**Apple Watch App Documentation**

Last updated 4/28/19

**Daniel O’Brien**

* For any questions email me
* djobrie@clemson.edu

**Feature Summary:**

File reading - reads files and parses the data to run tests.

Start at time - the watch will wait for the iPhone to send a time to start. It will begin then.

File select - the watch lets you select which test you want to run with a picker, hover over a test and it will be the one used when the test begins.

Logging - the watch records times of button presses and event starts throughout the test, hold them in a ‘String’ data structure, then after completing an experiment it sends this data to the iPhone. The iPhone then converts this to a .txt file as well as displays the log on the screen. This should all happen automatically.

Acknowledge - Each event has a button to allow the user to acknowledge they saw the event. Pressing this records the time the button was pressed.

Vibration / Sound - There are vibrations and sounds in the app that occur based on certain events. I did not implement this and did not really look into it, however it should be relatively straightforward for a developer to change these in the code.

iOS app to control the watch - An iPhone app shares data an controls the watch. It also lets you build the watch app a bit easier, see the ‘Building the App on Devices’ section.

Time select / send (iOS) - To send a time for the watch to start, input a time on the watch in the input section, this is a minute and second format. E.g. inputting 3022 means the app will start at the 30 minute and 22 second mark of the hour. Press ‘Send’ to actually send the data. Sometimes it takes a few seconds for the data to be received.

**Changing Test Data**

To change test data open one of the patientsX.txt files in **Excel** or some similar editor. Then simply change the values. Opening the file with a regular word processor can cause parsing errors for some unknown reason likely due to whitespace and formatting. Note that the first and last event must be of type NoEv, this is a limitation with the code, it can definitely be changed but I don’t have enough time. The first event must also start at 0 in the time field. Additional events can be added, they will start at the time given, for example if you have ‘10’ that event will start 10 seconds after the test begins. Once the data is changed, the app must be rebuilt on the device, see ‘Building the App on Devices section’.

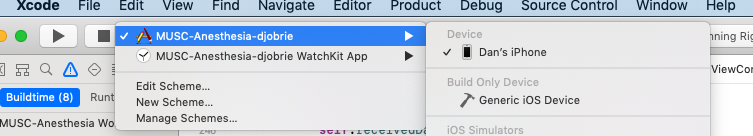
**Reading Log File**

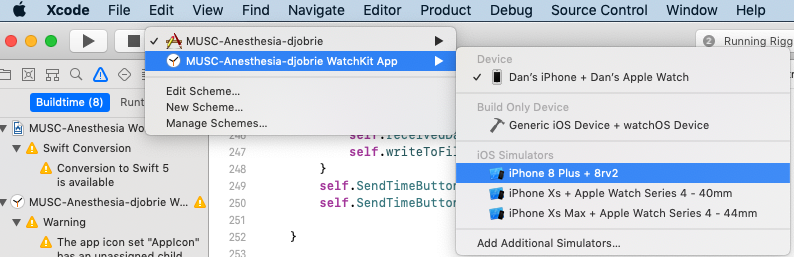
The app will create a new log file for each experiment, it creates the log file name based on the date. Multiple experiments might be logged on the same file if experiments occur quickly back to back but they will be labelled with time and the input file used. The log files are automatically created in a public folder on the iPhone which can be accessed with the ‘Files’ app and going to the ‘On My iPhone’ location.

**Building the App on Devices**

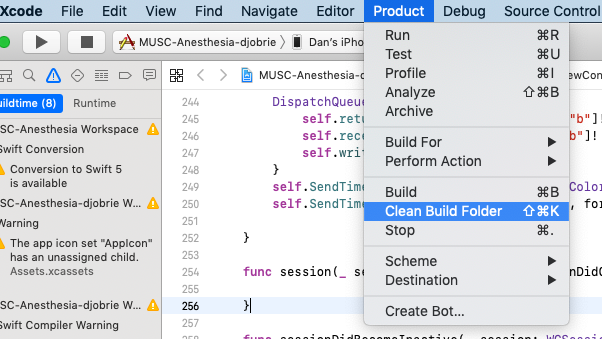
Putting the app on the watch can be a hassle. The way Apple recommends installing to the physical watch is choosing the watchkit device in the schema and then pressing build. This always takes multiple minutes and XCode often has troubles connecting to the watch through the phone. This method can work but I found a much quicker and reliable method for deploying the app:

I found the best way to do it is to choose the following scheme, which just deploys the main iOS app on the iPhone. Once XCode is done deploying the iOS app go to the ‘Watch’ app on the iPhone. At the bottom of the main ‘My Watch’ page, there should be an option to install the Watchkit portion of the app to the watch. Click this and your iPhone should install the WatchKit portion of the app to the Watch. (Pictured below)



Deploying and running on an emulator tends to be somewhat faster. To do this you can simply choose an emulator you have set up in the schema, be sure to choose the schema with the iPhone and Watchkit. You will need to press ‘Add additional simulators’ then customize an emulator to have a watch paired with it. (Pictured below)

\*Note cleaning the build folder before every single build is recommended. If you don’t the watch app will often crash for a number of reasons, XCode citing that there was not enough time to setup the Watch or something similar. (Pictured below)



If you are building the app on a physical device I recommend deleting the app from the watch and iPhone before building, this is just to make sure that you actually installed the newest version and installation is complete. Sometimes it takes so long to build the app that you might think it’s fully installed but it isn’t. This part isn’t necessary.

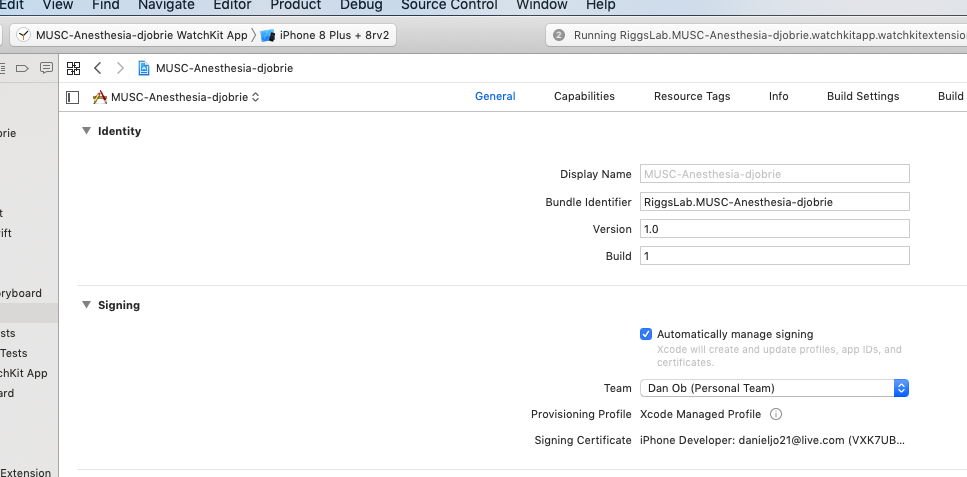
**Further Development**

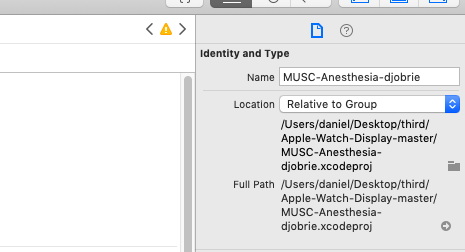
Disclaimer:

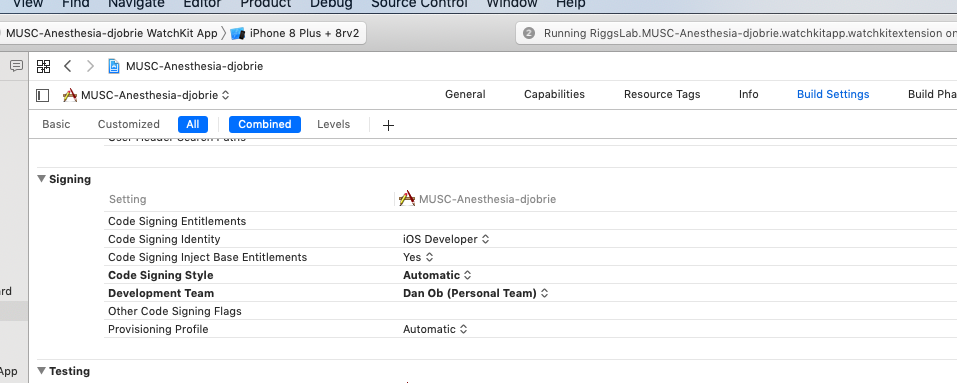
I attempted to comment my code a lot. I know that there are a lot of inefficient / nonoptimal implementations, this was my first time using Swift at all and I was building on an app that another Swift beginner made.

Setting Up:

Setting up the development environment was very difficult for me. I got a lot of errors in XCode and had to rename a lot of parts of the project. The problem is setting up the project requires a lot of changes based on your developer account and a few changes based on your file structure. You will need to change the bundle identifier and the development team based on your Apple account. XCode should point you to where you need to do these things (with errors) but it is still a hassle. Below are some things I had to change to get rid of these initial errors:







**Important files for development:**

These files are the main files you will need to change for development. I worked with other files very little aside from a few error messages.

MUSC-Anesthesia/ ViewController.swift

This is the file that controls the main iPhone screen.

MUSC-Anesthesia/ Main.storyboard

This is the file that controls the visual part of the main iPhone screen.

MUSC-Anesthesia-WatchKit App/ Interface.storyboard

This is the file that contains the the visual part of the Watch App.

MUSC-Anesthesia-WatchKit App Extension / patients1.txt

This and the other patients files contain the test data and are read.

MUSC-Anesthesia-WatchKit App Extension / startview.swift

This controls the starting page of the watch app where you pick a test file and wait to start. This page also receives messages from the iPhone.

MUSC-Anesthesia-WatchKit App Extension / interfaceController.swift

This page controls the actual tests. It reads the test data and logs data.