

iOS SDK Integration Swift

Integration Overview

Integrations are done in a few simple steps:

- 1. Integrate Sardine's SDK into your application.
- Set the Ingo provided clientID into the JS SDK. This ID is unique to the SDK and varies from your participant_id.
- 3. Call IngoPayments RiskSession API to obtain a risk_session_token .
- 4. Inject the risk_session_token as the sessionKey value inside the JS SDK.
- 5. Inject the customer_ID value that will be used in the subsequent RiskScore request as the userIDHash inside the JS SDK.
- 6. Call IngoPayments RiskScore API to obtain a risk level assessment of the transaction prior to committing.

Privacy Policy Updates Before you release your application with sardine SDK, we recommend following the below steps.

Privacy Policy For the iOS 17 and App Store Connect privacy updates, please follow these steps.

Note that settings might vary from the shared steps, if your app performs certain privacy related actions outside of the > > Sardine SDK integration (e.g. tracking, user identity linking).

SDK Integration

iOS 12 not supported with the next SDK

Sardine will not support iOS 12 from the next SDK release (v1.1.6) scheduled on June 30th, 2025. This is because Apple has officially discontinued support for iOS 12 in January, 2023. Therefore, we strongly recommend updating your minimum supported iOS version to 13 or higher to benefit from our latest SDK features and improvements.

If your organization would like to continue using iOS 12 due to compelling business reasons, we recommend you continue > using the iOS SDK v1.1.5, to ensure you receive the best possible service from Sardine.

Using Swift Package

1. Contact your assigned Ingo Payments Integration Manager to provide you with your github access token. 2. Open your app in Xcode, in menu click File → Add Packages 3. Sign in to Github from swift package manager dialog by clicking on "Plus" icon at the bottom left then click "Add Source Control Account". 4. After login to Github, in Swift package manager dialog search field, enter our Github repo link "<u>https://github.com/sardine-sdk/SardineSdkSwiftPackage</u>". 5. Select the package then click: "Add Package"

Initialize SDK

Initialize SDK in AppDelegate.swift with options.

Behavior Biometrics

If you use the Behavior Biometrics feature, please ensure that your End User agrees to your privacy policy before you begin collecting their Behavior Biometric data.

Option	Description
flow	Client defined value (ex. payment, deposit, account_funding)
clientId	Ingo Payments provided client identifier

```
import MobileIntelligence
```

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
```

```
let options = OptionsBuilder()
```

```
.setClientId(with: "YOUR_CLIENT_ID")
.setSessionKey(with: "SESSION_KEY")
.setUserIdHash(with: "USER_ID")
.setEnvironment(with: Options.ENV_SANDBOX) //Options.ENV_PRODUCTION
.setSourcePlatform("Native")
.build()
MobileIntelligence(withOptions: options)
return true
```

}

Track page time

The SDK is able to track the time spent by the user or device on each page as a part of Sardine's behavior biometric feature. To enable this feature, make sure to call the trackPage(_ pageName: String) API every time the user navigates to a new page or screen. This will allow the SDK to accurately track the time spent on each view.

```
override func viewDidLoad() {
    ...
    MobileIntelligence.trackPage(pageName: "login")
}
```

Configuration enablement

To ensure our SDK can collect all the necessary signals for your project, please configure your project by following these instructions. This project-level configuration is essential for optimal fraud prevention

Enable location service

Steps:

 Each iOS project has its own Info.plist file, which contains essential bundle information and configuration settings. The location prompt message must be configured in info.plist :

<dict>

```
...
<key>NSLocationAlwaysAndWhenInUseUsageDescription</key>
<string>Text to display in the location permission prompt.</string>
<key>NSLocationWhenInUseUsageDescription</key>
<string>Text to display in the location permission prompt.</string>
</dict>
```

2. Initialize Apple's LocationManager and ensure it receives live location updates

```
class LocationService: NSObject, CLLocationManagerDelegate {
   let locationManager: CLLocationManager?
   override init() {
        locationManager = CLLocationManager()
        locationManager?.requestWhenInUseAuthorization()
        super.init()
        locationManager?.delegate = self
    }
    func locationManager(_ manager: CLLocationManager, didChangeAuthorization status:
CLAuthorizationStatus) {
        if status == .authorizedWhenInUse {
            if CLLocationManager.isMonitoringAvailable(for: CLBeaconRegion.self) {
                if CLLocationManager.isRangingAvailable() {
                    let latitude = manager.location?.coordinate.latitude
                    let longitude = manager.location?.coordinate.longitude
                    print("latitude: \(latitude), longitude: \(longitude)")
                }
           }
        }
    }
}
```

Enable wi-fi information

Wi-Fi BSSID and SSID serve as essential secondary data for determining a device's true location. If GPS data is unavailable and the remote IP cannot be collected, these

signals may be used as fallback sources for location detection.

Steps:

- 1. Enable location services in your project first by following Enable location service section above (MUST)
- 2. Wi-fi information access should be given:
 - Go to the target scheme.
 - Click Signing & Capabilities.
 - Click + Capabilities button on the top left.
 - Add Access Wi-fi information capability.
- 3. (Optional) If you manage code signing manually, you will need to update your provisioning profile to include the Access Wi-Fi Information capability.

Enable remote control software detection

On iOS, Apple restricts access to data outside of an application, preventing the iOS SDK from reading the state of other apps in real-time. However, under certain conditions, it can detect whether a specific application is installed.

Steps:

 The project must include each app's scheme name in the LSApplicationQueriesSchemes section of its info.plist file:

```
<dict>
...
<key>LSApplicationQueriesSchemes</key>
<array>
<string>anydesk</string> // Anydesk
<string>zohoassistcustomer</string> // Zoho Assist
<string>teamviewerTVCv1</string> // TeamViewer
<string>iperius_remote_mac</string> // Iperius Remote Desktop
<string>cydia</string> // Cydia
</array>
...
</dict>
```

Enable iCloud record ID

v1.1.6 or above

An iCloud record ID remains consistent across a user's devices when they're logged in with the same Apple account. Within iCloud, record IDs are unique to each application, primarily due to the app-specific containers they reside in.

Steps:

- 1. Enable iCloud in your project
 - Go to the target scheme.
 - Click Signing & Capabilities.
 - Click + Capabilities button on the top left.
 - Add iCloud capability.
- 2. Once you've added the iCloud capability, select the CloudKit service, and then choose the container associated with your application. See more details on

Enabling CloudKit in Your App.

3. Ensure that you set the flag within the SDK's configuration options

```
options = OptionsBuilder()
               .setClientId(with: kClientId)
               .setSessionKey(with: kSessionKey)
               .setUserIdHash(with: UUID().uuidString)
               ...
               .isCloudEntitlementSet(isEntitled: true) // This flag must be set to
true for the SDK to operate correctly.
               .build()
```

MobileIntelligence(withOptions: options)

4. (Optional) If you manage code signing manually, you will need to update your provisioning profile to include the iCLoud capability.

DO NOT set this value to true if CloudKit entitlements have not been properly configured for your application, otherwise the SDK will crash.

Enable Biometric authentication detection (FaceID, TouchID)

v1.1.6 or above

Sardine SDK detects the type and success of any biometric authentication that occurs while the application is running.

Steps:

1. The project must include Face ID key, NSFaceIDUsageDescription, in info.plist file:

<dict>

```
<key> NSFaceIDUsageDescription</key>
<string>Face ID used to verify who you are.</string>
```

</dict>

2. Platform-Specific API Usage

In Native development: use SardineLAContext instead of LAContext.

```
@IBAction func BiometricButtonClicked(_ sender: Any) {
    let context = SardineLAContext() <--- use instead of LAContext
    if context.canEvaluatePolicy(.deviceOwnerAuthentication, error: nil) {
        context.evaluatePolicy(.deviceOwnerAuthentication, localizedReason: "Please
    verify your identity") { success, error in
        if success {
            print("Biometric authentication is successfully verifed")</pre>
```

```
} else {
    print("Biometric authentication is not successfully verifed")
    }
    }
} else {
    print("biometric not supported.")
}
```

In React native development: use the authenticateWithDeviceBiometric API.

```
_authenticateWithBiometric = async () => {
    const localizedMessage = 'Please authenticate to login';
    const result = await Sardinesdk.authenticateWithDeviceBiometric(localizedMessage);
   console.log(result);
 };
. . . . . .
<TouchableOpacity
    activeOpacity={0.7}
    style={[
        styles.btnLogin,
        { backgroundColor: isLoading ? 'lightgrey' : themeColor },
    ]}
    onPress={this._authenticateWithBiometric}
>
. . . . . .
</TouchableOpacity>
```

In Flutter development: use the authenticateWithDeviceBiometric API.

Map<String, dynamic> payload = {

"localizedReason":"A concised reason for authentication" // Application reason for authentication. This string must be provided in correct localization and should be short and clear,

};

platform.invokeMethod("authenticateWithDeviceBiometric", payload);