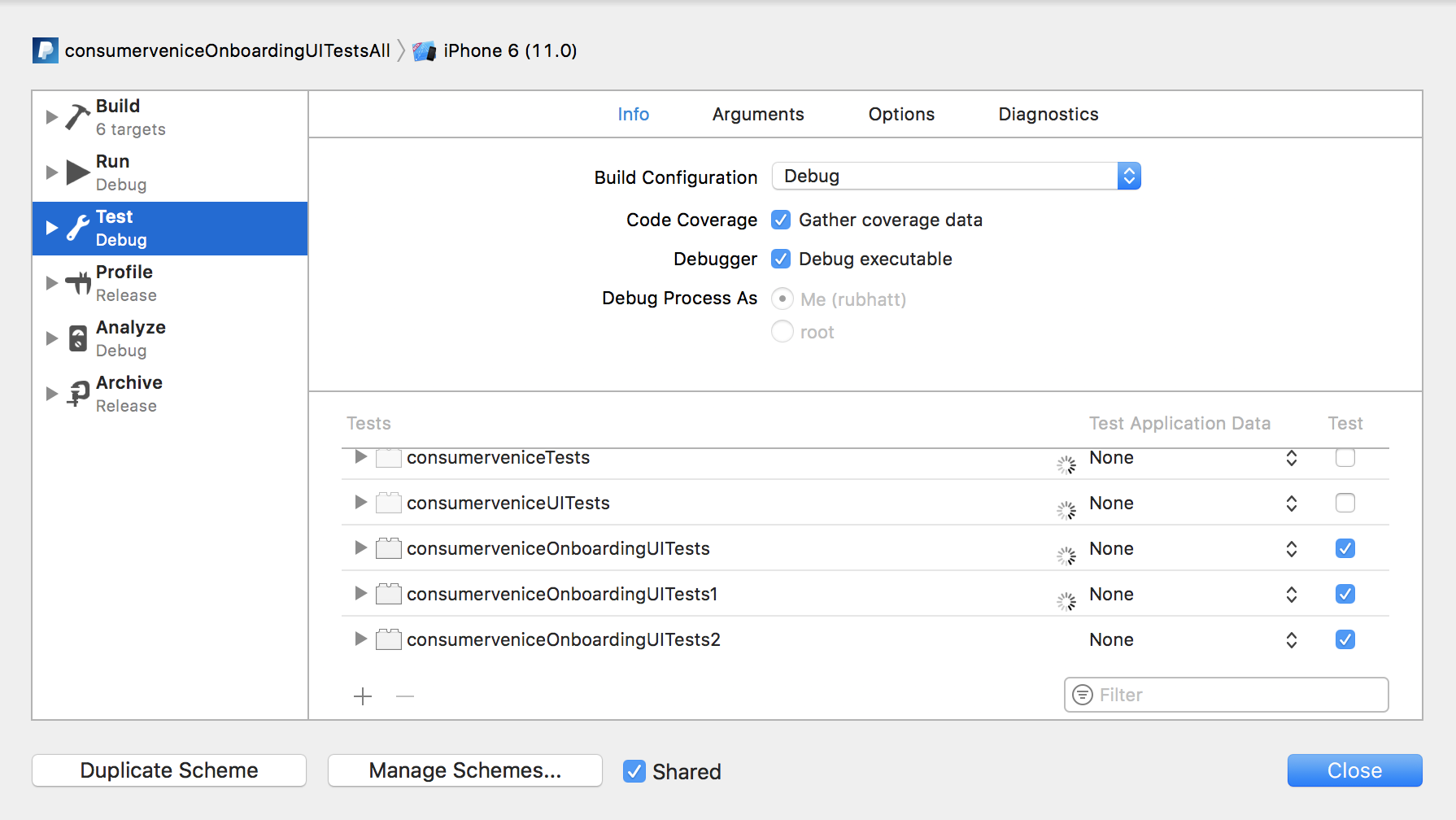
First thing you need to do is configure Xcode 9 to run Consumervenice iOS application which requires some monkey patches in the code. Follow my internal blogpost [**here**](https://engineering.paypalcorp.com/blog/configuring-xcode-9-to-run-consumervenice-ios-app/) and run through all the steps one by one. **Make sure you follow steps 3 and 4** which are specifically to run XCTests in simulator. Unfortunately, Xcode 9 beta has many unreported issues with XCTests and simulators, and the hacks provided in step 4 helps us with workarounds and get the desired performance.

Next thing is to create new UI Targets based on your requirement and split your tests accordingly. The idea is to create n targets if you want to run n simulators simultaneously. So create the folder structure like below and write and split your tests in those files so that each target takes almost the same time to execute, hence giving the even distribution among all the simulators.



You can use the codebase below for the reference. Codebase: <https://github.paypal.com/rubhatt/consumer-venice-ios>

Finally, you need to create one common scheme that includes all those created targets as testing target. You can then build that scheme for testing and use the -only-testing feature to specify target on device.



You are almost done. All you have to do is create n simulators based on your requirements, launch them and run the following command.

Command:

One time build:

**xcodebuild build-for-testing -workspace “/Users/rubhatt/Documents/PayPalApp/consumer-venice-ios/consumervenice.xcworkspace” -scheme “consumerveniceOnboardingUITestsAll” -destination “platform=iOS Simulator,name=iPhone 6,OS=11.0” -derivedDataPath “/Users/rubhatt/Documents/PayPalApp/consumer-venice-ios/build”**

test in parallel using following command in terminal:

**xcodebuild test-without-building -xctestrun “build/Build/Products/consumerveniceOnboardingUITestsAll\_iphonesimulator11.0-x86\_64.xctestrun” -destination “platform=iOS Simulator,id=<sim1 UUID>” ‘-only-testing:consumerveniceOnboardingUITests/consumerveniceOnboardingUITests\_us’ & xcodebuild test-without-building -xctestrun “build/Build/Products/consumerveniceOnboardingUITestsAll\_iphonesimulator11.0-x86\_64.xctestrun” -destination “platform=iOS Simulator,id=<sim2 UUID>” ‘-only-testing:consumerveniceOnboardingUITests1/consumerveniceOnboardingUITests1\_uk’ &** **xcodebuild test-without-building -xctestrun “build/Build/Products/consumerveniceOnboardingUITestsAll\_iphonesimulator11.0-x86\_64.xctestrun” -destination “platform=iOS Simulator,id=<sim3 UUID>” ‘-only-testing:consumerveniceOnboardingUITests2/consumerveniceOnboardingUITests2\_pt’ &**

To ease up the future runs, you can create a [script file](https://www.hastac.org/blogs/joe-cutajar/2015/04/21/how-make-simple-bash-script-mac) and then the run the script every time you need to execute the tests.

References:

1. <https://engineering.paypalcorp.com/blog/configuring-xcode-9-to-run-consumervenice-ios-app/>
2. <https://medium.com/@t.camin/parallelizing-ui-tests-28c16000f141>
3. <https://medium.com/xcblog/speed-up-ios-ci-using-test-without-building-xctestrun-and-fastlane-a982b0060676>
4. <http://shashikantjagtap.net/wwdc-2016-continuous-integration-with-xcode-server-and-xctest-updates/>
5. <http://shashikantjagtap.net/hands-xcuitest-features-xcode-9/>