iOS

This document contains the Programmable Wallets SDK reference for iOS development.Suggest Edits

The User Wallet provides an App SDK in web, iOS, and Android for the customer to integrate. The App SDK secures the process when users input their secret data, whether as PIN or security answer, and the SDK app encrypts the request body by the secret key.

Due to the security requirement for Circle to control the end-to-end process, the SDK also exposes functionality for the customer to customize the description and layout:

- UI Title and Subtitle Customization:
- Modify the title and subtitle to reflect your brand identity or provide specific instructions.
- Custom PIN Code Input Layout:
- Adjust the layout and styling of the PIN code input field to align with your application's design guidelines.
- Question List Configuration:
- Set the list of security questions displayed in the User Wallet UI, allowing users to choose from a predefined set.
- SDK Initialization:
- Initialize the Web SDK by setting the endpoint server, ensuring seamless communication between your application and our services.
- Predefined Error Messages:
- Customize the error messages displayed to users, providing a more personalized experience and guidance.
- ChallengeID Acceptance and Operation Retrieval:
- Easily accept the challengeID and retrieve any relevant operations within the SDK.

Tip:

To use the SDK in the most flexible way, combine this SDK reference with the iOS SDK UI Customization API article.

SDK Installation

Requirements

The iOS SDK supports:

1. iOS 13.0+
2. macOS 12.5+
3. Xcode 14.1+

*Earlier versions are not supported.

CocoaPods (suggested)

CocoaPods is a dependency manager for Cocoa projects. You can install it with the following command:

Terminal brew install cocoapods How to install Homebrew in MacOSLink To integrate into your Xcode project using CocoaPods, specify it in your Podfile :

Podfile source 'https://github.com/CocoaPods/Specs.git' source 'https://github.com/circlefin/w3s-ios-sdk-podspecs.git'

target '' do
pod 'CircleProgrammableWalletSDK' end Then, run the following command:

Terminal pod install

Manually

Download the iOS SDKhere .

iOS SDK

WalletSdk

Public Methods /// Singleton

public static let shared = WalletSdk() /// Set SDK configuration /// - Parameter configuration: Configure the Circle endpoint for SDK to connect and other settings. /// - Throws: An error is thrown while the configuration is invalid.

public func setConfiguration(_ configuration: Configuration) /// Execute the operations by challengeId and call the completion after sending the request to the Circle endpoint.

public func execute(userToken: String, secretKey: String, challengeIds: [String], completion: ExecuteCompletion? = nil) /// Set SDK LayoutProvider /// - Parameter provider: WalletSdkLayoutProvider

public func setLayoutProvider(_ provider: WalletSdkLayoutProvider) /// Set messenger for custom error messages /// - Parameter messenger: ErrorMessager

public func setErrorMessenger(_ messenger: ErrorMessenger) /// Set SDK delegate /// - Parameter delegate: WalletSdkDelegate

public func setDelegate(_ delegate: WalletSdkDelegate) /// Set up biometrics to protect PIN code /// - Parameters: /// - userToken: User token /// - encryptionKey: Encryption key /// - completion: The completion handler that will be called when biometrics setup is complete.

public func setBiometricsPin(userToken: String, encryptionKey: String, completion: ExecuteCompletion? = nil)

WalletSdk.Configuration

public struct Configuration

Fields String endPoint

Init with Circle endpoint. SDK will connect to this endpoint. String appId

AppID from Circle Web3 Services Console SettingsManagement settings

Configure other settings and features

SettingsManagement

Swift public struct SettingsManagement {

/// Enable biometrics to protect PIN code
let enableBiometricsPin: Bool

public init(enableBiometricsPin: Bool = false)

}

Notice:

Important : In any app that uses biometrics, include theNSFaceIDUsageDescription key in the app's Info.plist file. Without this key, the system won't allow your app to use biometrics (Face ID).

ExecuteCompletion

public typealias ExecuteCompletion = ((ExecuteCompletionStruct) -> Void)

ExecuteCompletionStruct

Swift public struct ExecuteCompletionStruct { public let challenges: [String] public let result: Result public let onErrorController: UINavigationController? public let onWarning: ExecuteWarning? } * When receiving an error or warning, we will not close the current page by default. * You will get the onErrorController in the callback, and then you can decide to dismiss/push the page by your error-handling strategy.

ExecuteResult

Swift public struct ExecuteResult {

/// The type of the operation that the challenge represents. public let resultType: ChallengeType

/// The execute result public let status: ChallengeStatus

/// The data of the execution. (optional) public private(set) var data: ExecuteResultData? = nil }

ChallengeType

Swift public enum ChallengeType: String, Decodable { case SET_PIN case CHANGE_PIN case RESTORE_PIN case SET_SECURITY_QUESTIONS case SET_BIOMETRICS_PIN case CREATE_WALLET case CREATE_TRANSACTION case ACCELERATE_TRANSACTION case CANCEL_TRANSACTION case CONTRACT_EXECUTION case SIGN_MESSAGE case SIGN_TYPEDDATA case UNKNOWN }

ChallengeStatus

Swift public enum ChallengeStatus: String, Decodable { case PENDING case IN_PROGRESS case COMPLETE case FAILED case EXPIRED }

ExecuteResultData

public struct ExecuteResultData: Decodable {

/// The signature for SIGN_MESSAGE and SIGN_TYPEDDATA (optional)
public let signature: String?

}

ExecuteWarning

Swift public struct ExecuteWarning {

/// The warning type of the operation (the challenge execution is success)
public let warningType: WarningType

/// Description of the warning type
public var warningString: String

}

WarningType

Swift public enum WarningType: Int {

// Biometrics

case deviceNotSupportBiometrics = 155709
case biometricsKeyPermanentlyInvalidated = 155710 // For Android
case biometricsUserSkip = 155711
case biometricsUserDisableForPin = 155712
case biometricsUserLockout = 155713 // For Android
case biometricsUserLockoutPermanent = 155714
case biometricsUserNotAllowPermission = 155715
case biometricsInternalError = 155716

}

ApiError

public struct ApiError

Fields ApiError.ErrorCode errorCode String errorString /// Error string from the SDK String displayString /// Error string for UI display ApiError.ErrorCode

public enum

Enum Cases case unknown = -1
case success = 0
case apiParameterMissing = 1
case apiParameterInvalid = 2
case forbidden = 3
case unauthorized = 4
case retry = 9

case walletIdNotFound = 156001
case tokenIdNotFound = 156002
case transactionIdNotFound = 156003
case walletSetIdNotFound = 156004

case notEnoughFounds = 155201
case notEnoughBalance = 155202
case exceedWithdrawLimit = 155203
case minimumFundsRequired = 155204
case invalidTransactionFee = 155205
case rejectedOnAmlScreening = 155206
case tagRequired = 155207
... WalletSdkLayoutProvider

SetWalletSdkLayoutProvider for layout customization.

Swift public protocol WalletSdkLayoutProvider: AnyObject { /// Set security question list /// /// - Returns: [SecurityQuestion] func securityQuestions() -> [SecurityQuestion]

/// Set security question required count (default is 2), used in SecurityQuestionsViewController /// /// - Returns: Int func securityQuestionsRequiredCount() -> Int

/// Set security confirm item list /// /// - Returns: [SecurityConfirmItem] func securityConfirmItems() -> [SecurityConfirmItem]

/// Set local and remote images /// /// - Returns: ImageStore func imageStore() -> ImageStore

/// Set theme font programmatically /// /// - Returns: ThemeFont func themeFont() -> ThemeConfig.ThemeFont?

/// Set the date format for display date strings. (Default is "yyyy-MM-dd") /// Reference: https://stackoverflow.com/a/52297497 /// /// - Returns: Date format string func displayDateFormat() -> String } SecurityQuestion

public struct SecurityQuestion

Fields String title

The question string SecurityQuestion.InputType inputType

The input type of the question. SecurityQuestion.InputType

public enum InputType

Enum Cases text
datePicker ... SecurityConfirmItem

public struct SecurityConfirmItem

Fields UIImage? Image The drawable resource ID for the introduction item. URL? url

The remote image url String Text

The Text for the introduction item. ImageStore

public struct ImageStore

Fields [Img: UIImage] local The images from local [Img: URL] remote

The images from remote ImageStore.Img

public enum Img

Enum Cases naviClose naviBack

securityIntroMain selectCheckMark dropdownArrow errorInfo securityConfirmMain

biometricsAllowMain

showPin hidePin

WalletSdkDelegate

Methods for managing CircleProgrammableWalletSDK process and customizing controllers.

Swift public protocol WalletSdkDelegate: AnyObject { /// Tells the delegate that the SDK is about to be present the controller. /// You can customize the layout as you wish dynamically. /// /// - Parameter controller: The UIViewController to be presented func walletSdk(willPresentController controller: UIViewController)

/// Tells the delegate that the forget PIN button was selected in the controller /// /// - Parameter controller: Current controller /// - Parameter onSelect: Void func walletSdk(controller: UIViewController, onForgetPINButtonSelected onSelect: Void) }

ErrorMessenger

Define the customized error description. All error codes are listed in ApiError.ErrorCode.

Swift public protocol ErrorMessenger { func getErrorString(_ code: ApiError.ErrorCode) -> String? }

Sample Code

Set endpoint

Swift let configuration = WalletSdk.Configuration(endPoint: endPoint, appId: appId)

do { try WalletSdk.shared.setConfiguration(configuration) } catch { // error handling } Set endpoint and enable biometrics

Swift let settings: WalletSdk.SettingsManagement = .init(enableBiometricsPin: true) let configuration = WalletSdk.Configuration(endPoint: endPoint, appId: _appId, settingsManagement: settings)

do { try WalletSdk.shared.setConfiguration(configuration) } catch { // error handling } Execute

Swift Wallets.shared.execute(userToken: "USERTOKEN", encryptionKey: "ENCRYPTIONKEY", challengeIds: ["challengeId_1", "challengeId_2"]) Set Biometrics Pin

Swift WalletSdk.shared.setBiometricsPin(userToken: "USERTOKEN", encryptionKey: "ENCRYPTIONKEY") WalletSdkLayoutProvider

Swift class SomeClass: WalletSdkLayoutProvider {

```
func securityQuestions() -> [SecurityQuestion] {
    return [...]
}

func securityConfirmItems() -> [SecurityConfirmItem] {
    return [...]
}

func imageStore() -> ImageStore {
    let local: [ImageStore.Img: UIImage] =  [...]
    let remote: [ImageStore.Img : URL] = [...]
    return ImageStore(local: local, remote: remote)
}

func displayDateFormat() -> String {
    return "dd/MM/yyyy"
}
```

}

WalletSdk.shared.setLayoutProvider(SomeClass()) Set Error Messages

Swift class CustomErrorMessenger: ErrorMessenger {

```
func getErrorString(_ code: ApiError.ErrorCode) -> String? {
    switch code {
    case .pinCodeNotMatched:
        return "The code you entered is not the same as the first one."

    case .insecurePinCode:
        return "Your PIN can’t have repeating or consecutive numbers."

    case .hintsMatchAnswers:
        return "Your hint can’t be the same as the answer."

    default:
        return nil
    }
}
```

}

WalletSdk.shared.setErrorMessenger(CustomErrorMessenger()) Confirm the delegate for customizing the UI

Swift class SomeClass: WalletSdkDelegate { func walletSdk(willPresentController controller: UIViewController) {

// You can customize the controller's UI here

```
    if let vc = controller as? SecurityIntrosViewController {
        vc.introImageView.sd_setImage(with: URL(string: "https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png"), placeholderImage: UIImage(named: "imangeName"))
        vc.introLink = "https://www.google.com"
        vc.introDescLabel.text = "Test Desc here"
    }
}
```

} See our sample app guideshere . Updated4 months ago

What’s Next

-
  -
    - WalletSdk
-
  -
    - Sample Code