iOS SDK Guide

This guide describes the steps to integrate the Collector framework into your iOS application.

Requirements

before starting the integration, ensure you have:

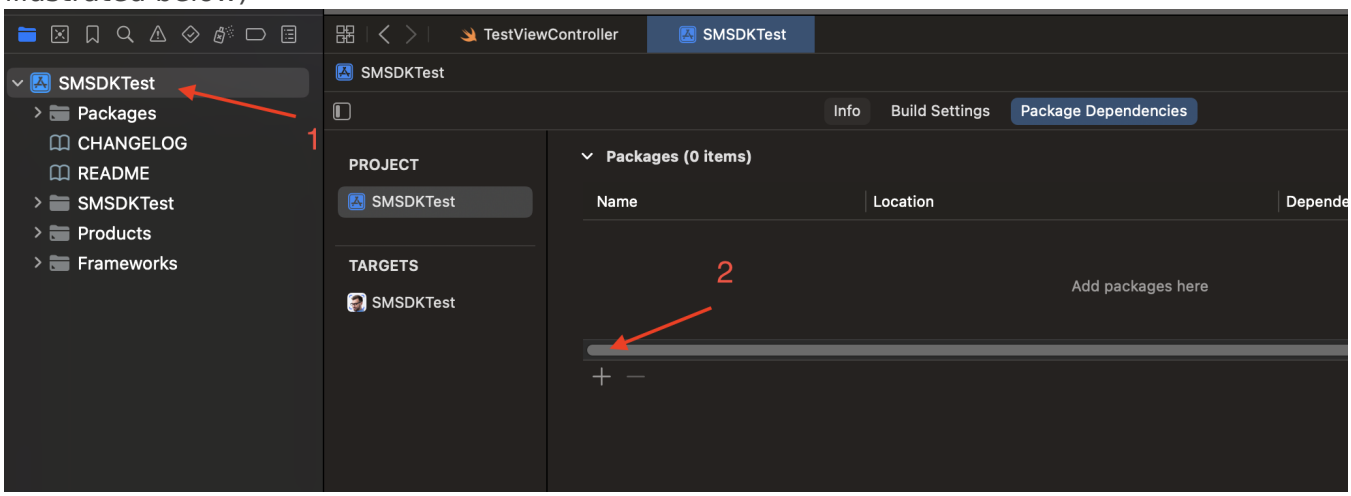
- Access to the xcframework framework

Integration Steps

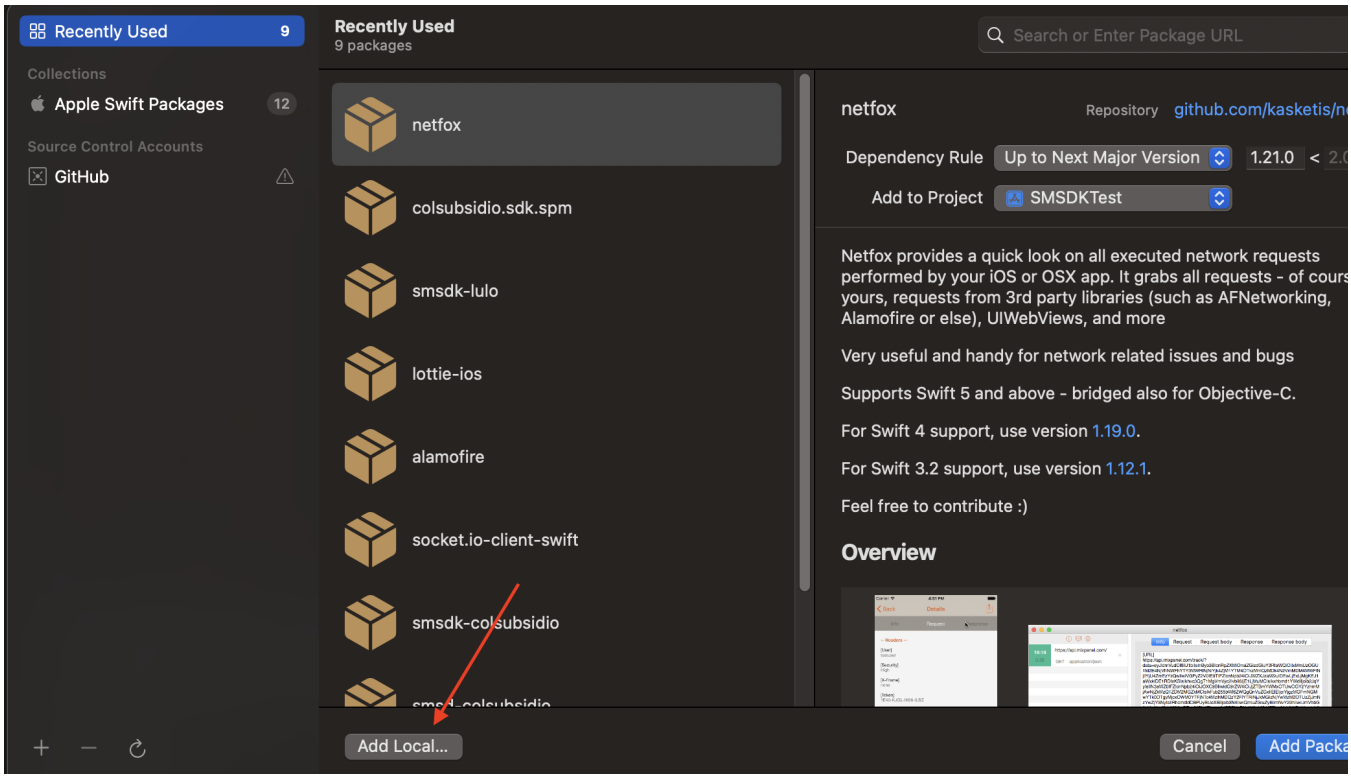
You'll recieve zip file containing the Swift Package. Complete the following steps to import the framework into your Xcode project:

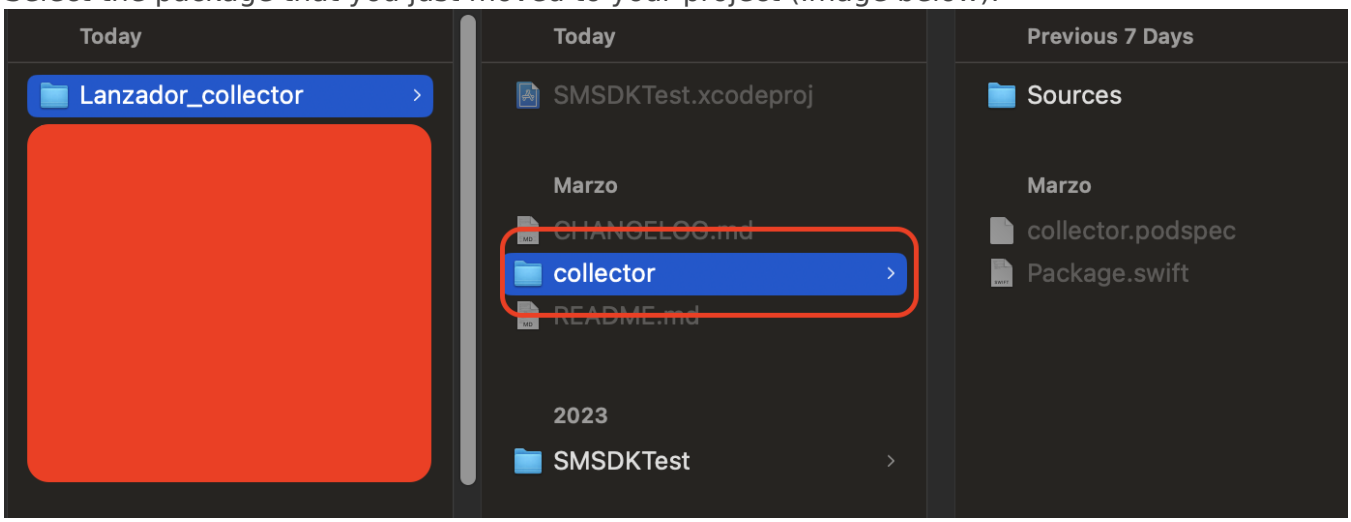
- Download the .zip which contains Collector Framework
- Unzip the contents of the zip file
- Move "Collector" folder inside your project. Place it as illustrated below.

Next steps:

- Open your project in XCode
- Select Project, Open Project Settings, Switch to Package Dependencies & Click add (+) (as illustrated below)
- 

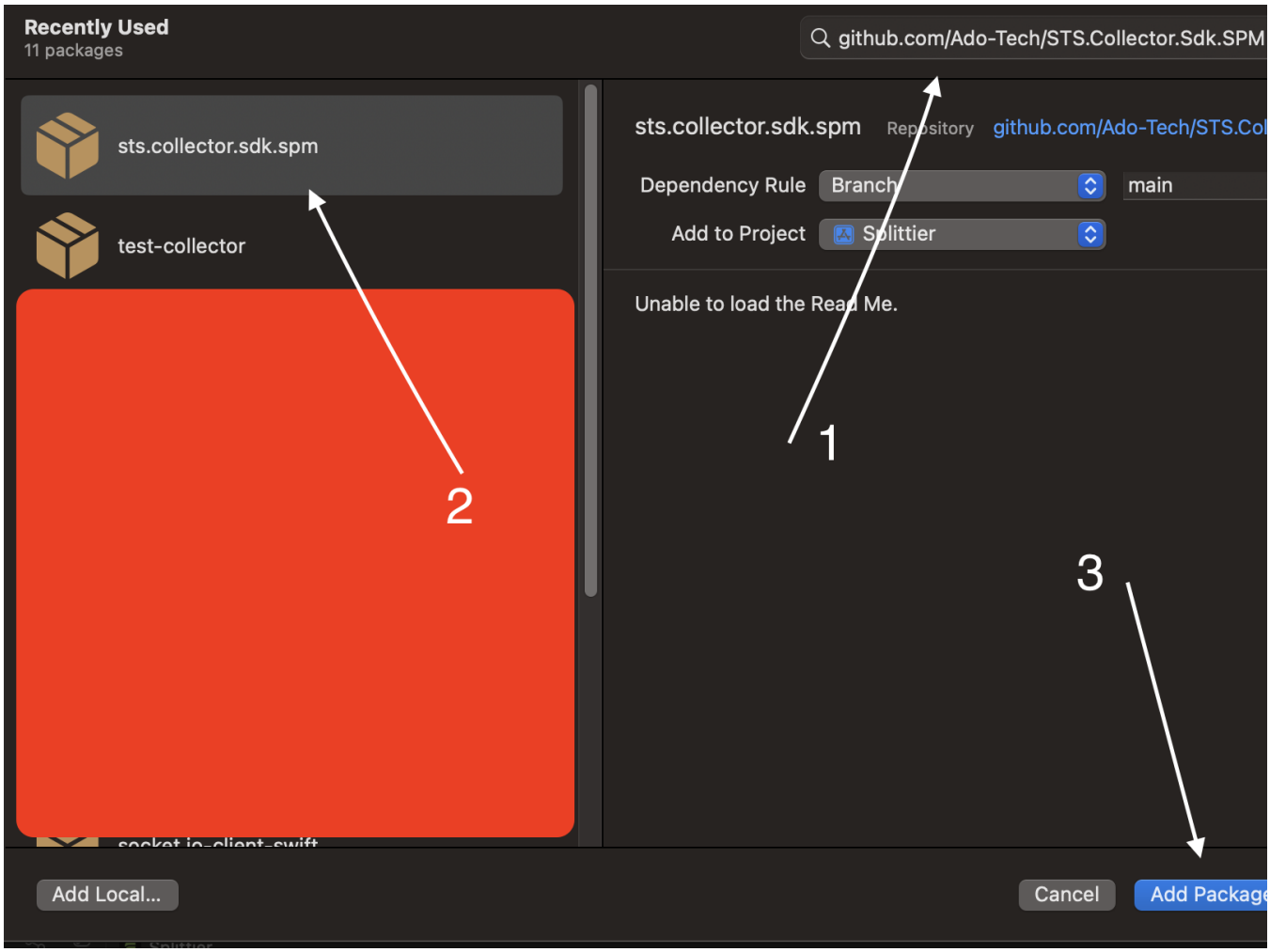
- **AGREGAR LOCAL**
- Click "Add Local" (Image below).

- 

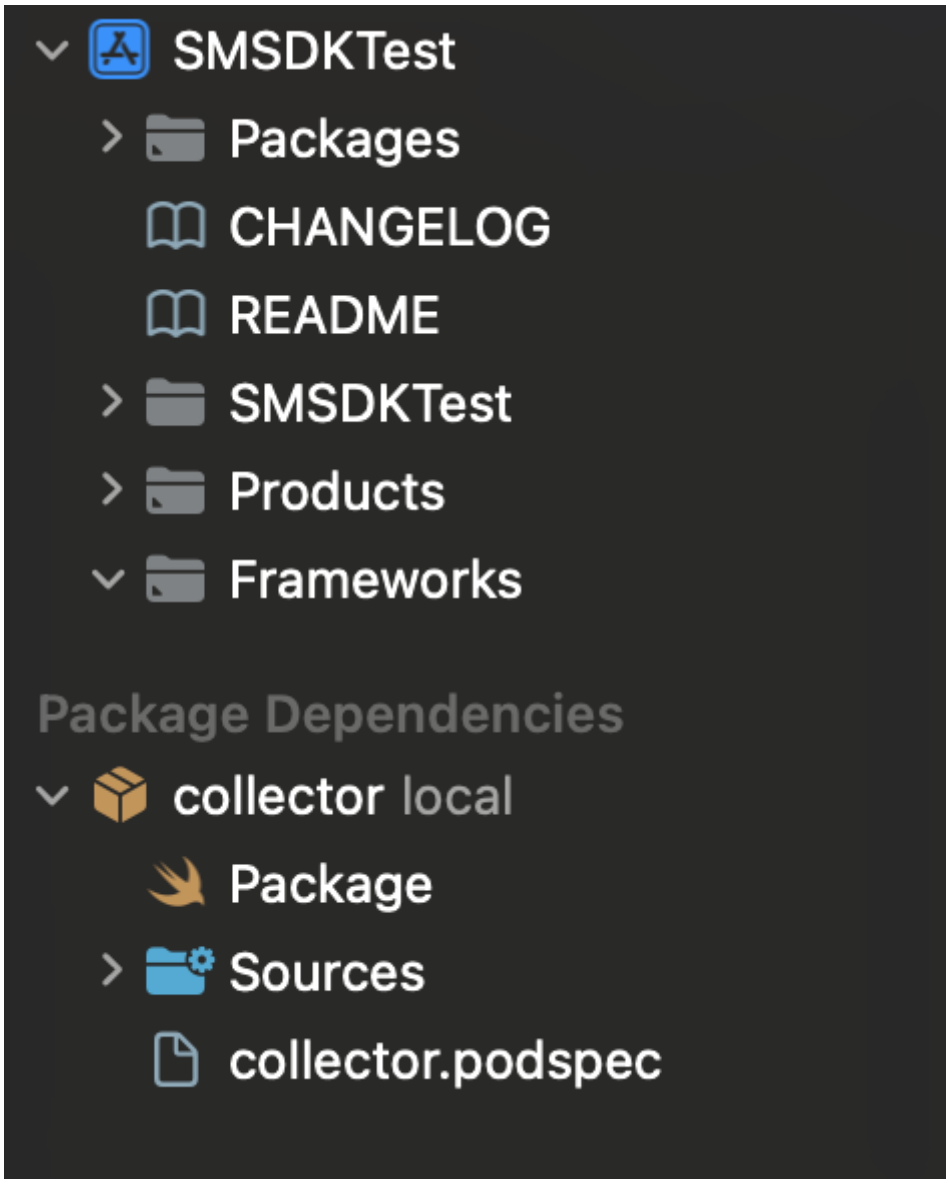
- Select the package that you just moved to your project (Image below).
- 

- ## AGREGAR DE FORMA REMOTA

- Debe tener acceso al repositorio en su computador, por eso es recomendable subirlo en un github propio

- 

- Now, Select all packages & select Target in which you wish to add those packages
- After import you should see the "collector" in the project structure. (Image below)



Linking and Embedding the SDK Framework

Ensure that the dependency is available to your application, Open Project settings, select target, switch to General Tab & make sure that dependency is added.

Application code

The general collector allows for using the camera along with some sensor collecting.

You can start listening and collecting sensor events like this:

```
import collector


Collector.shared.initialize(cid: cid, baseUrl: baseUrl, csid: csid, userID: username) {
  Collector.shared.startListeningToEvents()
  Collector.shared.collect()
}
```

`cid` = Customer Specific Identifier. This identifier will be provided by IronVest baseUrl

`csid` = Customer Session Identifier. Usually it's a unique session identifier for each user session. This is the identifier by which the session can be queried during validation or looked for in the dashboard

`userID` = Unique User Identifier.

Logging failed user logins

There is a way to report failed user logins, the backend could match the amount of failed login attempts with the userids that are being used to authenticate to correct the possibility of fraud happening from the specific device.

```
// failed login using password
Collector.shared.sendFailedLogin(uid: userId, method: "password")


// failed login using otp
Collector.shared.sendFailedLogin(uid: userId, method: "otp")


// failed login using biometric
Collecrtor.shared.sendFailedLogin(uid: userId, method: "biometric")
```

Troubleshooting

The SDK generates logs throughout its execution. The most recent logs are stored in memory and can be retrieved for problems troubleshooting. Below is an example of how these logs can be put in a variable or in the Pasteboard to be shared with someone else.

```
// Example 1: Collecting Logs in a variable
var sdkLogs = Collector.shared.getLogs()
// Example 2: Collecting logs into Pasteboard & App-User can paste it anywhere (whatsapp, mail, telegram etc)
```

```
UIPasteboard.general.string = Collector.shared.getLogs()
```

Full example implementation

Below is an example HTML structure that demonstrates how to configure the SDK in your web application. This example includes links to the SDK and assets, configuration entries and the emit context button.

```
import UIKit
import collector

class TestViewController: UIViewController, UITextFieldDelegate {

    @IBOutlet weak var context: UITextField!
    @IBOutlet weak var customerId: UITextField!
    @IBOutlet weak var userId: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        context.delegate = self
        customerId.delegate = self
        userId.delegate = self

    }

    func sendContext(context: String) {
        Collector.shared.sendContext(context: context)
    }
    func initCollector(cid: String, baseUrl: String, csid: String, userID: String) {
        Collector.shared.initialize(cid: cid, baseUrl: baseUrl, csid: csid, userID: userID) {
          Collector.shared.startListeningToEvents()
          Collector.shared.collect()
        }
    }

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
```

```
    }

    @IBAction func citizenshipIDAction(_ sender: UIButton) {
        initCollector(
            cid: "test-ado",
            baseUrl: "https://bom.stats-qa.ado-tech.com",
            csid: customerId.text ?? "",
            userID: userId.text ?? ""
        )
    }

    @IBAction func identityCardAction(_ sender: UIButton) {
        sendContext(context: context.text ?? "")
    }

}
```

Revision #12
Created 6 May 2024 16:21:15 by Admin
Updated 25 November 2024 16:06:38 by roger de avila