



Document version: v1.006

Snapshot. What a great piece of software. It automates screenshots. It may not seem like much, but when you think that you have to cover up to 6 devices, even a few screens can easily add up to 30 or more. And if you consider multiple languages, it starts to get rather tedious, and time consuming.

Enter Snapshot. It's pretty simple to install and use, but for those who aren't familiar with some of the methods, it might be a bit daunting. Let's try and simplify it, so everyone can utilize this fantastic method of automation.

Snapshot is a part of 'Fastlane', a complete automation solution for parts of Android and iOS app prep. I suggest you get Fastlane, and use Snapshot as a part of it, but this tutorial just focuses on the issues I've seen setting up Snapshot.

I hope it's not too confusing; I'm writing about 'snapshot', but "I" am using it as part of fastlane, and not alone. Please try to understand the differences when looking at the examples. However, launching from the command line is pretty similar.

Installing Snapshot or Fastlane

You install snapshot from the command line. Instructions (that should be used in conjunction with these instructions) are available at:

<https://github.com/fastlane/fastlane/tree/master/snapshot>

Follow the same instructions as the installation notes;

1) using terminal, type 'xcode-select --install' <return>

It takes a few minutes, depending on machine, internet, etc...

2) using terminal, type 'sudo gem install snapshot' <return>



Setting up your 'Snapshot' environment for Buzztouch projects

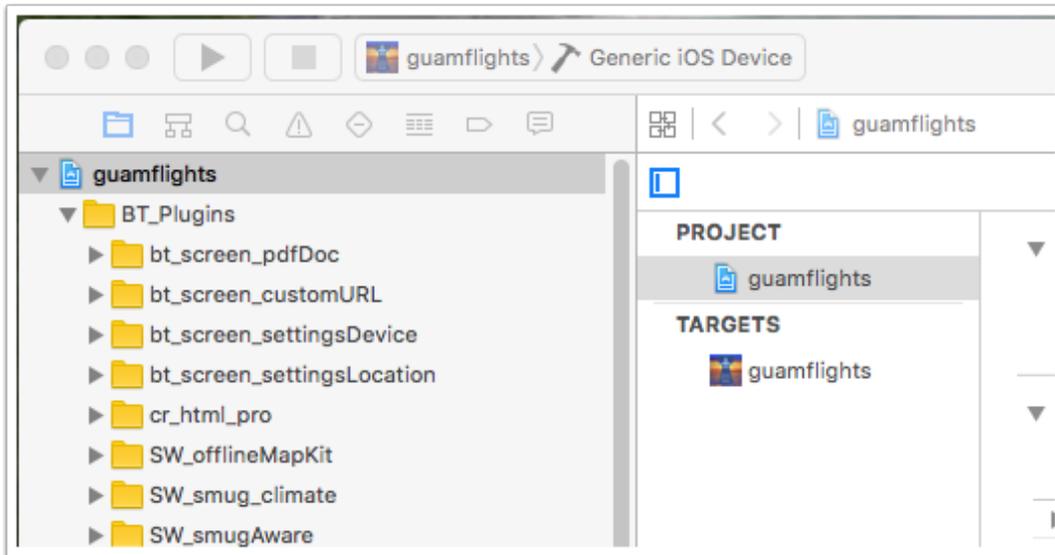
This takes a few minutes, but after everything is installed it should have a message like 'XX gems installed'. Mine said '12 gems installed'. yours may be different, depending on how few/many other kinds of packages you've installed in the past.

```
Last login: Thu Apr 21 12:17:49 on console
[SmugBookPro:~ SmugWimp$ xcode-select --install
xcode-select: note: install requested for command line developer tools
[SmugBookPro:~ SmugWimp$ sudo gem install snapshot
[Password:
Fetching: commander-4.3.5.gem (100%)
Successfully installed commander-4.3.5
Fetching: credentials_manager-0.15.1.gem (100%)
Successfully installed credentials_manager-0.15.1
Fetching: plist-3.1.0.gem (100%)
Successfully installed plist-3.1.0
Fetching: rubyzip-1.1.7.gem (100%)
Successfully installed rubyzip-1.1.7
Fetching: multipart-post-2.0.0.gem (100%)
Successfully installed multipart-post-2.0.0
Fetching: faraday-0.9.2.gem (100%)
Successfully installed faraday-0.9.2
Fetching: sentry-raven-0.15.6.gem (100%)
Successfully installed sentry-raven-0.15.6
Fetching: terminal-table-1.4.5.gem (100%)
Successfully installed terminal-table-1.4.5
Fetching: fastlane_core-0.41.3.gem (100%)
Successfully installed fastlane_core-0.41.3
Fetching: rouge-1.10.1.gem (100%)
```



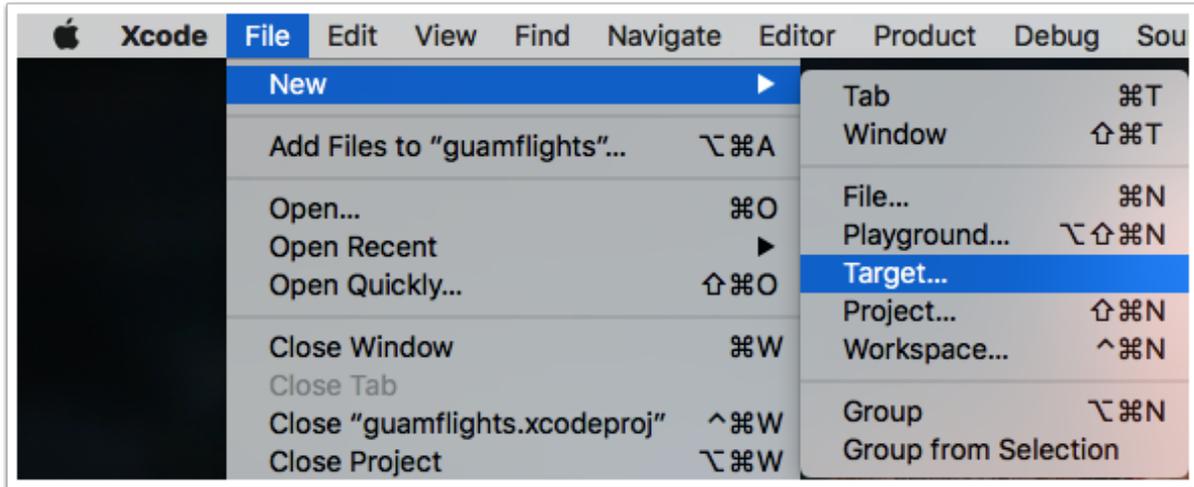
Add a 'test' target to your desired project

You'll need to do this for each project you want to create snapshots with.



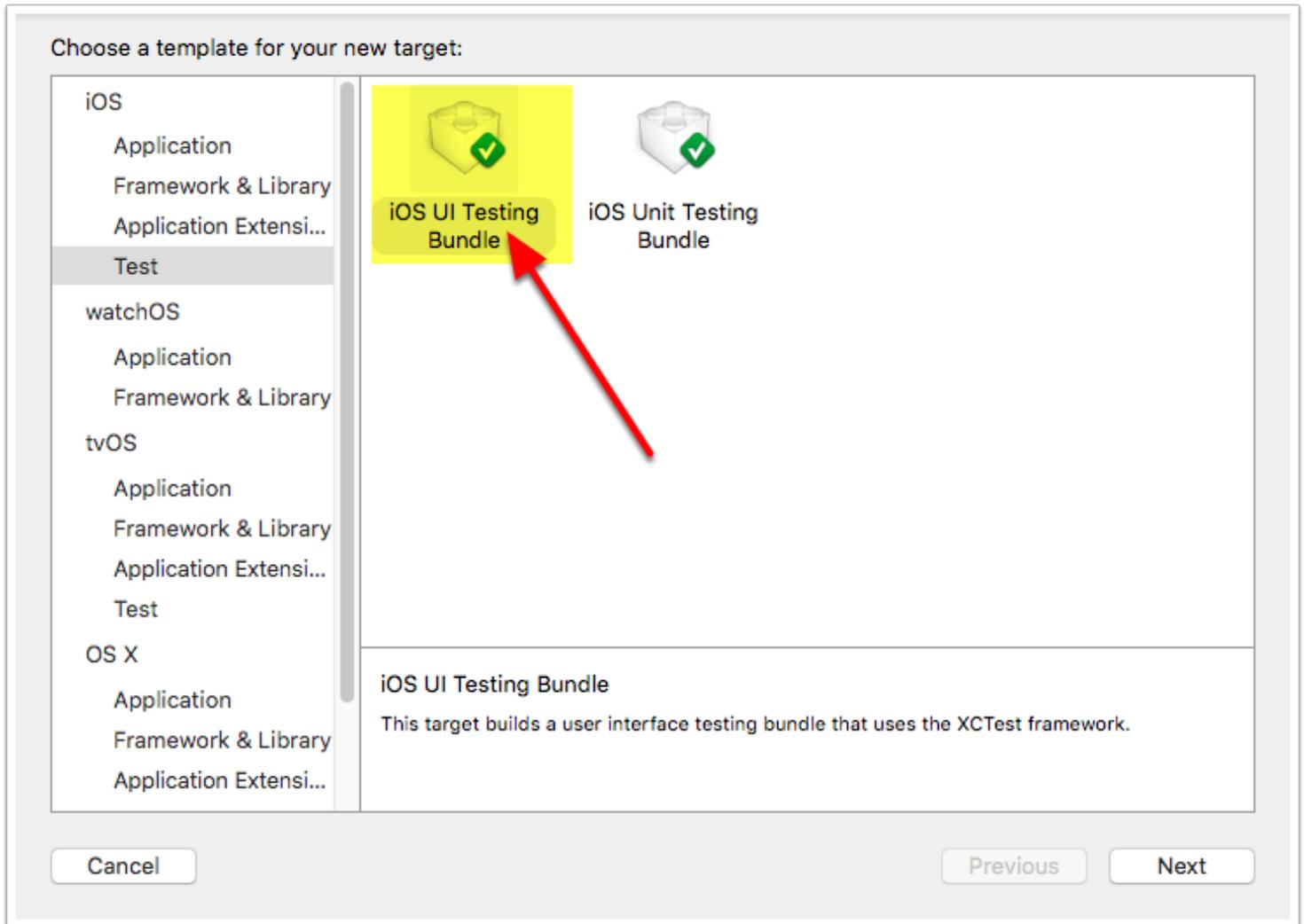


Add New Target





Choose iOS UI Testing Bundle





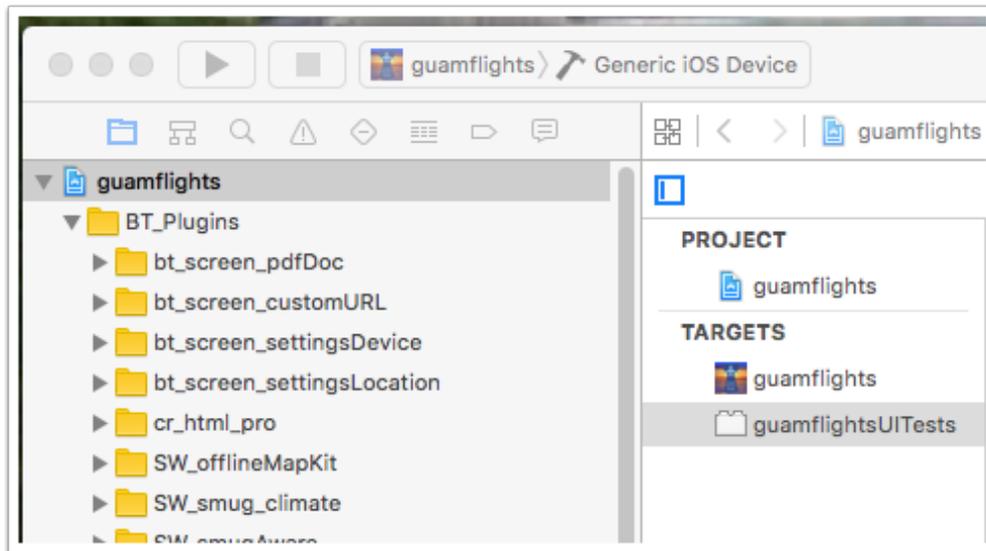
Leave as default

Choose options for your new target:

Product Name:	<input type="text" value="guamflightsUITests"/>
Organization Name:	<input type="text" value="Marianas GPS, LLC"/>
Organization Identifier:	<input type="text" value="com.mgps.utility"/>
Bundle Identifier:	<input type="text" value="com.mgps.utility.guamflightsUITests"/>
Language:	<input type="text" value="Objective-C"/>
Project:	<input type="text" value="guamflights"/>
Target to be Tested:	<input type="text" value="guamflights"/>



You now have a 'test' target. Congrats. Have a cool refreshing drink.





Back to the command line...

In your terminal window, type 'snapshot init' in your project root directory (the same directory that has 'appName.xcodeproj')

Your results 'should' be similar to those in the image. If not, please go back and check that you have not skipped any steps.

```
guamflights-iOS-BTv3.0 — -bash — 75x24
drwxr-xr-x  3 SmugWimp  staff   102 Mar 25 20:21 ja.lproj
drwxr-xr-x  3 SmugWimp  staff   102 Mar 25 20:22 ko.lproj
drwxr-xr-x@ 6 SmugWimp  staff   204 Sep 21  2015 mapbox
drwxr-xr-x  3 SmugWimp  staff   102 Mar 25 21:11 ru.lproj
[drwxr-xr-x  3 SmugWimp  staff   102 Mar 25 20:14 zh-Hant.lproj ]
drwxr-xr-x  3 SmugWimp  staff   102 Mar 25 20:14 zh.lproj
SmugBookPro:guamflights-iOS-BTv3.0 SmugWimp$ snapshot init
Successfully created SnapshotHelper.swift './SnapshotHelper.swift'
Successfully created new Snapfile at './Snapfile'

-----
Open your Xcode project and make sure to do the following:
1) Add the ./fastlane/SnapshotHelper.swift to your UI Test target
   You can move the file anywhere you want
2) Call `setupSnapshot(app)` when launching your app

    let app = XCUIApplication()
    setupSnapshot(app)
    app.launch()

3) Add `snapshot("@Launch")` to wherever you want to create the screenshots

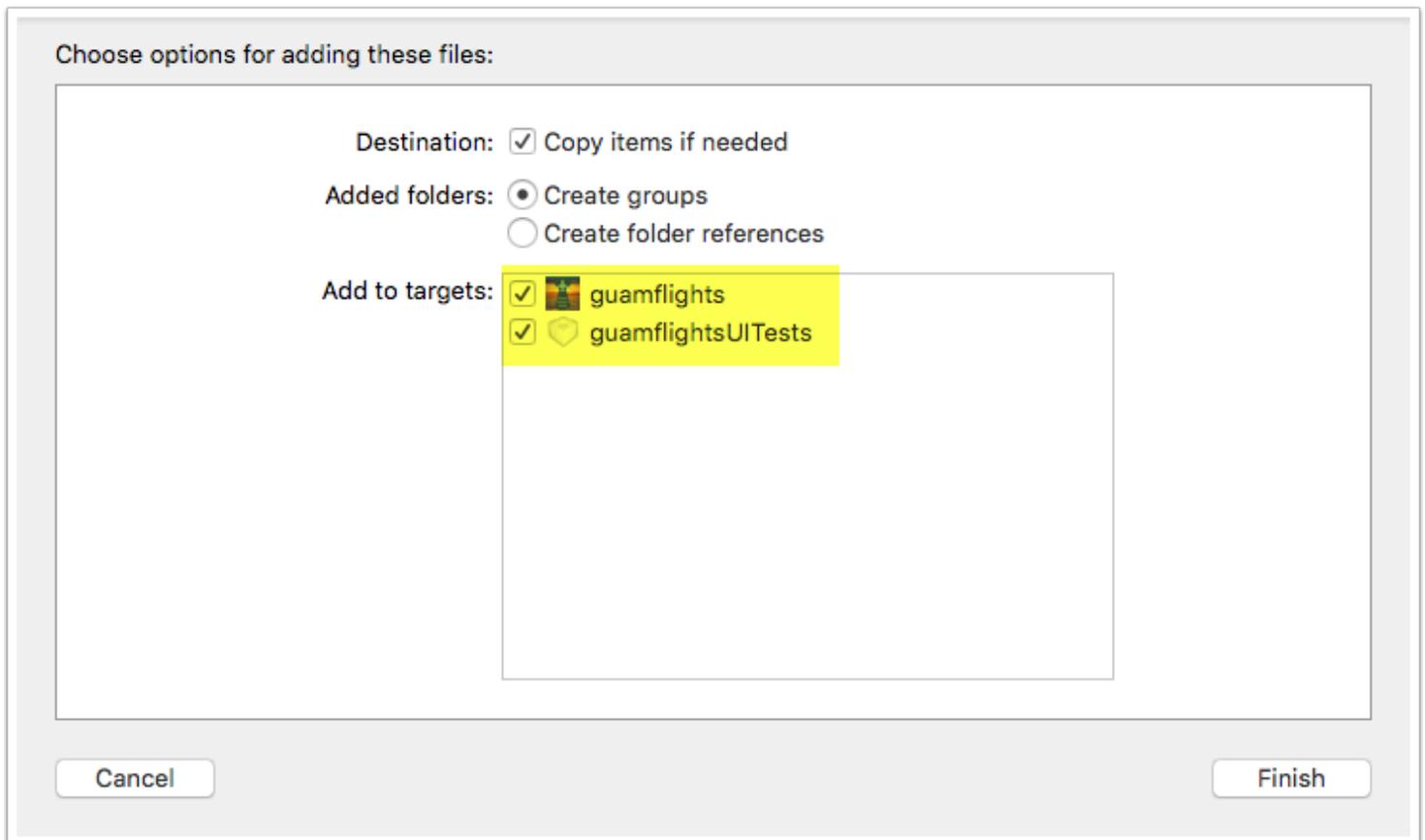
More information on GitHub: https://github.com/fastlane/fastlane/tree/master/snapshot
SmugBookPro:guamflights-iOS-BTv3.0 SmugWimp$
```



Adding your Snapshot files to your project

After running the command, it will generate two files; 'SnapshotHelper.swift' and 'Snapfile'.

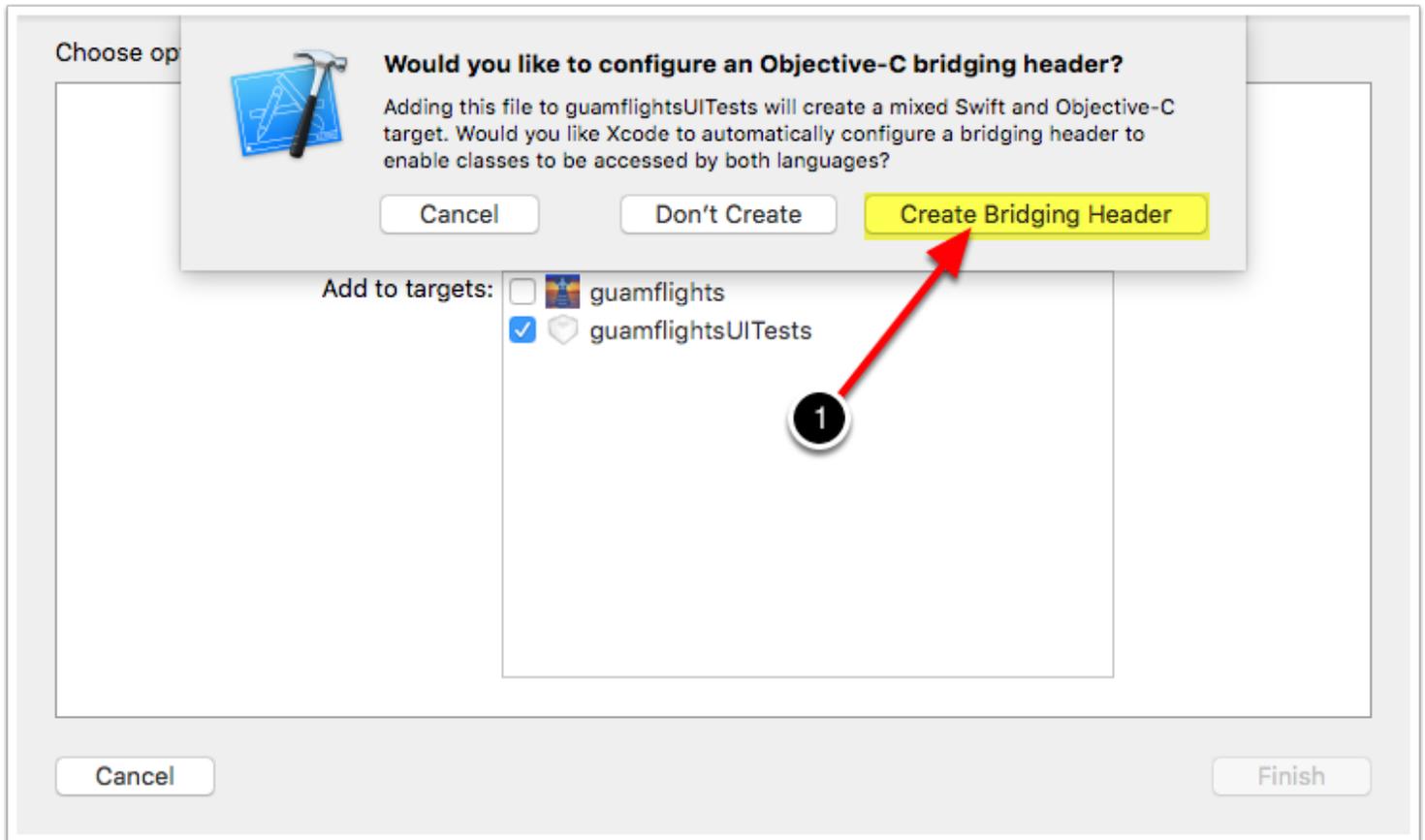
Add the 'SnapshotHelper.swift' file to your project. Be sure to deselect your 'production' app as target, and select the 'UITests' target. Where do you put it? Pretty much anywhere. I put mine in my <projectname>UITests directory, so I'd remember it was there. You're welcome to put yours in a different location.





Create a Bridging Header between ObjC and Swift.

1) If added correctly, it will prompt you to create a 'bridging header' to tie the ObjC and Swift code together, sort of. Yes, Create Bridging Header.



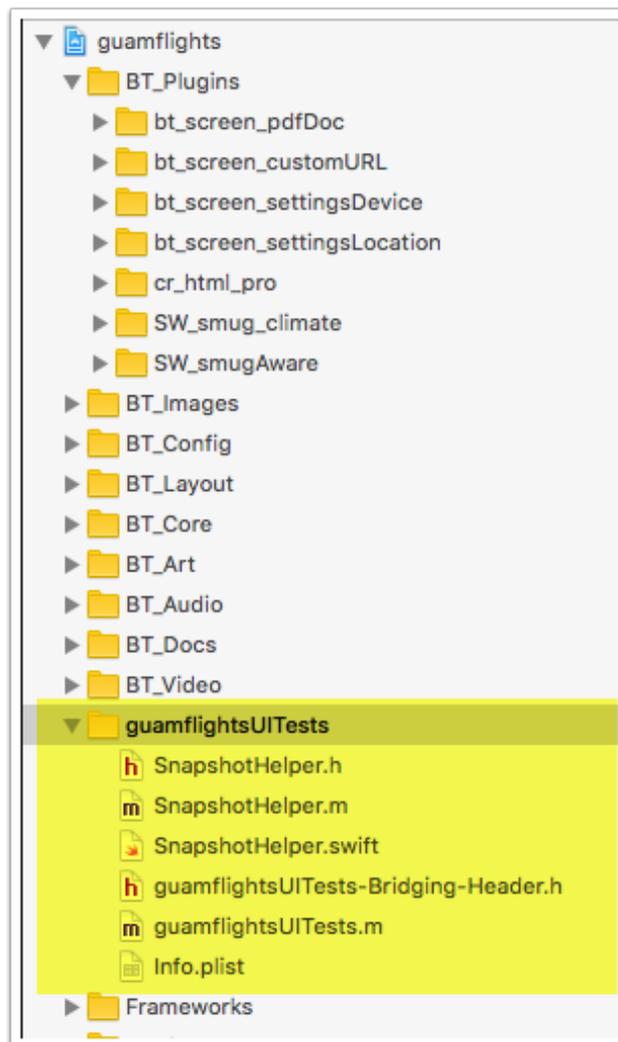


Your app project should reflect 'similar' structure.

Wait... Where did that "SnapshotHelper.h" and "SnapshotHelper.m" file come from?

It came from a problem I had; during compile, regardless of adding 'import', it could not find my swift file.

After googling the problem, I discovered that sometimes it works, sometimes it doesn't. In my case, it didn't. So I had to add the h/m file and reference it, even though the swift file is still required. You can grab a copy off my server: <https://www.marianasgps.com/public/SnapShotHelper.zip>





Setting up your 'Snapshot' environment for Buzztouch projects

Ok, now what?

Now you start to 'record' the screens that you want to have created. You do this in the test environment, NOT the traditional way.

- 1) Place your cursor in the middle of the 'setup' method, after 'launch' is called
- 2) Press the red 'record' button. It will launch the app.

The screenshot shows the Xcode IDE with the following components:

- Left Panel (Project Navigator):** Shows a project named 'guamflights' with a sub-project 'guamflightsUITests'. The file 'guamflightsUITests.m' is selected.
- Right Panel (Editor):** Displays the code for the 'setUp' method in 'guamflightsUITests.m'. The code includes comments and calls to 'launch' and 'myApp launch'. A red arrow points to the line following the 'launch' call, with a callout box that says "Put your cursor here".
- Bottom Panel (Debug Console):** Shows the output of the test run, including log messages from 'BT_fileManager', 'SW_smug_climate', and 'Banner View Loading'. A red circle highlights the 'record' button (a red dot) in the toolbar above the console, with a callout box that says "Then press the red 'record' button".



Launch Test App

The App will launch, and you'll notice a little 'tick' where you left your cursor. It will record your button clicks.

```
29 happens for each test method,  
30 [[[XCUApplication sharedInstance] initWithApplicationIdentifier:@"com.smugwimp.guamflights"];  
31 XCUApplication myApp = [[XCUApplication alloc] initWithApplicationIdentifier:@"com.smugwimp.guamflights"];  
32 // myApp launch;  
33 //  
34  
35  
36  
37  
38  
39  
40  
41  
42 // In UI tests it's important to set the initial state - such as interface orientation -  
43 // required for your tests before they run. The setUp method is a good place to do this.  
44  
45  
46 - (void)tearDown {  
47 // Put teardown code here. This method is called after the invocation of each test method in  
48 // the class.  
49 }  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

You'll notice blanks at the bottom. My 'adMob' banners go there, and I didn't want them to appear in my screenshots that I submit to Apple.



Setting up your 'Snapshot' environment for Buzztouch projects

Start your clicking

So, now just 'navigate' to the screens you want images of. Each click will echo a command that is inserted into your 'setup' method.

1. You click on a screen element
2. The command to emulate that screen click is placed in your projectNameUITests.m file

The screenshot shows an iPhone simulator on the left displaying an 'Arrivals' app. The app has a table with columns: Arrival Time, Estimated Time, Origin, Flight No., Airline, and Status. A red arrow labeled '1' points to a row in the table. On the right, the Xcode editor shows the 'setUp' method in 'guamflightsUITests.m'. A red arrow points from the table row to a line of code: `[[DKUIApplication alloc] initWithCustomURL:@"http://www.marianagps.com/flights/arrivals.php?lang=en"];`. A callout box with a '2' and the text 'Command is 'echoed' into your code' points to this line.

Arrival Time	Estimated Time	Origin	Flight No.	Airline	Status
01:00	01:36	Seoul, Korea	KE111	Korean Air	landed
01:25	01:02	Osaka Kansai	UA178	United	landed
01:45	01:13	Nagoya, JP	UA172	United	landed
01:55	01:31	Tokyo, JP - Narita	UA874	United	landed
02:00	01:30	Seoul, Korea	7C3106	Jeju Air	landed
02:35	02:38	Osaka Kansai	KE733	Korean Air	at 2:38a
04:00	03:37	Manila, Philippines	PR110	Philippine Air	at 3:37a
04:45	04:31	Hong Kong	UA118	United	at 4:31a
05:05	05:05	Saipan	UA2924	United	on time
05:20	05:06	Manila, Philippines	UA184	United	at 5:06a
05:35	05:31	Koror	UA192	United	at 5:31a
08:30	09:30	Saipan	UA2922	United	on time
10:05	10:05	Manila, Philippines	5J101	Cebu Pacific Air	on time
14:40	14:40	Tokyo, JP - Narita	DL608	Delta	on time
15:00	15:00	Seoul, Korea	LJ641	Jin Air	on time
15:10	15:10	Tokyo, JP - Narita	JL941	Japan Air Lines	on time
15:10	15:10	Saipan	UA2920	United	on time
15:30	15:30	Seoul, Korea	KE113	Korean Air	on time
15:40	15:40	Tokyo, JP - Narita	UA827	United	on time
15:45	15:45	Osaka Kansai	UA150	United	on time
15:45	15:45	Manila, Philippines	UA190	United	on time
16:05	16:05	Nagoya, JP	UA136	United	on time
16:40	16:40	Fukuoka	UA166	United	on time
17:55	17:55	Chubu	UA154	United	on time
18:05	18:05	Honolulu	UA201	United	on time
18:55	18:55	Saipan	UA2928	United	on time
21:40	21:40	Tokyo, JP - Narita	DL610	Delta	on time
22:30	22:30	Tokyo, JP - Narita	UA197	United	on time
22:40	22:40	Saipan	UA2926	United	on time



Finished? Not yet.

Now, you have the commands to replicate navigating to the screens. Now we need to add the commands to take a screen shot.

Press the red 'record' button again to halt the recording.

```
24
25 // Put setup code here. This method is called before the invocation of each test metho
    the class.
26
27 // In UI tests it is usually best to stop immediately when a failure occurs.
28 self.continueAfterFailure = NO;
29 // UI tests must launch the application that they test. Doing this in setup will make
    happens for each test method.
30 [[XCUIApplication alloc] init] launch];
31
32 XCUIApplication *myApp = [[XCUIApplication alloc] init] launch];
33 //
34 [myApp launch];
35
36 //
37
38 XCUIElementQuery *tabBarsQuery = [[XCUIApplication alloc] init].tabBars;
39 [tabBarsQuery.buttons[@"Arrivals"] tap];
40 [tabBarsQuery.buttons[@"Departures"] tap];
41 [tabBarsQuery.buttons[@"Live Tracking"] tap];
42 [tabBarsQuery.buttons[@"Weather"] tap];
43
44
45
46
47
48
49
50 //
51 // In UI tests it's important to specify the orientation
    required for your tests.
52 }
```

Look for my notes in the 'gotchas' section for some information on accessing buttons for different languages! Important!

Press to start, press to stop.



Creating an Instance of 'SnapShotHelper' and using it to create your shots at the right screens...

This is no definitive manual, but I hope it will get you started enough to where a little trial and error will get you where you need to go.

1. Create an Instance of SnapShotHelper, and place it below the app declaration, but before launch.
2. Add a 'screenshot' command to each screen you wish to create.
3. If you're going to use 'Deliver', one of the scripts in Fastlane, remember that they are uploaded in alphabetical order. If you want your screen shots to be automatically positioned the way you desire, it's best to ensure that the name of the file can be easily sorted the way you want. Adding a numerical order to the name makes it pretty simple.

What is happening: during the 'test', the script 'steps' through each line of your setup. some can be done semi-automatically. Others must be manually inserted, such as the command for your screenshots.



Setting up your 'Snapshot' environment for Buzztouch projects

```
guamflights > guamflightsUITests > guamflightsUITests.m > -setUp  
@implementation guamflightsUITests  
- (void)setUp {  
    [super setUp];  
  
    // Put setup code here. This method is called before the invocation of each test method in the class.  
  
    // In UI tests it is usually best to stop immediately when a failure occurs.  
    self.continueAfterFailure = NO;  
    // UI tests must launch the application that they test. Doing this in setup will make sure your tests don't  
    // happen for each test method.  
    [[XCUIApplication alloc] init] launch;  
  
    XCUIApplication *myApp = [[XCUIApplication alloc] init];  
    // SnapshotHelper *myHelp = [[SnapshotHelper alloc] initWithApp:myApp]; 1  
    [myApp launch];  
  
    //  
    XCUIElementQuery *tabBarsQuery = [[XCUIApplication alloc] init].tabBars;  
    [tabBarsQuery.buttons[@"Arrivals"] tap];  
    [myHelp snapshot:@"01Arrivals" waitForLoadingIndicator:YES]; 2  
    [tabBarsQuery.buttons[@"Departures"] tap];  
    [myHelp snapshot:@"02Departures" waitForLoadingIndicator:YES];  
    [tabBarsQuery.buttons[@"Live Tracking"] tap];  
    [myHelp snapshot:@"03Status" waitForLoadingIndicator:YES];  
    [tabBarsQuery.buttons[@"Weather"] tap];  
    [myHelp snapshot:@"04Weather" 3 ForLoadingIndicator:YES];  
  
    // In UI tests it is usually best to stop immediately when a failure occurs.  
    // re... the initial... entation... to do th  
}
```

Filenames. Cool, right?



Your Snap File

Your snapfile is the basic configuration that you're going to use for your project screen shots. Device types and Languages go here. If it looks a little like json, good. Remember that the last item in the list doesn't trail with a comma.

Your snapfile is created when you typed 'snapshot init' (or fastlane init) in your project. It should be located in your project root, or in your fastlane directory. edit it to your needs.

```
File Path ▾ : ~/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3
Snapfile
devices( [
  "iPhone 6",
  "iPhone 6 Plus",
  "iPhone 5",
  "iPhone 4s",
  "iPad Retina",
  "iPad Pro" ])

languages( [
  "en-US",
  "zh-HK",
  "ja-JP",
  "ko-KR",
  "ru-RU" ])

scheme "guamflights"
output_directory "./screenshots"
clear_previous_screenshots true # remove the '#' to clear all previously generated
project "./guamflights.xcodeproj"

# For more information about all available options run
# snapshot --help
```



Now try it.

Now that we think we're ready to go, open up 'terminal' and from within your project directory, type 'snapshot' (or 'fastlane snapshot', or however you configured your install to react. my fastlane 'lane' [or scriptchannel] is called 'screenshot', so my command reflects this). If you have no errors, it will start to launch each device, for each screen, for each language and capture the screenshot.

```
SmugBookPro:guamflights-iOS-BTv3.0 SmugWimps fastlane screenshot
[02:34:42]: -----
[02:34:42]: --- Step: Verifying required fastlane version ---
[02:34:42]: -----
[02:34:42]: fastlane version valid
[02:34:42]: -----
[02:34:42]: --- Step: default_platform ---
[02:34:42]: -----
[02:34:42]: Driving the lane 'ios screenshot' 🚀
[02:34:42]: -----
[02:34:42]: --- Step: snapshot ---
[02:34:42]: -----
[02:34:44]: xcrun xcodebuild -list -project './guamflights.xcodeproj'

-----
Summary for snapshot 1.12.1
-----
| devices                | ["iPhone 6", "iPhone 6 Plus", "iPhone 5", "iPhone 4s", "iPad Retina", "iPad Pro"]
| languages              | ["en-US", "de-DE", "zh-HK", "ja-JP", "ko-KR", "ru-RU"]
| scheme                | guamflights
| output_directory      | /Users/SmugWimp/Documents/App Development/Under Development/ABWo...
| clear_previous_screenshots | true
| project               | ./guamflights.xcodeproj
| launch_arguments      | [""]
| ios_version           | 9.3
| skip_open_summary     | false
| reinstall_app         | false
| erase_simulator       | false
| app_identifier        | com.ngps.gag
| buildlog_path         | ~/Library/Logs/snapshot
| clean                 | false
| number_of_retries     | 0
| stop_after_first_error | false
-----

[02:34:47]: Clearing previously generated screenshots
[02:34:47]: Building and running project - this might take some time...
[02:34:47]: Patching '/Users/SmugWimp/Library/Preferences/com.apple.iphonesimulator.plist' to scale simulator to 100%
```



Ignore the warnings... for now.

Face it. You're going to have warnings. But they're just that. 'Warnings'. 'Hey, this could be a problem eventually'... but for now, just get it to work. Go back and address the warnings later. Errors are in red. You must clear errors. Warnings? Not so much. Important, but not Critical. These are the same warnings you get in xcode when you're doing all of this by hand, so the warnings here, are the same warnings from there. Nothing new to see here.

```
guamflights-iOS-BTv3.0 — fastlane screenshot ▶ ruby — 117×31
[17:18:03]: ▶ /Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0
/BT_Config/guamflights_appDelegate.h:110:19: 'UIAlertView' is deprecated: first deprecated in iOS 9.0 - UIAlertView i
s deprecated. Use UIAlertController with a preferredStyle of UIAlertControllerStyleAlert instead
[17:18:03]: ▶ -(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex;
[17:18:03]: ▶
[17:18:03]: ▶ /Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0
/BT_Plugins/SW_smugAware/SW_smugAware.h:71:21: 'UIActionSheet' is deprecated: first deprecated in iOS 8.3 - UIActionS
heet is deprecated. Use UIAlertController with a preferredStyle of UIAlertControllerStyleActionSheet instead
[17:18:03]: ▶ -(void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:(NSInteger)buttonIndex;
[17:18:03]: ▶
[17:18:03]: ▶ Compiling guamflightsUITests.m
[17:18:12]: ▶ /Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0
/BT_Layout/BT_viewController.h:81:19: 'UIAlertView' is deprecated: first deprecated in iOS 9.0 - UIAlertView is depre
cated. Use UIAlertController with a preferredStyle of UIAlertControllerStyleAlert instead [-Wdeprecated-declarations]
[17:18:12]: ▶ -(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex;
[17:18:12]: ▶
[17:18:12]: ▶ /Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0
/BT_Config/guamflights_appDelegate.h:110:19: 'UIAlertView' is deprecated: first deprecated in iOS 9.0 - UIAlertView i
s deprecated. Use UIAlertController with a preferredStyle of UIAlertControllerStyleAlert instead [-Wdeprecated-declar
ations]
[17:18:12]: ▶ -(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex;
[17:18:12]: ▶
[17:18:12]: ▶ /Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0
/BT_Plugins/SW_smugAware/SW_smugAware.h:71:21: 'UIActionSheet' is deprecated: first deprecated in iOS 8.3 - UIActionS
heet is deprecated. Use UIAlertController with a preferredStyle of UIAlertControllerStyleActionSheet instead [-Wdepre
cated-declarations]
[17:18:12]: ▶ -(void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:(NSInteger)buttonIndex;
[17:18:12]: ▶
[17:18:12]: ▶ Linking guamflightsUITests
[17:18:13]: Running Tests: ▶ Touching guamflightsUITests.xctest
```



Setting up your 'Snapshot' environment for Buzztouch projects

Sweet!

When it's complete, it creates and displays an HTML page with all of the languages, and all of the device sizes that it created screen shots for.

Is this freakin' cool, or what!?!?!?

en-US

3.5-Inch

4-Inch

You can click on the images, and it will give you a single view slide show of your screen shots one by one, device by device, language by language. Pretty slick.

The screenshot shows a browser window with a grid of mobile app screenshots. The browser address bar shows 'file:///Users/SmugWimp/Documents/App%20Develop...'. The grid contains four columns of screenshots for different device sizes: 3.5-Inch and 4-Inch. Each column has four screenshots for different languages: en-US, es-ES, fr-FR, and de-DE. The screenshots show various app screens: Arrivals, Departures, Live Status, and Guam Weather. A red arrow points to the 'Arrivals' screenshot for the 3.5-Inch device in the en-US language. A callout box with a green background and black text explains that clicking on the images provides a single view slide show of the screen shots one by one, device by device, language by language.



Setting up your 'Snapshot' environment for Buzztouch projects

Sweet! - 2

This is what your terminal results could look like, with a well configured setup. Look into the 'gotchas' for how I figured out the way to get mine all green.

```

guamflights-iOS-BTv3.0 — -bash — 118x45
[17:40:14]: Collecting screenshots...
[17:40:14]: Found 4 screenshots...
[17:40:14]: Copying '/Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0/screenshots/en-US/iPhone4s-01Arrivals.png'...
[17:40:14]: Copying '/Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0/screenshots/en-US/iPhone4s-02Departures.png'...
[17:40:14]: Copying '/Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0/screenshots/en-US/iPhone4s-03Status.png'...
[17:40:14]: Copying '/Users/SmugWimp/Documents/App Development/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv3.0/screenshots/en-US/iPhone4s-04Weather.png'...

+-----+
| snapshot results |
+-----+
| Device | en-US |
+-----+
| iPad Pro | ♥ |
| iPad 2 | ♥ |
| iPhone 6 | ♥ |
| iPhone 6 Plus | ♥ |
| iPhone 5 | ♥ |
| iPhone 4s | ♥ |
+-----+

[17:40:14]: Generating HTML Report
[17:40:14]: Successfully created HTML file with an overview of a
elopment/Under Development/ABWonpat/Code/iOS/guamflights-iOS-BTv
[17:40:15]: Sending Crash/Success information. More information
[17:40:15]: No personal/sensitive data is sent. Only sharing the
[17:40:15]: {:fastlane_version=>1, :default_platform=>1, :snapsh
[17:40:15]: This information is used to fix failing actions and improve integrations that are often used.
[17:40:15]: You can disable this by adding `opt_out_usage` to your Fastfile

+-----+
| fastlane summary |
+-----+
| Step | Action | Time (in s) |
+-----+
| 1 | Verifying required fastlane version | 0 |
| 2 | default_platform | 0 |
| 3 | snapshot | 1376 |
+-----+

[17:40:15]: fastlane.tools just saved you 23 minutes! 🎉
SmugBookPro:guamflights-iOS-BTv3.0 SmugWimp$

```

I'm doing it language by language until I can figure out how to get the BT_config_xx.txt (language file) to load in the simulator during script execution. It changes the iOS System language, but for some reason, we're not picking it up for BT Screens and I'm not sure why just yet. When I know, I'll let you know. If you know, let me know.

News Flash! Now I know, lol!
Read the part in gotchas/localization - 4 for the absolute 'how to'.



Ok, that's how it's supposed to work. Does it really do all that?

As with everything iOS (and even more so on Android), some assembly is required. This means it's up to you to take care of some details to enable 'your particular configuration' to work with the package. Sometimes simple, sometimes not so simple.

I've already run into a few things that have me scratching my head. And as I work through them, I'll try and document it so we all don't have to freak out. In the meantime, here are a few things that immediately stood out:



Some gotchas -- Screenshot Timing; too fast?

A few of my screens load from the Web. As such, the screen shot came and went before the screen finished loading. Not good. So what I did was, I went into the 'SnapShotHelper.m' file and adjusted the wait time. I set it to 10 seconds rather than 1 second. True, it's a while, but it allowed all of the screens to load. And since it's automatic, I figured slow and steady always wins the race. You may need this, you may not. Remember, it's going to have to be redone each project. All of this stuff will.

```
guamflights > guamflightsUITests > SnapshotHelper.m > -snapshot:waitForLoadingIndicator:
26
27 - (void)snapshot:(NSString *)name waitForLoadingIndicator:(BOOL)wait {
28     if (wait) {
29         [self waitForLoadingIndicatorToDisappear];
30     }
31
32     printf("snapshot: %s\n", name.UTF8String);
33     sleep(10); // originally '1'
34     [XCUIDevice sharedDevice].orientation = UIDeviceOrientationUnknown;
35 }
36
37 - (void)waitForLoadingIndicatorToDisappear {
38     XCUIElementQuery* query = [[[_app.statusBars childrenMatchingType:XCUIElementBoundByIndex:1] childrenMatchingType:XCUIElementTypeOther];
39
40     while (query.count > 4) {
41         sleep(10); // originally '1'
42         NSLog(@"Number of Elements in Status Bar: %ld... waiting for status bar to disappear",
43               (unsigned long)query.count);
44     }
45 }
46 - (void)setLanguage {
47     NSString* prefix = [self pathPrefix];
```



Not Enough Time



Some gotchas -- Unknown Type? It's known. It just needs to be documented.

I wish I had kept better documentation. And when I do my next project, I'll keep this in mind. In my initial project setup, I had troubles with snapshot giving errors about a "unknown" type... This was things like, UIView, BT_item, etc...

All I can say is, add the 'header' file ('#import BT_item.h', etc...) to the file that is giving the error. I remember it happening in the BT_viewController, and a couple of other files. The errors were all similar, but not the same. Adding 'import' to the files fixed the error. It works in BT, because of the subclass hierarchy. But not with Snapshot, I guess. Good luck.



Some gotchas -- Localization (Languages) - 1

This stumped me for a few days. I thought I'd never figure it out. And I'm sure it 'must' be documented somewhere, because all the 'pros' kept talking about using accessibility labels for elements that were in different languages, but they never spelled out 'how' to use them. Here's how.

1) give everything you're going to 'click on' an Accessibility profile. This means

- telling xcode that the object will be an accessibility object
- giving the object an accessibility identifier (if applicable)
- giving the object an accessibility label
- giving the object an accessibility value

Do this for everything you're going to need to see, feel or touch in every language to get the screens you want to appear. Menus, tables, buttons, everything. And it seems to be hierchial (I could be wrong, but) so if it's the right navbar button, you have to setup the button, and the navbar. because it accesses the navbar first. So it can be tedious. But if it were easy, everyone would be doing it ;)

```
43 //   XCUIElement *myRefresh = [myApp.navigationBars.buttons elementBoundByIndex:0];
44 //   [myRefresh tap];
45
46 XCUIElementQuery *tabBarsQuery = myApp.tabBars;
47 [tabBarsQuery.buttons[@"button_1"] tap];
48 [tabBarsQuery.buttons[@"Arrivals"] tap];
49
50
51
```

These two lines are effectively the same.
But the first line uses 'accessibility'



Some gotchas -- Localization (Languages) - 3

Edit your 'snapfile' to include:

- erase_simulator true
- reinstall_app true
- clean true
- (by omission, 'clear_previous_screenshots' is false, which is what we want. If 'clear_previous_screenshots' is present, either remove it, or change the value to 'false'.)

What we're doing here is setting the app to use any language file as default; that will be setup in code. We also erase any simulator, so it starts off without any previous copies of the app, and install a fresh copy on the simulator. This forces the 'first' BT_config to be the language config. We must do this once per language, to ensure that we get the correct screen display for the screen shot. It's a little slower, but still better than doing it manually.

Once 'all' of the languages are complete, the 'aggregate' will be available for upload to the itunes store, just as if it had occurred in one complete run.

```
scheme "guamflights"  
output_directory "./screenshots"  
project "./guamflights.xcodeproj"  
erase_simulator true  
reinstall_app true  
clean true
```



Some gotchas -- Localization (Languages) - 4

Finally, REMOVE (or comment out) This section of your BT_loadConfigDataViewController.m file

THIS is what has been keeping my tests from being able to dynamically switch languages. If you don't use a 'configToUse' property, then get rid of this. If you do use the 'configToUse' property, then figure it out yourself. I don't need it, so my brain stops here. ;)

Now we can use a list of languages in our Snapfile, across all devices. Now *this* is efficiency!

Granted, it was a little work on the outside, but now, we know what we need to know to do what we need to do efficiently.

```
143 NSString *language = [theLang substringToIndex:2];
144 [BT_debugger showIt:self message:[NSString stringWithFormat:@"The Language is currently: %@", language]];
145
146 //build the name of the language file
147 NSString *languageFileName = [NSString stringWithFormat:@"BT_config_%@.txt", language];
148
149 //does this file exist?
150 if ([[BT_fileManager doesFileExistInBundle:languageFileName]) {
151     [self setConfigurationFileName:languageFileName];
152     [BT_debugger showIt:self message:@"The Language File Exists"];
153 } else {
154     [self setConfigurationFileName:@"BT_config.txt"];
155     [BT_debugger showIt:self message:@"The Language File Does Not Exist, and we are reverting to the default english config."];
156 }
157 [BT_debugger showIt:self message:[NSString stringWithFormat:@"The Language is config has been set to: %@", languageFileName]];
158
159 /*
160 I think this might be why it's screwing up the language files... let's find out, shall we?
161
162 //use a remembered config file if it exists. A plugin may allow a user to change the JSON config file...
163 NSString *tmpConfigFileName = [BT_strings getPrefString:@"configToUse"];
164 if ([tmpConfigFileName length] > 5){
165     [BT_debugger showIt:self theMessage:[NSString stringWithFormat:@"Will use remembered (non-standard) JSON configuration file \"%@\" if a newer version
166     the device", tmpConfigFileName]];
167     [self setConfigurationFileName:tmpConfigFileName];
168 }else{
169     [BT_debugger showIt:self theMessage:@"Will use the default JSON configuration file \"BT_config.txt\" if a newer version is not cached on the device."
170 }
171 */
172
173 //the file name of the cached JSON file after downloading (if a dataURL is used)...
174 [self setSaveAsFileName:@"cachedAppConfig.txt"];
175
176 //the file name of the cached file that holds the last modified date after the reportToCloud method runs...
177 [self setModifiedFileName:@"appModified.txt"];
178
179 //remember these values in the app's delegate so we can reference them anywhere in the app...
180 [appDelegate setConfigurationFileName:[self configurationFileName]];
181 [appDelegate setSaveAsFileName:[self saveAsFileName]];
182 [appDelegate setModifiedFileName:[self modifiedFileName]];
183
184 //begin with NO JSON data
```



Some gotchas -- Localization (Languages) - 5

Granted, sometimes it's going to drop the ball... it's a fairly complex script. But it's a simple task now to just edit out what I don't need in my Snapfile, and rerun the script to get the botched screens. Voila!

snapshot results					
Device	en-US	ko-KR	ja-JP	ru-RU	zh-TW
iPad Pro	♥	♥	♥	♥	♥
iPad 2	♥	♥	♥	♥	♥
iPhone 6	♥	♥	♥	♥	♥
iPhone 6 Plus	♥	♥	♥	♥	♥
iPhone 5	♥	♥	♥	♥	✖
iPhone 4s	♥	♥	♥	♥	♥



Some gotchas -- My script seemed to stop/is hung...?

If it's hung up, chances are there is an answer in the logs. What logs? I'm glad you asked.

```
32     printf("snapshot: %s\n", name.UTF8String);
33     sleep(10); // originally '1'
34     [XCUIDevice sharedDevice].orientation = UIDeviceOrientationUnknown;
35 }
36
37 - (void)waitForLoadingIndicatorToDisappear {
38     XCUIElementQuery* query = [[[_app.statusBars
39
40     while (query.count > 5) {
41         sleep(10); // originally '1'
42         NSLog(@"Number of Elements in Status Bar: %d", query.count);
43     }
44 }
45
46 - (void)setLanguage {
47     NSString* prefix = [self pathPrefix];
48     if (prefix == nil) {
49         -----
50     }
51 }
```

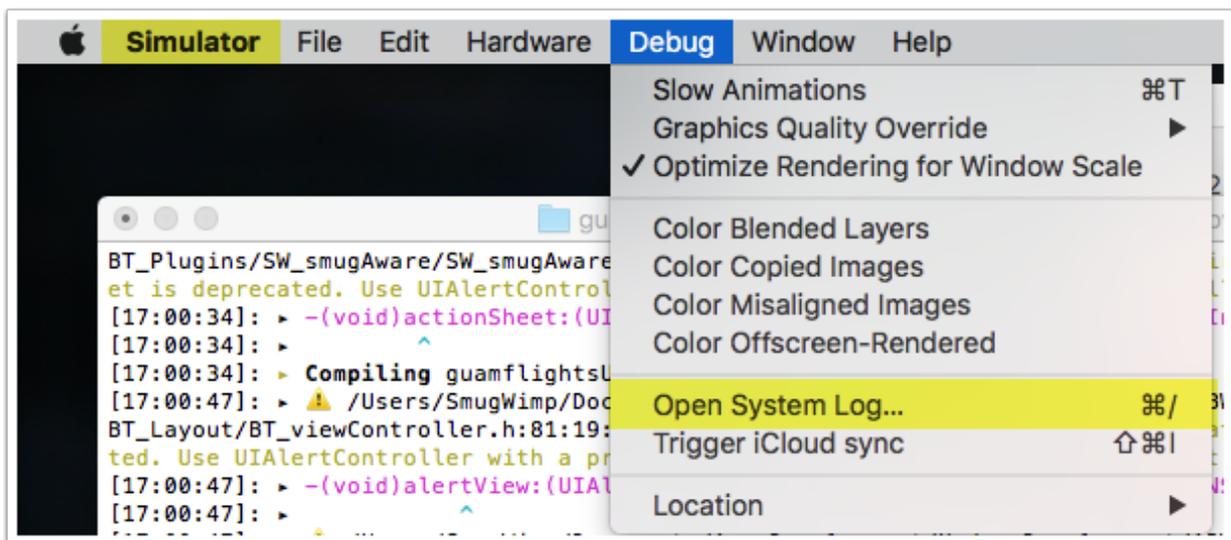
Originally set at '4', on 'iPad Pro' it wouldn't go below '5' for some reason, resulting in a loop. Changing the value doesn't seem to have hurt anything. Time will tell.



Some gotchas -- Where are the logs? - 1

Normally you would be looking in your xcode interface for the logs, debugger, whatever you want to call it.

Since we're running a script that launches xcode as a command line thing, we have to look elsewhere for our debugger output. Fortunately, we just bring the simulator to be the front active app. In the Simulator's 'Debug' menu, select 'Open System Log'.





Some gotchas -- Where are the logs? - 2

And in the system log window are the same familiar log entries you know and love. BT_debugger, NSLog, etc... they all show up here, just as if you were using xcode. Well, you are, but... anyway. That's where the logs are. Look there if you have problems. That's how I discovered that it was stuck on 5 elements in a 4 element loop. Logs, dude. Logs.

```
Apr 25 17:01:56 SmugBookPro guamflights[14439]: SW_smug_climate: Loading Weather Dictionary
Apr 25 17:01:56 SmugBookPro guamflights[14439]: BT_fileManager: File does exist in Xcode bundle: "swAniWeatherCodes.txt"
Apr 25 17:01:56 SmugBookPro guamflights[14439]: BT_fileManager: readTextFileFromBundleWithEncoding: "swAniWeatherCodes.txt" encoding: -1
Apr 25 17:01:56 SmugBookPro guamflights[14439]: SW_smug_climate: Not Deleting: weather_17.txt
Apr 25 17:01:56 SmugBookPro guamflights[14439]: SW_smug_climate: Current weather file: weather_17.txt
Apr 25 17:01:56 SmugBookPro guamflights[14439]: BT_fileManager: File does not exist in cached directory: "weather_17.txt"
Apr 25 17:01:56 SmugBookPro guamflights[14439]: SW_smug_climate: no cached version of this screens data available; downloading file
Apr 25 17:01:57 SmugBookPro guamflights[14439]: BT_fileManager: saveTextFileToCacheWithEncoding: "weather_17.txt" encodingFlag: -1
Apr 25 17:01:57 SmugBookPro guamflights[14439]: SW_smug_climate: parseWeatherData
Apr 25 17:01:57 SmugBookPro guamflights[14439]: SW_smug_climate: JustBeforeLayoutScreen
Apr 25 17:01:57 SmugBookPro guamflights[14439]: SW_smug_climate: layoutScreen
Apr 25 17:01:57 SmugBookPro guamflights[14439]: SW_smug_climate: Banner View Loading
Apr 25 17:01:57 SmugBookPro guamflights[14439]: SW_smug_climate: Banner View finished loading
Apr 25 17:02:00 SmugBookPro nsurlstoraged[14382]: realpath() returned NULL for /Users/SmugWimp/Library/Developer/CoreSimulator/Devices/CBAB1EFE-BB91-4C1D-AF63-2EA1F9EB41BB/data/Containers/Data/Application/6447147B-559A-4601-8EA8-9A329A81E76D/Library/Caches/com.mgps.gag
Apr 25 17:02:00 --- last message repeated 2 times ---
Apr 25 17:02:00 SmugBookPro SpringBoard[14342]: HW kbd: Failed to set (null) as keyboard focus
Apr 25 17:02:00 SmugBookPro assertiond[14346]: assertion failed: 15E65 13E230: assertiond + 16726 [18D9E3D0-5485-3412-8682-4BE50C825E80]: 0x1
Apr 25 17:02:00 --- last message repeated 1 time ---
Apr 25 17:02:00 SmugBookPro testmanagerd[14351]: testmanagerd exiting, idle with no test activity.
Apr 25 17:02:00 SmugBookPro com.apple.CoreSimulator.SimDevice.CBAB1EFE-BB91-4C1D-AF63-2EA1F9EB41BB.launchd_sim[14326] (UIKitApplication:com.mgps.gag[0xa32e][14439]): Service exited due to signal: Terminated: 15
```



Some gotchas -- Oops! I need to stop the script! What do I do!?!?

It's running on Python. Just hit 'control + c' (both keys at the same time) and your script should stop.



Some gotchas -- Editing your Snapfile 1

It's pretty easy, but it's not exactly laid out for understanding. Basically you use the same commands that you see 'on screen'.

If you have 'nothing' in your snapfile, the answer is assumed to be 'false'. So a lot of things are false, merely because you didn't specify them. That also means that unless you specify them as 'true', they will be 'false'. For most things, this is cool. But if you want to change something, it doesn't say exactly how to do it. It's simple; just add the command, and the value.

example: `skip_open_summary true`

```
Summary for snapshot 1.12.1
devices      ["iPhone 6", "iPhone 6 Plus", "iPhone 5", "iPhone 4s"]
languages    ["zh-TW"]
scheme       guamflights
output_directory /Users/SmugWimp/Documents/App Development/Under Development/ABWo...
project      ./guamflights.xcodeproj
launch_arguments [""]
ios_version   9.3
skip_open_summary false
clear_previous_screenshots false
reinstall_app false
erase_simulator false
app_identifier com.mgps.gag
buildlog_path ~/Library/Logs/snapshot
clean        false
number_of_retries 0
stop_after_first_error false

[11:29:07]: Building and running project - this might take some time...
[11:29:07]: Patching '/Users/SmugWimp/Library/Preferences/com.apple.iphonesimulator.plist' to scale simulator to 100%
[11:29:21]: -----
```



Some gotchas -- Editing your Snapfile 2

Not all of the changes were highlighted, to keep some space between arrows, lol! but you can see the old snapfile in other screen shots, and the changes that are made here. It's just more of the same BS, with different names. In the array sections (devices, languages) keep it 'json' looking, with a comma between values, except for the last value. If the value is a string, enclose it in quotes.

```
Summary for snapshot 1.12.1
-----
| devices          | ["iPhone 6", "iPhone 6 Plus", "iPhone 5", "iPhone 4s"]
| languages        | ["zh-TW"]
| scheme           | guamflights
| output_directory | /Users/SmugWimp/Documents/App_Development/Under_Development/ABWo...
| project          | ../guamflights.xcodeproj
| erase_simulator  | true
| reinstall_app    | true
| clean            | true
| launch_arguments | [""]
| ios_version      | 9.3
| skip_open_summary | false
| clear_previous_screenshots | false
| app_identifier   | com.ngps.gag
| buildlog_path    | ~/Library/Logs/snapshot
| number_of_retries | 0
| stop_after_first_error | false

[13:56:45]: Building and running project - this might take some time...
[13:56:45]: Patching '/Users/SmugWimp/Library/Preferences/com.apple.iphonesimulator.plist' to scale simulator to 100%
[13:56:56]: Erasing iPhone 6...
```

```
devices([
  "iPhone 6",
  "iPhone 6 Plus",
  "iPhone 5",
  "iPhone 4s"
])

languages([
  "zh-TW" ])

scheme "guamflights"
output_directory "./screenshots"
project "../guamflights.xcodeproj"
erase_simulator true
reinstall_app true
clean true

# For more information about all available
# snapshot --help
```

You can easily see the syntax needed



Some gotchas -- Editing your Snapfile 3

So, I ran the script loaded. 6 devices, 5 languages. And it went through each and every one of them, but it dropped the ball in Chinese on the iPhone 5 screenshots. So I have to redo them. I was very pleased to see we can throw in pound signs (#) to comment out a part, rather than cut/paste things around. So now the simulation is only running on Chinese, iPhone 5 to make the rest complete. Yaay.

```
File Path ▾ : ~/Documents/App Development/Under Development/ABWonpat/Code/i
Snapfile
1 devices( [
2   # "iPad Pro",
3   # "iPad 2",
4   # "iPhone 6",
5   # "iPhone 6 Plus",
6   "iPhone 5" #,
7   # "iPhone 4s"
8 ]
9
10 languages( [
11 # "en-US",
12 # "ko-KR",
13 # "ja-JP",
14 # "ru-RU",
15 # "zh-TW" ])
16
17 scheme "guamflights"
18 output_directory "./screenshots"
19 project "./guamflights.xcodeproj"
20 erase_simulator true
21 reinstall_app true
22 clean true
23
24
25 # For more information about all available options run
26 # snapshot --help
27
28
```



Some gotchas -- Sometimes you just can't sudo.

Regardless of getting successful screenshots, and html generation, the script always ended in an error. I'm not sure why. Well, I'm sure 'now', but at the time it was puzzling. In Short: Take control! Take ownership!

- Using the 'finder', navigate to the parent folder of your BT Project.
- Right click; select 'get info' (or command-I, or whatever)
- Authenticate, if you must, to change sharing permissions
- Select your own account/username
- Ensure it allows for 'Read & Write'. if not, change it.
- Using the 'gear' icon menu, select "Apply to enclosed items" "yes" to everything.
- close it up.

The next time you run the script under your own account (which is where you should be) it won't error on you when it creates the finishing files. What we just did was ensure that the project directory and all subdirectories (remember, we had to sudo to install fastlane/snapshot, so it might not have been using our credentials, right?) have 'our' account listed with read/write privileges, which means any script under our account has read/write privileges.

```
[17:02:09]: Variable Dump:
[17:02:09]: {:DEFAULT_PLATFORM=>:ios, :PLATFORM_NAME=>:ios, :LANE_NAME=>"ios screenshot"}
[17:02:09]: Tests failed - check out the log above

-----
🚨 An error occurred. Please enable crash reports using `fastlane enable_crash_reporting`.
👍 This makes resolving issues much easier and helps improve fastlane.
📄 The reports will be stored securely on getsentry.com.
🔒 More information about privacy: https://github.com/fastlane/fastlane/releases/tag/1.33.3
-----

/Library/Ruby/Gems/2.0.0/gems/fastlane-1.81.0/lib/fastlane/documentation/docs_generator.rb:39:in `write': [!] Permissi
on denied - ./fastlane/README.md (Errno::EACCES)
    from /Library/Ruby/Gems/2.0.0/gems/fastlane-1.81.0/lib/fastlane/documentation/docs_generator.rb:39:in `run'
    from /Library/Ruby/Gems/2.0.0/gems/fastlane-1.81.0/lib/fastlane/lane_manager.rb:55:in `cruise_lane'
    from /Library/Ruby/Gems/2.0.0/gems/fastlane-1.81.0/lib/fastlane/command_line_handler.rb:30:in `handle'
    from /Library/Ruby/Gems/2.0.0/gems/fastlane-1.81.0/bin/fastlane:38:in `block (2 levels) in run'
    from /Library/Ruby/Gems/2.0.0/gems/commander-4.3.5/lib/commander/command.rb:178:in `call'
    from /Library/Ruby/Gems/2.0.0/gems/commander-4.3.5/lib/commander/command.rb:178:in `call'
    from /Library/Ruby/Gems/2.0.0/gems/commander-4.3.5/lib/commander/command.rb:153:in `run'
    from /Library/Ruby/Gems/2.0.0/gems/commander-4.3.5/lib/commander/runner.rb:428:in `run_active_command'
    from /Library/Ruby/Gems/2.0.0/gems/fastlane_core-0.41.3/lib/fastlane_core/ui/fastlane_runner.rb:23:in `run!'
    from /Library/Ruby/Gems/2.0.0/gems/commander-4.3.5/lib/commander/delegates.rb:15:in `run!'
    from /Library/Ruby/Gems/2.0.0/gems/fastlane-1.81.0/bin/fastlane:168:in `run'
    from /Library/Ruby/Gems/2.0.0/gems/fastlane-1.81.0/bin/fastlane:174:in `'
    from /usr/bin/fastlane:23:in `load'
    from /usr/bin/fastlane:23:in `'
```