Hitachi Portal Integration (iOS)

Version:	0.3
File Name:	Hitachi Portal Integration (iOS)_v0.3
Confidentiality Category:	For Internal Use Only

Document Change Control

Version	Date	Author(s)	Description
0.1	15 July 2024	Chris Lee	Initial Version
0.2	30 July 2024	Chris Lee	Update Minimum Version and addEventListener
0.3	6 Aug 2024	Chris Lee	Update Notify Listener

Table of Contents

1 Introduction

1.1 Overview

This document provides a comprehensive overview of the HASPortal SDK, highlighting its primary features and benefits. The SDK includes functionalities such as Wrapping, Token Interceptor, and Publish-Subscribe to enhance the overall performance, security, and scalability of the application.

Objectives:

- **Encapsulation and Security:** The SDK ensures that the portal functionality is encapsulated within a secure environment, providing better control over lifecycle events and consistent integration across different parts of the application.
- **Enhanced Security and Streamlined Authentication:** The Token Interceptor automates the management of authentication tokens, simplifying user session handling and ensuring secure communication between the app and backend services.
- **Asynchronous Communication and Decoupling:** The Pub-Sub mechanism facilitates non-blocking, event-driven communication, promoting a modular and scalable architecture by decoupling event producers and consumers.

Key Benefits:

- **Encapsulation:** Keeps portal interactions secure and consistent, with enhanced control over initialization, usage, and termination.
- **Security and Efficiency:** Token management, reducing the risk of errors and ensuring secure network communications.
- **Modularity and Flexibility:** Supports dynamic addition and removal of event listeners, improving application responsiveness and maintainability through asynchronous event handling.

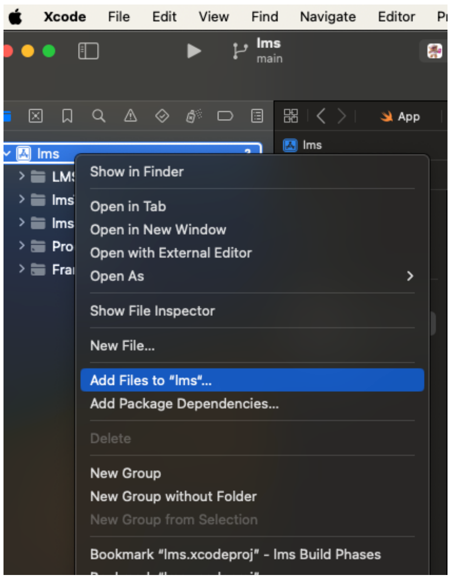
2 Specification

- iOS minimum operating system version: 12
- SDK / xcframework Size : 1.7mb

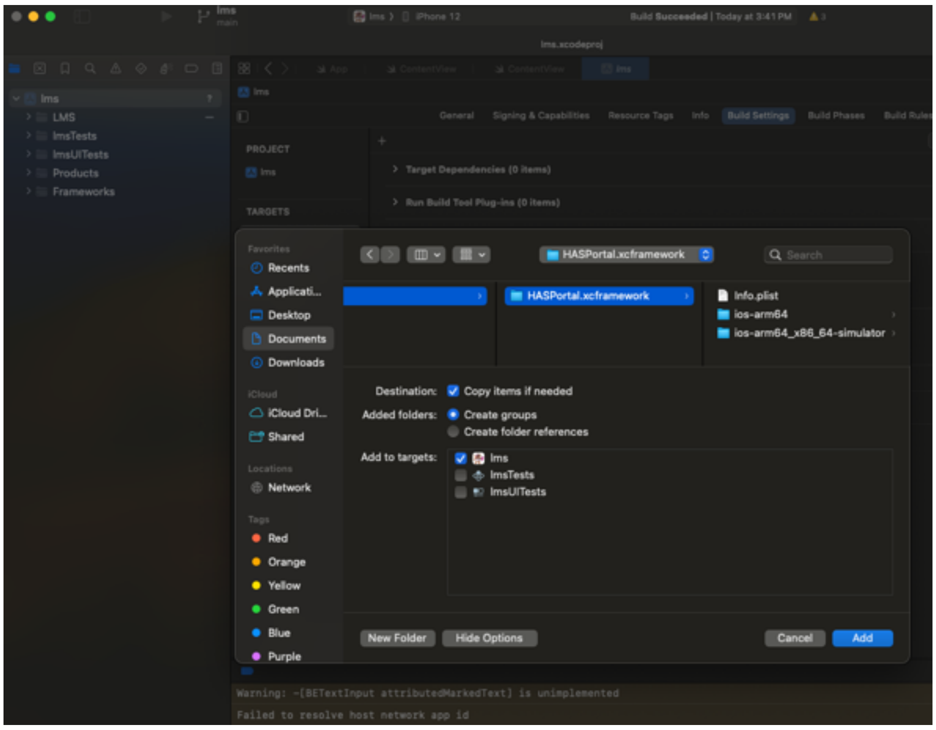
3 Installation

3.1 Add the Framework to Your Project

- Open your project in Xcode.
- In the Project Navigator, right-click on your project name and select Add Files to "YourProjectName".



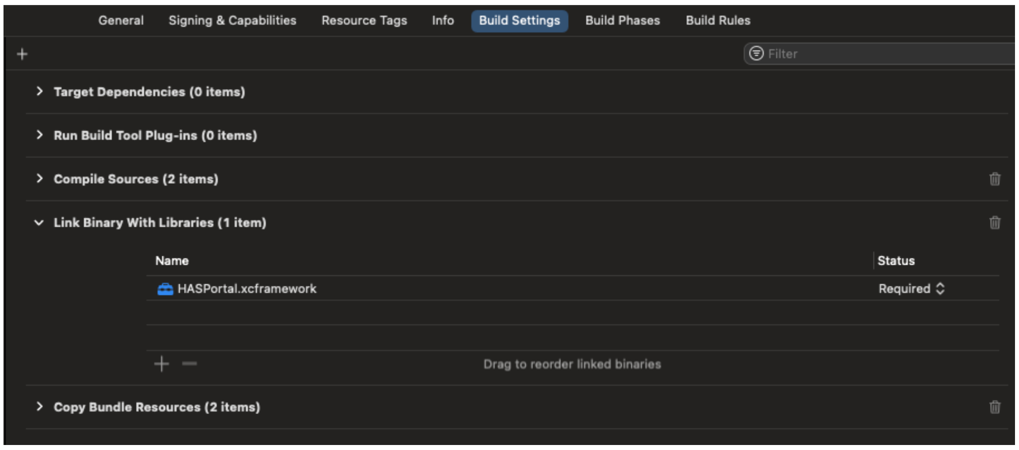
- Navigate to the location of your xcframework file and select it. Ensure that the Copy items if needed option is checked, and add it to your target(s).



Destination: [✓ Copy items if needed]
Added folders: [✓ Create groups] Create folder references
Add to targets: [✓ lms] lmsTests lmsUITests

3.2 Update Build Settings

- Select your project in the Project Navigator, then select your app target.
- Go to the Build Phases tab.
- In the Link Binary With Libraries section, ensure your xcframework is listed. If not, click the + button, find your xcframework, and add it.



Target Dependencies (0 items)
Run Build Tool Plug-ins (0 items)
Compile Sources (2 items)
Link Binary With Libraries (1 item)

Name	Status
HASPortal.xcframework	Required

Drag to reorder linked binaries
Copy Bundle Resources (2 items)

4 Example Usage

Below is an example of how to use the integrated xcframework in your iOS project:

```
import UIKit
import HASPortal
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()

        let configuration: [String: Any] = [
            "accessToken": "accessToken",
        ]
        let portal = Portal(configuration: configuration)

        portal.addEventListener(eventName: "eventsCallbacks") { data in
            print("Listener received event data: \(data)")

            // Ensure data is a dictionary
            if let eventData = data as? [String: Any] {
                // Access the 'name' element from the dictionary
                if let name = eventData["name"] as? String {
                    // Access the 'type' element from the dictionary, if it exists
                    let type = eventData["type"] as? String

                    // Handle specific events based on 'name' and 'type'
                    if name == "redeem" && type == "navigation" {
                        print("Handling redeem navigation event")
                        portal.close()
                        // Add your code to handle the redeem navigation event here
                    } else if name == "close" {
                        print("Handling close event")
                        // Add your code to handle the close event
                    } else if name == "token-expired" {
                        print("Handling Invalid Token event")
                        // Add your code to handle the token expired event here
                        let newConfiguration: [String: Any] = [
                           "accessToken": "new access token":,
                        ]
                        portal.notifyListener(eventName: "token-updated", data:
newConfiguration)
                    }
                }
            }
            return nil
        }
        let urlString = "https://example.com"
        portal.open(urlString: urlString)
    }
}
```

5 API

5.1 Portal

```
class Portal(configuration: [String: Any]) => Void
```

Parameters

Param	Type	Additional Information
configuration	[String: Any]	

Sample

```
// Configuration dictionary for the Portal instance
var configuration: [String: Any] = [
    "accessToken": "access token"
]
// Initialize the Portal with the configuration
let portal = Portal(configuration: configuration)
```

5.2 open

```
func open(urlString: String) => Void
```

Parameters

Param	Type	Additional Information
urlString	String	

Sample

```
let urlString = "https://example.com"
portal.open(urlString: urlString)
```

5.3 close

```
func close() => Void
```

Sample

```
portal.close()
```

5.4 addEventListener

```
func addEventListener(eventName: String, listener: @escaping
HASPortal.EventListener)
```

Parameters

Param	Type	Additional Information
eventName	String	
listener	@escaping HASPortal.EventListener	

Sample

```
portal.addEventListener(eventName: "eventsCallbacks") { data in
    print("Listener received event data: \(data)")

    // Ensure data is a dictionary
    if let eventData = data as? [String: Any] {
        // Access the 'name' element from the dictionary
        if let name = eventData["name"] as? String {
            // Access the 'type' element from the dictionary, if it exists
            let type = eventData["type"] as? String

            // Handle specific events based on 'name' and 'type'
            if name == "redeem" && type == "navigation" {
                print("Handling redeem navigation event")
                portal.close()
                // Add your code to handle the redeem navigation event here
            } else if name == "close" {
                print("Handling close event")
                // Add your code to handle the close event
            } else if name == "token-expired" {
                print("Handling Invalid Token event")
                // Add your code to handle the access token expired event here
               let newConfiguration: [String: Any] = [
                    "accessToken": "new access token":,
               ]
               portal.notifyListener(eventName: "token-updated", data:
newConfiguration)

            }
        }
    }
    return nil
}
```

5.5 notifyListener

```
func notifyListener(eventName: String, data: [String: Any]) => Void
```

Parameters

Param	Type	Additional Information
eventName	String	
data	[String: Any]	

Sample

```
let newConfiguration: [String: Any] = [
    "accessToken": "new access token":,
]
portal.notifyListener(eventName: "token-updated", data: newConfiguration)
```