Mobile SDK IOS Guide

[iOS] Mobile SDK Intro

Cardpay Unlimint mobile SDK for iOS (UnlimintSdk) helps you to:

Embed card data forms in the merchant's mobile app and securely collect and transmit the user's card data for:

- Card tokenization (without a payment) on the Unlimint side
- Making a mobile payment
- Making a payment with card token

Installation

UnlimintSDK is available through CocoaPods

To install UnlimintSDK with CocoaPods, add the following lines to your `Podfile`.

```
source 'https://github.com/cardpay/ios-sdk-podspec.git'

platform :ios, '11.0'
use_frameworks!

pod 'UnlimintSDK'
```

Then run pod install command. For details of the installation and usage of CocoaPods, visit its official website.

Basic Usage

Environment

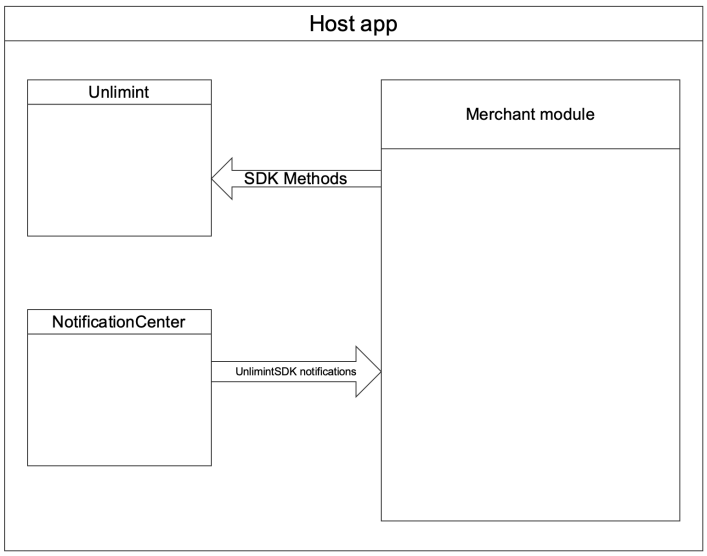
```
Unlimint.shared.environment = .sandbox

public enum Environments {

    case sandbox

    case prod

}
```

UI customization

Full info here. [iOS] Mobile SDK Customization

```
'Unlimint.shared.theme'

public struct Theme {

    public var navigationStyle: NavigationBarStyle

    public var mainButtonStyle: MainButtonStyle

    public var viewControllerStyle: ViewControllerStyle

    public init(navigationStyle: NavigationBarStyle = .init(bar: .largeNavBar,
                                                            statusBarStyle: .default,
                                                            navigationBarColor: .transparentDark,
                                                            tintColor: .clear),
                                mainButtonStyle: MainButtonStyle = .init(cornerRadius: 2,
                                                                         titleColor: (UIConstants.Colors.primaryBlack,
                                                                                      UIConstants.Colors.primaryGray),
                                                                         backgroundColor: (UIConstants.Colors.primaryGreen,
                                                                                           UIConstants.Colors.primaryWhite)),
                                viewControllerStyle: ViewControllerStyle = .init(backgroundColor: .white))
}
```

[iOS] Mobile SDK API description



Methods

Binding

```
     /**
     Use this method for binding card.
     */
    public func bindNewCardFor(for mobileToken: String, with data: UnlimintSDK.BindingMethodData,
presentationStyle style: UnlimintSDK.PresentationStyle)
```

Payment

```
    /**
     Use this method for payment.
     */
    func payment(for token: String,
                 with data: PaymentMethodData,
                 presentationStyle style: PresentationStyle)
```

Payment with token

```
           /**
     Use this method for payment.
     */
    public func paymentWithToken(for mobileToken: String, with data: UnlimintSDK.PaymentTokenMethodData,
presentationStyle style: UnlimintSDK.PresentationStyle)
```

6:04

Description of the API data classes

```
public struct BillingAddress : Codable {

    /**
     ISO 3166-1 code of billing country: 2 or 3 latin letters or numeric code
     */
    public let country: String

    /**
     The state or province of the billing address associated with the card being used for this purchase.
     It's recommended to sent in following format: The country subdivision code defined in ISO 3166-2.
     May include whitespaces, hyphens, apostrophes, commas and dots
     */
    public let state: String?

    /**
     Billing postal code
     */
    public let zip: String

    /**
     Billing city. May include whitespaces, hyphens, apostrophes, commas and dots
     */
    public let city: String

    /**
     First line of the street address or equivalent local portion of the Cardholder billing address associated
with the card used for this purchase.
     May include whitespaces, hyphens, apostrophes, commas, quotes, dots, slashes and semicolons.
     Required (if available) unless market or regional mandate restricts sending this information.
     Field will be ignored if filing.id is presented in request (continue one-click scenario)
     */
    public let addrLine1: String

    /**
     Second line of the street address or equivalent local portion of the Cardholder billing address associated
with the card used for this purchase. Required (if available) unless market or regional mandate restricts
sending this information.
     Field will be ignored if filing.id is presented in request (continue one-click scenario)
     */
    public let addrLine2: String?

    public init(country: String, state: String?, zip: String, city: String, addrLine1: String, addrLine2:
String?)
}

public struct BindingCustomer : Codable {

    /**
     Customer ID is a unique identifier of a cardholder at the Recurring payments service. Each card used by a
cardholder within the service is linked to Customer ID and Filing ID.
     */
    public let id: String

    /**
     Customer’s e-mail address
     Optional for wallets where setting in PM "May omit customer email" is enabled
     */
    public let email: String

    /**
     Preferred locale for the payment page (ISO 639-1 language code).
     The default locale (en or other locale if it's set as default in Merchant account) will be applied if the
selected locale (received in request) is not supported.
     Supported locales are: ar, az, bg, cs, de, el, en, es, fr, hu, hy, id, it, ja, ka, ko, ms, nl, pl, pt, ro,
ru, sr, sv, th, tr, uk, vi, zh
     */
    public let locale: String?

    /**
     Customer’s phone number
     Recommended to send phone number in following format "+1 111111111" with country code and subscriber
```

```
sections (only digits are accepted) of the number, "+" as prefix and "space" as delimiter.
     Refer to ITU-E.164 for additional information on format and length.
     Mandatory for wallets where setting in PM "May omit customer email" is enabled and customer.email isn't
presented in request
     */
    public let phone: String?

    /**
     The home phone number provided by the Cardholder. Required (if available) unless market or regional
mandate restricts sending this information.
     Characters format: recommended to send phone number in following format "+1 111111111" with country code
and subscriber sections (only digits are accepted) of the number, "+" as prefix and "space" as delimiter.
     Refer to ITU-E.164 for additional information on format and length.
     Field will be ignored if filing.id is presented in request (continue one-click scenario)
     */
    public let homePhone: String?

    /**
     The work phone number provided by the Cardholder. Required (if available) unless market or regional
mandate restricts sending this information.
     Characters format: recommended to send phone number in following format "+1 111111111" with country code
and subscriber sections (only digits are accepted) of the number, "+" as prefix and "space" as delimiter.
     Refer to ITU-E.164 for additional information on format and length.
     Field will be ignored if filing.id is presented in request (continue one-click scenario)
     */
    public let workPhone: String?

    public init(id: String, email: String, locale: String? = nil, phone: String? = nil, homePhone: String? =
nil, workPhone: String? = nil)
}

public struct BindingMethodData : Codable {

    public struct MerchantOrder : Codable {

        /**
         Order ID used by the merchant's shopping cart
         */
        public let id: String

        /**
         Description of product/service being sold
         */
        public let description: String

        public init(description: String, id: String)
    }

    /**
     The currency for the lowest payment from card for binding
     */
    public let currency: UnlimintSDK.Currency

    /**
     Customer data
     */
    public let customer: UnlimintSDK.BindingCustomer

    /**
     Merchant order data
     */
    public let merchantOrder: UnlimintSDK.BindingMethodData.MerchantOrder?

    /**
     Card account data
     */
    public let cardAccount: UnlimintSDK.CheckCardAccount?

    public init(currency: UnlimintSDK.Currency, customer: UnlimintSDK.BindingCustomer, merchantOrder:
UnlimintSDK.BindingMethodData.MerchantOrder? = nil, cardAccount: UnlimintSDK.CheckCardAccount? = nil)
}
```

```
public struct CheckCardAccount : Codable {

    /**
     Address for billing
     */
    public let billingAddress: UnlimintSDK.BillingAddress?

    public init(billingAddress: UnlimintSDK.BillingAddress?)
}

public struct Currency : Codable {

    /**
     ISO 4217 currency code
     */
    public let value: String

    public init(with value: String)
}

public struct Item : Codable {

    /**
     The name of product / service, provided to the customer
     */
    public let name: String

    /**
     The description of product / service, provided to the customer
     */
    public let description: String?

    /**
     The count of product / service, provided to the customer. Any positive number
     */
    public let count: Int?

    /**
     Price of product / service with dot as a decimal separator, must be less than a million
     */
    public let price: Float?
}

public struct MainButtonStyle {

    public var cornerRadius: CGFloat

    public var titleColor: (normal: UIColor?, disabled: UIColor?)

    public var backgroundColor: (normal: UIColor?, disabled: UIColor?)

    public init(cornerRadius: CGFloat, titleColor: (normal: UIColor?, disabled: UIColor?), backgroundColor:
(normal: UIColor?, disabled: UIColor?))
}

public struct PaymentCustomer : Codable {

    /**
     The home phone number provided by the Cardholder. Required (if available) unless market or regional
mandate restricts sending this information.
     Characters format: recommended to send phone number in following format "+1 111111111" with country code
and subscriber sections (only digits are accepted) of the number, "+" as prefix and "space" as delimiter.
     Refer to ITU-E.164 for additional information on format and length.
     Field will be ignored if filing.id is presented in request (continue one-click scenario)
     */
    public let homePhone: String?

    /**
     The work phone number provided by the Cardholder. Required (if available) unless market or regional
mandate restricts sending this information.
```

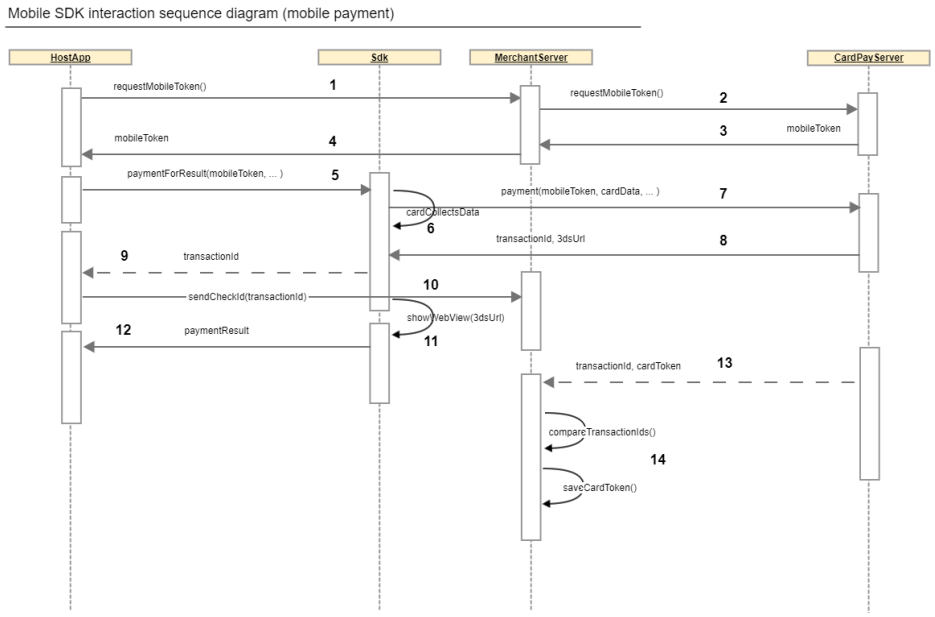
```
     Characters format: recommended to send phone number in following format "+1 111111111" with country code
and subscriber sections (only digits are accepted) of the number, "+" as prefix and "space" as delimiter.
     Refer to ITU-E.164 for additional information on format and length.
     Field will be ignored if filing.id is presented in request (continue one-click scenario)
     */
    public let workPhone: String?

    /**
     Customer’s e-mail address
     Optional for wallets where setting in PM "May omit customer email" is enabled
     */
    public let email: String

    /**
     Preferred locale for the payment page (ISO 639-1 language code).
     The default locale (en or other locale if it's set as default in Merchant account) will be applied if the
selected locale (received in request) is not supported.
     Supported locales are: ar, az, bg, cs, de, el, en, es, fr, hu, hy, id, it, ja, ka, ko, ms, nl, pl, pt, ro,
ru, sr, sv, th, tr, uk, vi, zh
     */
    public let locale: String?

    public init(homePhone: String?, workPhone: String?, email: String, locale: String?)
}

public struct PaymentData : Codable {

    /**
     The total transaction amount in selected currency with dot as a decimal separator, must be less than 100
millions
     */
    public let amount: Decimal

    /**
     ISO 4217 currency code
     */
    public let currency: String

    /**
     Note about the transaction that will not be displayed to customer
     */
    public let note: String?

    /**
     Short description of the service or product, must be enabled by CardPay manager to be used.
     For Visa cards: maximum length 25 symbols, for MasterCard cards - 22 symbols.
     */
    public let dynamicDescriptor: String?

    /**
     Identifies the type of transaction being authenticated.

     Values accepted:
     - 01 - Goods/ Service Purchase
     - 03 - Check Acceptance
     - 10 - Account Funding • 11 = Quasi-Cash Transaction
     - 28 - Prepaid Activation and Load Note: Values derived from the 8583 ISO Standard.
     */
    public let transType: String

    public init(amount: Decimal, currency: String, note: String?, dynamicDescriptor: String?, transType: String)
}

public struct PaymentMerchantOrder : Codable {

    /**
     Order ID used by the merchant's shopping cart
     */
    public let id: String

    /**
```

```
     Description of product/service being sold
     */
    public let description: String

    /**
     Shipping Address
     */
    public let shippingAddress: UnlimintSDK.ShippingAddress?

    /**
     Array of items (in the shopping cart)
     */
    public let items: [UnlimintSDK.Item]?

    public init(description: String, id: String, shippingAddress: UnlimintSDK.ShippingAddress? = nil, items:
[UnlimintSDK.Item]? = nil)
}

public struct PaymentMethodData : Codable {

    public struct CardAccount : Codable {

        /**
         Billing Address
         */
        public let billingAddress: UnlimintSDK.BillingAddress?

        public init(token: String?, billingAddress: UnlimintSDK.BillingAddress?)
    }

    /**
     The name of the merchant
     */
    public let merchantName: String

    /**
     Payment method type name
     */
    public let paymentMethod: String

    /**
     Customer data
     */
    public let customer: UnlimintSDK.PaymentCustomer

    /**
     Merchant order data
     */
    public let merchantOrder: UnlimintSDK.PaymentMerchantOrder

    /**
     Payment data
     */
    public let paymentData: UnlimintSDK.PaymentData

    /**
     Card account data
     */
    public let cardAccount: UnlimintSDK.PaymentMethodData.CardAccount?

    public init(with merchantName: String, paymentMethod: String, customer: UnlimintSDK.PaymentCustomer,
merchantOrder: UnlimintSDK.PaymentMerchantOrder, paymentData: UnlimintSDK.PaymentData, cardAccount: UnlimintSDK.
PaymentMethodData.CardAccount?)
}

public struct PaymentTokenMethodData : Codable {

    public struct CardAccount : Codable {

        /**
         Card token value, used instead of card information, except card.security_code (it's mandatory)
```

```
         */
        public let token: String

        /**
         The last 4 digits of the card number
         */
        public let pan: String

        /**
         Billing Address
         */
        public let billingAddress: UnlimintSDK.BillingAddress?

        public init(token: String, pan: String, billingAddress: UnlimintSDK.BillingAddress?)
    }

    /**
     The name of the merchant
     */
    public let merchantName: String

    /**
     Payment method type name
     */
    public let paymentMethod: String

    /**
     Customer data
     */
    public let customer: UnlimintSDK.PaymentCustomer

    /**
     Merchant order data
     */
    public let merchantOrder: UnlimintSDK.PaymentMerchantOrder

    /**
     Payment data
     */
    public let paymentData: UnlimintSDK.PaymentData

    /**
     Card account data
     */
    public let cardAccount: UnlimintSDK.PaymentTokenMethodData.CardAccount

    public init(with merchantName: String, paymentMethod: String, customer: UnlimintSDK.PaymentCustomer,
merchantOrder: UnlimintSDK.PaymentMerchantOrder, paymentData: UnlimintSDK.PaymentData, cardAccount: UnlimintSDK.
PaymentTokenMethodData.CardAccount)
}


public struct ShippingAddress : Codable {

    /**
     ISO 3166-1 code of delivery country: 2 or 3 latin letters or numeric code
     Required for BANKCARD payment method, if shipping_address is presented
     */
    public let country: String?

    /**
     The state or province of the shipping address associated with the card being used for this purchase.
     It's recommended to send in following format: The country subdivision code defined in ISO 3166-2.
     May include whitespaces, hyphens, apostrophes, commas and dots
     */
    public let state: String?

    /**
     Delivery postal code.
     For BANKCARD payment method - max length 12
     */
```

```
    public let zip: String?

    /**
     Delivery city. May include whitespaces, hyphens, apostrophes, commas and dots
     */
    public let city: String?

    /**
     Valid customer phone number
     */
    public let phone: String?

    /**
     First line of the street address or equivalent local portion of the Cardholder shipping address associated
with the card used for this purchase. Can include street and house number
     */
    public let addrLine1: String?

    /**
     Second line of the street address or equivalent local portion of the Cardholder shipping address
associated with the card used for this purchase.
     */
    public let addrLine2: String?
}
```

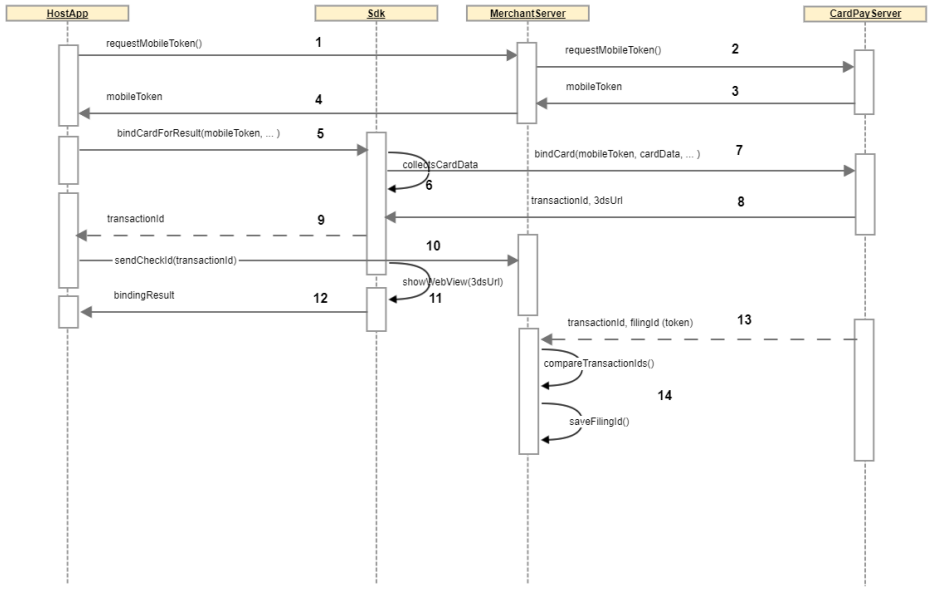
[iOS] Mobile API SDK interaction



User scenario for mobile payment process:

ID	Requirement text
MOB.SDK.P.01	Host (merchant) mobile application sends request for getting mobile token to merchant backend (requestMobileToken)
MOB.SDK.P.02	Merchant backend sends POST request (JSON) with valid access token for getting mobile token to API v3 endpoint api/mobile/token (Unlimint server)
MOB.SDK.P.03	API v3 returns response to merchant backend with requested mobile token
MOB.SDK.P.04	Merchant backend sends mobile token to the host application
MOB.SDK.P.05	Host application calls paymentForResult(mobileToken) method of the mobile SDK
MOB.SDK.P.06	Customer fills in card data in card data form (in SDK)
MOB.SDK.P.07	Mobile SDK sends request with customer, card data, received mobile token (JSON) for making a payment to API v3 endpoint api/mobile/payment
MOB.SDK.P.08	API v3 sends a payment response with redirect URL and transaction id (for 3DS verification) to the mobile SDK
MOB.SDK.P.09	Mobile SDK returns transaction id to the host application, host application resend it to the merchant backend
MOB.SDK.P.10	Host application sends transaction id to the merchant server
MOB.SDK.P.11	Mobile SDK presents webview with 3dsUrl to the customer for 3-D Secure verification
MOB.SDK.P.12	3-D Secure verification procedure passes and customer redirects to success or decline url (paymentResult)
MOB.SDK.P.13	Unlimint API v3 sends callback to the merchant backend (with transaction id and card token (if it was requested by customer)
MOB.SDK.P.14	Merchant backend compares received transaction id's from the host application and callback and saves a received card token for a future use (recommendations to do)

Mobile SDK interaction sequence diagram (binding card)



User scenario for card binding process:

ID	Requirement text
MOB.SDK.B.01	Host (merchant) mobile application sends request for getting mobile token to merchant backend (requestMobileToken)
MOB.SDK.B.02	Merchant backend sends POST request (JSON) with valid access token for getting mobile token to API v3 endpoint api/mobile/token (Unlimint server)
MOB.SDK.B.03	API v3 returns response to merchant backend with requested mobile token
MOB.SDK.B.04	Merchant backend sends mobile token to the host application
MOB.SDK.B.05	Host application calls bindCardForResult(mobileToken) method of the mobile SDK
MOB.SDK.B.06	Customer fills in card data in card data form (in SDK)
MOB.SDK.B.07	Mobile SDK sends request with customer, card data, received mobile token (JSON) for card binding to API v3 endpoint api/mobile /cardbinding
MOB.SDK.B.08	API v3 sends a card binding response with redirect URL and transaction id (for 3DS verification) to the mobile SDK
MOB.SDK.B.09	Mobile SDK returns transaction id to the host application, host application resend it to the merchant backend
MOB.SDK.B.10	Host application sends transaction id to the merchant server
MOB.SDK.B.11	Mobile SDK presents webview with 3dsUrl to the customer for 3-D Secure verification
MOB.SDK.B.12	3-D Secure verification procedure passes and customer redirects to success or decline url (bindingResult)
MOB.SDK.B.13	Unlimint API v3 sends callback to the merchant backend (with transaction id and filing id)
MOB.SDK.B.14	Merchant backend compares received transaction id's from the host application and callback and saves a received filing id for a future use (recommendations to do)

[iOS] Mobile SDK Customization

General

At the moment, the design system is implemented as a theme. You can use a standard theme or configure his own. Below is a table with the existing fields.

Theme:

Field	Component
navigationStyle	NavigationBarStyle
mainButtonStyle	MainButtonStyle
viewControllerStyle	ViewControllerStyle

Components

NavigationBarStyle

<table>
<tr><th>Field</th><th colspan="2">Data Type</th><th></th></tr>
<tr><td>bar</td><td colspan="2"><table><tr><th>Bar</th></tr><tr><td>largeNavBar</td></tr><tr><td>small</td></tr><tr><td>smallTranslucent</td></tr></table></td><td>Style of UINavigationBar</td></tr>
<tr><td>statusBarStyle</td><td colspan="2"><table><tr><th colspan="2">UIStatusBarStyle</th></tr><tr><td>default</td><td>A dark status bar, intended for use on light backgrounds.</td></tr><tr><td>lightContent</td><td>A light status bar, intended for use on dark backgrounds.</td></tr><tr><td>darkContent</td><td>A dark status bar, intended for use on light backgrounds.</td></tr></table></td><td>The style of the device’s status bar.</td></tr>
<tr><td>navigationBar</td><td colspan="2"><table><tr><th>Color</th></tr><tr><td>light</td></tr><tr><td>dark</td></tr><tr><td>transparentDark</td></tr><tr><td>transparentLight</td></tr><tr><td>custome (title, barTint, backgroundImage, background)</td></tr></table></td><td>Theme of NavigationBar</td></tr>
<tr><td>tintColor</td><td colspan="2">RGB color</td><td>Tint color of UINavigationBar</td></tr>
</table>

MainButtonStyle

cornerRadius	Float	Rounding radius of view.
titleColor	RGB color	Color of button text.
backgroundColor	**Color** normal: RGB color disabled: RGB color	Button background color.

ViewControllerStyle

backgroundColor	RGB color	The view's background color.