Article
Guillaume Rongier · Oct 15, 2020 9m read

Open Exchange

iOS, FHIR and IRIS for Health

Swift-FHIR-Iris

iOS app to export HealthKit data to InterSystems IRIS for Health (or any FHIR repository)

Table of Contents

- ToDos

Goal of this demo

The objective is to create an end-to-end demonstration of the FHIR protocol.

What I mean by end-to-end, from an information source such as an iPhone.
Collect your health data in Apple format (HealthKit), transform it into FHIR and then send it to the InterSystems IRIS for Health repository.

This information must be accessible via a web interface.

TL;DR: iPhone -> InterSystems FHIR -> Web Page.

How-To run this demo

Prerequisites

- For the client part (iOS)
  - Xcode 12
- For the server and Web app
  - Docker

Install Xcode

Not much to say here, open the AppStore, search for Xcode, Install.

Open the SwiftUi project

Swift is Apple's programming language for iOS, Mac, Apple TV and Apple Watch. It is the replacement for objective-C.

Double click on Swift-FHIR-Iris.xcodeproj

Open the simulator by a click on the top left arrow.

Configure the simulator

Go to Health

Click Steps

Add Data

Lunch the InterSystems FHIR Server

In the root folder of this git, run the following command:

```
docker-compose up -d
```

At the end of the building process you will be able to connect to the FHIR repository :

http://localhost:32783/fhir/portal/patientlist.html

This portal was made by @diashenrique.

With some modification to handle Apple's activity footsteps.

Play with the iOS app

The app will first request you to accept to share some information.

Click on authorize

Then you can test the FHIR server by clicking on 'Save and test server'

The default settings point to the docker configuration.

If succeed, you can enter your patient information.

First Name, Last Name, Birthday, Genre.

The save the patient to Fhir. A pop-up will show you your unique Fhir ID.

Consult this patient on the portal:

Go to: http://localhost:32783/fhir/portal/patientlist.html

We can see here, that there is a new patient "toto" with 0 activities.

Send her activities:

Go back to the iOS application and click on Step count

This panel summaries the step count of the week. In our case 2 entries.

Now you can send them to InterSystems IRIS FHIR by a click on send.

Consult the new activities on the portal:

We can see now that Toto has two new observation and activities.

You can event click on the chart button to display it as a chart.

How it works

iOS

Most of this demo is built on SwiftUI.

https://developer.apple.com/xcode/swiftui/

Who is the latest framework for iOS and co.

How to check authorization for health data works

It's in the SwiftFhirIrisManager class.

This class is a singleton and it will be carrying all around the application with @EnvironmentObject annotation.

More info at : https://www.hackingwithswift.com/quick-start/swiftui/how-to-use-environm...

The requestAuthorization method:

```
    // Request authorization to access HealthKit.
    func requestAuthorization() {
        // Requesting authorization.
        /// - Tag: RequestAuthorization

        let writeDataTypes: Set<HKSampleType> = dataTypesToWrite()
        let readDataTypes: Set<HKObjectType> = dataTypesToRead()

        // requset authorization
        healthStore.requestAuthorization(toShare: writeDataTypes, read: readDataTypes
) { (success, error) in
            if !success {
                // Handle the error here.
            } else {

                DispatchQueue.main.async {
                    self.authorizedHK = true
                }

            }
        }
    }
```

Where healthStore is the object of HKHealthStore().

The HKHealthStore is like the database of healthdata in iOS.

dataTypesToWrite and dataTypesToRead are the object we would like to query in the database.

The authorization need a purpose and this is done in the Info.plist xml file by adding:

```
    <key>NSHealthClinicalHealthRecordsShareUsageDescription</key>
    <string>Read data for IrisExporter</string>
    <key>NSHealthShareUsageDescription</key>
    <string>Send data to IRIS</string>
    <key>NSHealthUpdateUsageDescription</key>
    <string>Write date for IrisExporter</string>
```

How to connect a FHIR Repository

For this part I used the FHIR package from Smart-On-FHIR : https://github.com/smart-on-fhir/Swift-FHIR

The class used is the FHIROpenServer.

```
    private func test() {

        progress = true

        let url = URL(string: self.url)

        swiftIrisManager.fhirServer = FHIROpenServer(baseURL : url! , auth: nil)

        swiftIrisManager.fhirServer.getCapabilityStatement() { FHIRError in

            progress = false
            showingPopup = true

            if FHIRError == nil {
                showingSuccess = true
                textSuccess = "Connected to the fhir repository"
            } else {
                textError = FHIRError?.description ?? "Unknow error"
                showingSuccess = false
            }

            return
        }

    }
```

This create a new object fhirServer in the singleton swiftIrisManager.

Next we use the getCapabilityStatement()

If we can retrieve the capabilityStatement of the FHIR server this mean we successfully connected to the FHIR repository.

This repository is not in HTTPS, by default apple bock this kind of communication.

To allow HTTP support, the Info.plist xml file is edited like this:

```
    <key>NSAppTransportSecurity</key>
    <dict>
        <key>NSExceptionDomains</key>
        <dict>
            <key>localhost</key>
            <dict>
                <key>NSIncludesSubdomains</key>
                <true/>
                <key>NSExceptionAllowsInsecureHTTPLoads</key>
                <true/>
            </dict>
        </dict>
    </dict>
```

How to save a patient in the FHIR Repository

Basic operation by first checking if the patient already exists in the repository

```
Patient.search(["family": "\(self.lastName)"]).perform(fhirServer)
```

This search for patient with the same family name.

Here we can imaging other scenarios like with Oauth2 and JWT token to join the patientid and his token. But for this demo we keep things simple.

Next if the patient exist, we retrieve it, otherwise we create the patient :

```
    func createPatient(callback: @escaping (Patient?, Error?) -> Void) {
        // Create the new patient resource
        let patient = Patient.createPatient(given: firstName, family: lastName, dateO
fBirth: birthDay, gender: gender)

        patient?.create(fhirServer, callback: { (error) in
            callback(patient, error)
        })
    }
```

How to extract data from the HealthKit

It's done by querying the healthkit Store (HKHealthStore())

Here we are querying for footsteps.

Prepare the query with the predicate.

```
        //Last week
        let startDate = swiftFhirIrisManager.startDate
        //Now
        let endDate = swiftFhirIrisManager.endDate

        print("Collecting workouts between \(startDate) and \(endDate)")

        let predicate = HKQuery.predicateForSamples(withStart: startDate, end: endDat
e, options: HKQueryOptions.strictEndDate)
```

Then the query itself with his type of data (HKQuantityType.quantityType(forIdentifier: .stepCount)) and the predicate.

```
func queryStepCount(){

        //Last week
        let startDate = swiftFhirIrisManager.startDate
        //Now
        let endDate = swiftFhirIrisManager.endDate

        print("Collecting workouts between \(startDate) and \(endDate)")
```

```
        let predicate = HKQuery.predicateForSamples(withStart: startDate, end: endDat
e, options: HKQueryOptions.strictEndDate)

        let query = HKSampleQuery(sampleType: HKQuantityType.quantityType(forIdentifi
er: .stepCount)!, predicate: predicate, limit: HKObjectQueryNoLimit, sortDescriptors:
 nil) { (query, results, error) in

            guard let results = results as? [HKQuantitySample] else {
                  return
            }

            process(results, type: .stepCount)

        }

        healthStore.execute(query)

    }
```

How to transform HealthKit data to FHIR

For this part, we use the Microsoft package HealthKitToFHIR

https://github.com/microsoft/healthkit-to-fhir

This is a usefull package that offer factories to transform HKQuantitySample to FHIR Observation

```
     let observation = try! ObservationFactory().observation(from: item)
      let patientReference = try! Reference(json: ["reference" : "Patient/\(patientId
)"])
      observation.category = try! [CodeableConcept(json: [
          "coding": [
            [
              "system": "http://terminology.hl7.org/CodeSystem/observation-category",
              "code": "activity",
              "display": "Activity"
            ]
          ]
      ])]
      observation.subject = patientReference
      observation.status = .final
      print(observation)
      observation.create(self.fhirServer,callback: { (error) in
          if error != nil {
              completion(error)
          }
      })
```

Where item is an HKQuantitySample in our case a stepCount type.

The factory does most of the job of converting 'unit' and 'type' to FHIR codeableConcept and 'value' to FHIR valueQuantity.

The reference to the patientId is done manually by casting a json fhir reference.

```
let patientReference = try! Reference(json: ["reference" : "Patient/\(patientId)"])
```

Same is done for the category :

```
observation.category = try! [CodeableConcept(json: [
    "coding": [
      [
        "system": "http://terminology.hl7.org/CodeSystem/observation-category",
        "code": "activity",
        "display": "Activity"
      ]
    ]
])]
```

At last the observation is created in the fhir repository :

```
observation.create(self.fhirServer,callback: { (error) in
    if error != nil {
        completion(error)
    }
})
```

Backend (FHIR)

Not much to say, it's based on the fhir template form the InterSystems community :

https://openexchange.intersystems.com/package/iris-fhir-template

Frontend

It's based on Henrique works who is a nice front end for FHIR repositories made in jquery.

https://openexchange.intersystems.com/package/iris-fhir-portal

#Best Practices #FHIR #IoT #InterSystems IRIS for Health
Check the related application on InterSystems Open Exchange