Mobile SDK Core

Введение

Mobile SDK Core - это скорее библиотека, чем SDK. Она используется для шифрования данных карты в seToken, который можно безопасно передать и сохранить в любой системе (например, на сервере продавца). Это хорошее решение для продавцов, которые хотят максимальной гибкости или не хотят иметь дело с данными карты на своих серверах.

Обратите внимание, что для использования этой библиотеки может потребоваться соответствие стандарту PCI DSS.

Способы оплаты, поддерживаемые этим SDK:

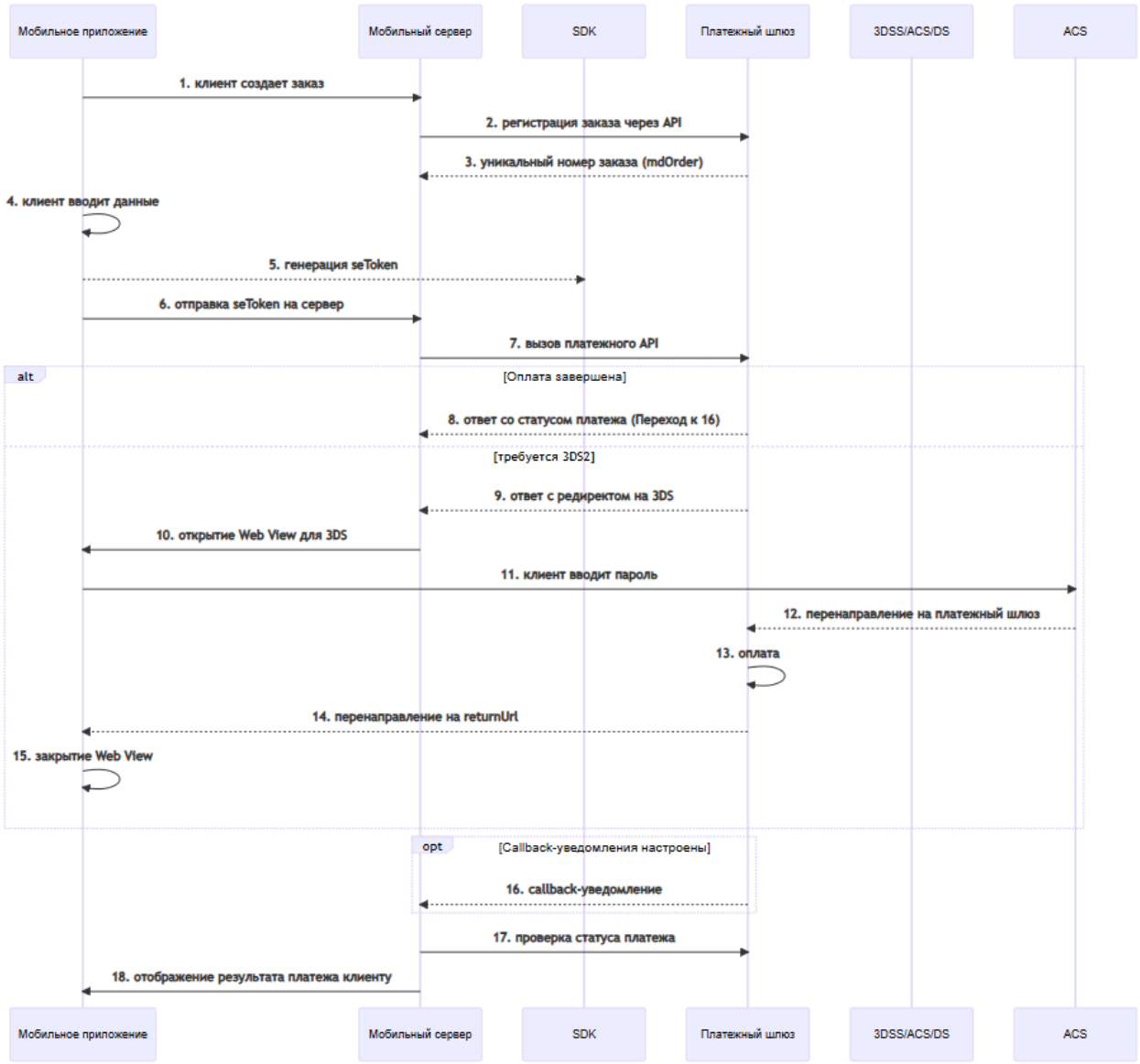
Способ оплаты	
Apple Pay	Не требуется
Google Pay	Не требуется
Банковская карта	Да
Сохраненные учетные данные/карта	Да

Если seToken содержит шифрованные данные о связке (`bindingId`), вместо запроса `paymentorder.do` для оплаты следует использовать запрос paymentOrderBinding.do.

Процесс оплаты SDK Core

Web View для 3DS

На приведенной ниже диаграмме показан процесс оплаты SDK Core с перенаправлением 3DS через Web View.



Описание:

1	Клиент создает заказ.
2	Мобильный сервер регистрирует этот заказ в платежном шлюзе через register.do. Используйте параметр `returnUrl` в качестве маркера для закрытия Web View после перенаправления из ACS на **Шаге 14**.
3	Мобильный сервер получает в ответе уникальный номер заказа `mdOrder`.
4	Клиент заполняет платежные данные в мобильном приложении.
5	Мобильное приложение вызывает SDK для создания seToken. (Android: **sdkCore.generateWithCard**; iOS: **CKCToken.generateWithCard**). Публичный ключ, который требуется в соответствующем методе, необходимо брать с online ресурса https://api.sbergate.com/payment/se/keys.do . Если по этой ссылке доступно несколько ключей, следует использовать первый ключ. (Имейте в виду, что для тестовой и рабочей сред используются разные ключи.)
6	Мобильное приложение отправляет seToken на мобильный сервер.

7	Мобильный сервер использует этот seToken для совершения платежа через paymentorder.do. • Используйте seToken вместо pan, cvc и даты истечения срока действия. • Не забудьте указать имя держателя карты в поле TEXT. Если вы не собираете имя держателя карты, просто отправьте значение CARDHOLDER.
8	Мобильный сервер получает ответ без редиректа на ACS. Это означает, что оплата завершена и нужно перейти к **Шагу 16**.
9	Мобильный сервер получает ответ с редиректом на ACS.
10	Мобильное приложение открывает Web View с данными редиректа на ACS.
11	Клиент вводит свой одноразовый пароль в форму ACS.
12	ACS перенаправляет клиента на платежный шлюз.
13	Платежный шлюз осуществляет платеж.
14	Платежный шлюз перенаправляет клиента на `returnUrl`, который можно использовать в качестве маркера для закрытия Web View.
15	Мобильное приложение закрывает Web View.
16	Платежный шлюз отправляет уведомление обратного вызова на сервер продавца, если оно настроено для продавца.
17	Мобильный сервер проверяет окончательный статус платежа через getOrderStatusExtended.do.
18	Мобильное приложение показывает результат платежа клиенту.

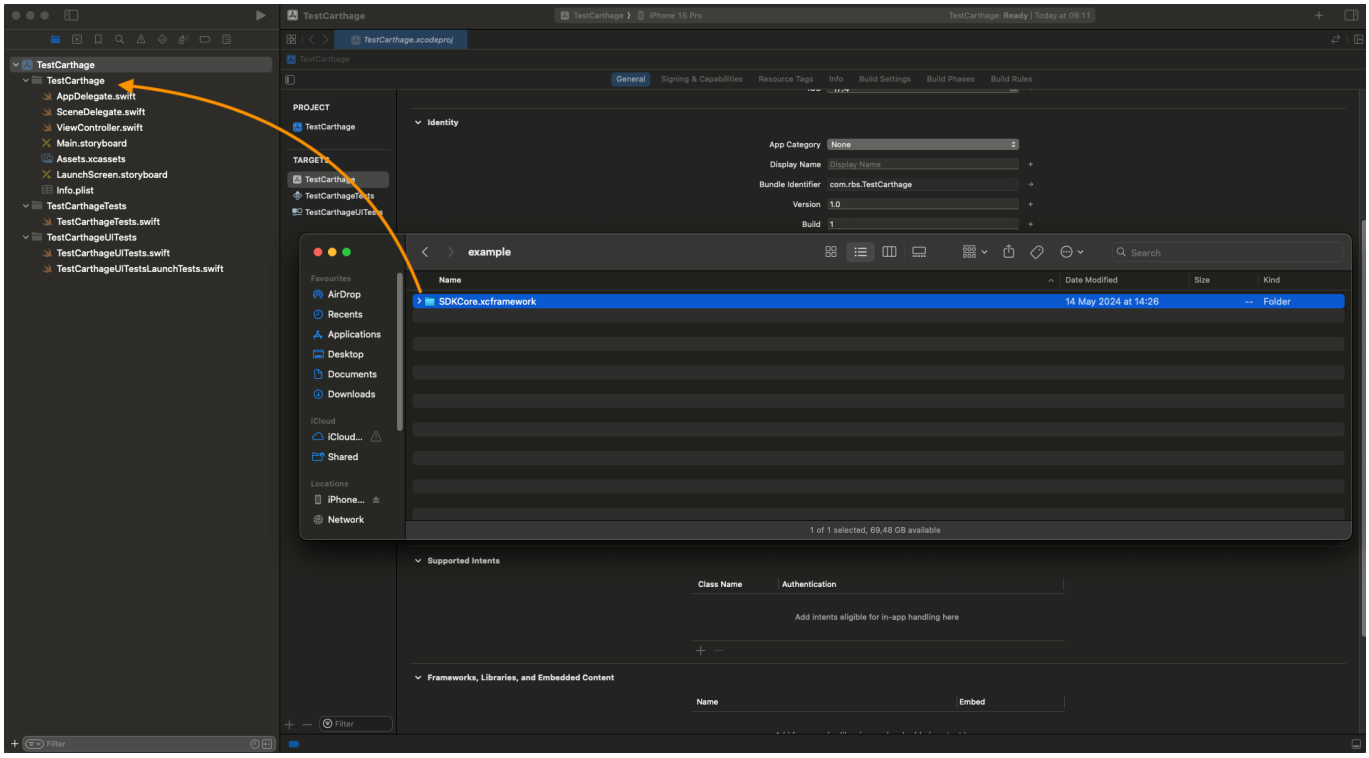
iOS

iOS-интеграция

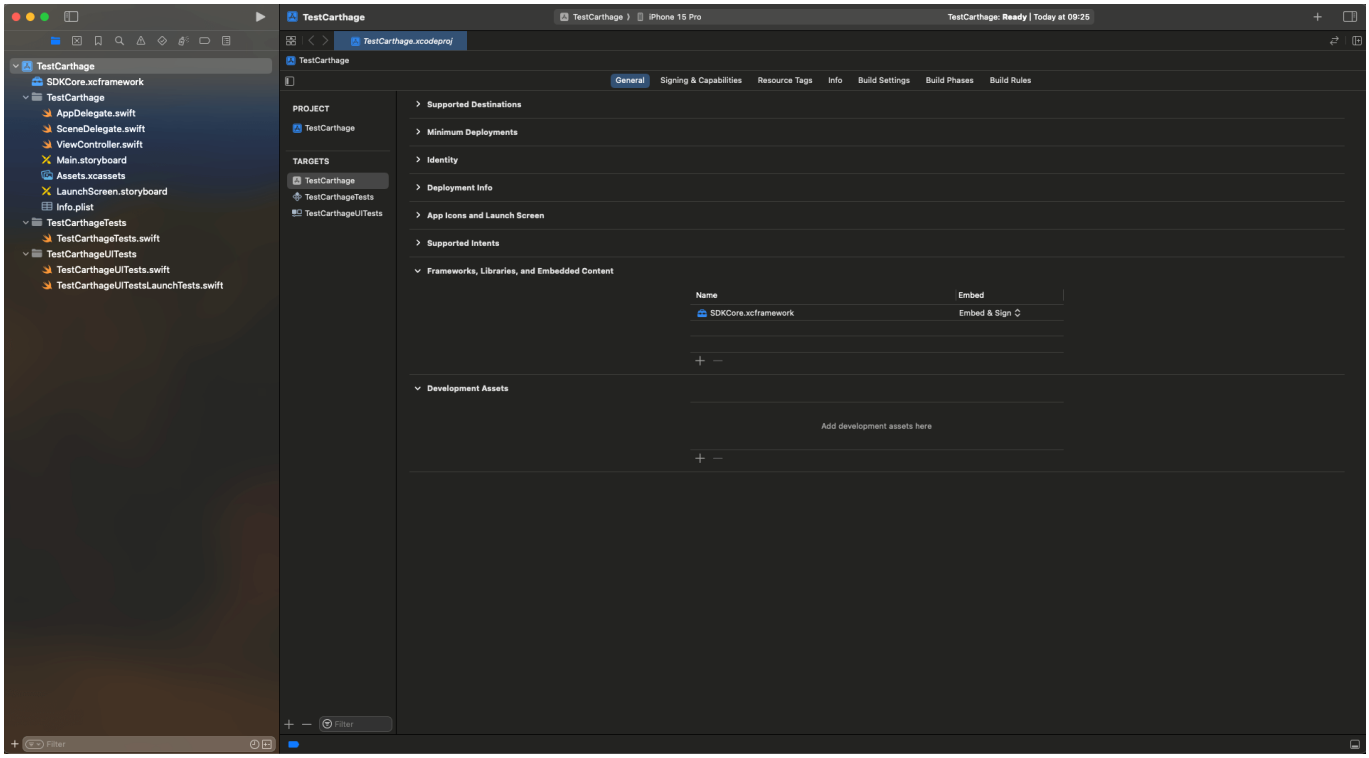
Интеграция SDKCore.framework

Вы можете интегрировать **SDKCore.framework**, добавив его вручную.

- Используйте последнюю версию фреймворка - sdkcore.zip ;
- Выберите файл `SDKCore.framework` и добавьте его в папку проекта.



- Откройте страницу **Targets → General → Frameworks, Libraries, and Embedded Content**. Для `SDKCore.framework` с столбце **Embed** замените **Do not Embed** на **Embed & Sign**.



- Затем импортируйте фреймворк в файл `ViewController.swift`.

```
//ViewController.swift
...
import SDKCore
```

```
...
```

Работа с API

Внешние зависимости

Для генерации токена необходимо задать открытый ключ.

```
let publicKey: String =
      "-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoIITqh9xlGx4tWA+aucb0V0YuFC9aXzJ
b0epdioSkq3qzNdRSZIxe/dHqcbMN2SyhzvN6MRVl3xyjGAV+lwk8poD4BRW3VwPUkT8xG/P/YLz
i5N8lY6ILlfw6WCtRPK5bKGGnERcX5dqL60LhOPRDSYT5NHbbp/J2eFWyLigdU9Sq7jvz9ixOLh6
xD7pgNgHtnOJ3Cw0Gqy03r3+m3+CBZwrzcp7ZFs41bit7/t1nIqgx78BCTPugap88Gs+8ZjdfDvu
DM+/3EwwK0UVTj0SQOv0E5KcEHENL9QQg3ujmEi+zAavulPqXH5907q21lwQeemzkTJH4o2RCCVe
YO+YrQIDAQAB-----END PUBLIC KEY-----"
```

Метод генерации токена

```
let sdkCore = SdkCore()

let cardParams = CardParams(
     pan: "4111111111111111",
     cvc: "123",
     expiryMMYY: "12/28",
     cardholder: "TEST CARDHOLDER",
     mdOrder: "mdOrder",
     pubKey: publicKey
)

let cardParamsConfig = SDKCoreConfig(
    paymentMethodParams: .cardParams(params: cardParams)
)
let tokenResult = sdkCore.generateWithConfig(config: cardParamsConfig)

let bindignParams = BindingParams(
    pubKey: publicKey,
    bindingId: "das",
    cvc: "123",
    mdOrder: "mdOrder"
)
let bindingParamsConfig = SDKCoreConfig(
    paymentMethodParams: .bindingParams(params: bindignParams)
)
```

```
let tokenResult = sdkCore.generateWithConfig(config: bindingParamsConfig)
```

Модели

CardParams

Название свойства	Тип данных	Значение по умолчанию	Необязательное	Описание
cardHolder	String	-	Да	Имя и фамилия держателя карты
mdOrder	String	-	Нет	Номер заказа
pan	String	-	Нет	Номер карты
pubKey	String	-	Нет	Открытый ключ
cvc	String	-	Нет	Секретный код карты
expiryMMYY	String	-	Нет	Срок действия карты

BindingParams

Название свойства	Тип данных	Значение по умолчанию	Необязательное	Описание
mdOrder	String	-	Нет	Номер заказа
bindingId	String	-	Нет	Номер связки для карты
cvc	String	-	Да	Секретный код карты
pubKey	String	-	Нет	Открытый ключ

Ошибки валидации полей

ParamField	Ошибка	
UNKNOWN	-	Неизвестная ошибка
PAN	required	Указано пустое поле
	invalid	Некорректное значение
	invalid-format	Используются недопустимые символы. Доступны только цифры.
CVC	required	Указано пустое поле
	invalid	Некорректное значение
EXPIRY	required	Указано пустое поле
	invalid	Некорректное значение
	invalid-format	Формат не соответствует шаблону MM/YY
CARDHOLDER	required	Указано пустое поле
	invalid	Некорректное значение
	invalid-format	Используются недопустимые символы. Доступны только буквы и пробелы.
BINDING_ID	required	Указано пустое поле
	invalid	Некорректное значение
MD_ORDER	required	Указано пустое поле
	invalid	Некорректное значение
PUB_KEY	required	Указано пустое поле

Android

Android-интеграция

Подключение к Gradle проекту путем добавления файлов .aar библиотеки

Последняя версия файла библиотеки -

sdk_core-release.zip

.

Необходимо добавить файл библиотеки sdk_core-release.aar в папку libs, а затем указать зависимость добавленной библиотеки.

build.gradle.kts

```
allprojects {
    repositories {
        // ...
        flatDir {
            dirs("libs")
        }
    }
}

dependencies {
    // dependency is mandatory to add
    implementation(group = "", name = "sdk_core-release", ext = "aar")
}
```

build.gradle

```
allprojects {
    repositories {
        // ...
        flatDir {
            dirs 'libs'
        }
    }
}

dependencies {
```

```
    // dependency is mandatory to add
    implementation(group: '', name: 'sdk_core-release', ext: 'aar')
}
```

Конфигурация Android

Логирование

Внутренние процессы логируются с тегом SDK-Core. Вы также можете логировать свои процессы.

Логирование доступно через объект Logger.

- Для добавления log- интерфейсов необходимо вызвать Logger-метод addLogInterface().

Пример для логирования в LogCat:

```
...
    Logger.addLogInterface(object : LogInterface {
        override fun log(classMethod: Class<Any>, tag: String, message:
String, exception: Exception?) {
                Log.i(tag, "$classMethod: $message", exception)
            }
        })
...
```

По умолчанию используется тег SDK-Core. Вы можете установить свой собственный, если хотите.

- Для логирования собственных событий необходимо вызвать Logger-метод log().

```
...
     Logger.log(this.javaClass, "MyTag", "My process...", null)
...
```

Пример Kotlin_core (без графического интерфейса)

Пример формирования криптограммы:

```
import net.payrdr.mobile.payment.sdk.core.SDKCore
import net.payrdr.mobile.payment.sdk.core.TokenResult
```

```
import net.payrdr.mobile.payment.sdk.core.model.BindingParams
import net.payrdr.mobile.payment.sdk.core.model.CardParams
import net.payrdr.mobile.payment.sdk.core.validation.BaseValidator
import net.payrdr.mobile.payment.sdk.core.validation.CardCodeValidator
import net.payrdr.mobile.payment.sdk.core.validation.CardExpiryValidator
import net.payrdr.mobile.payment.sdk.core.validation.CardHolderValidator
import net.payrdr.mobile.payment.sdk.core.validation.CardNumberValidator
import net.payrdr.mobile.payment.sdk.core.validation.OrderNumberValidator

class MainActivity : AppCompatActivity() {
    // initialization of validators for card information entry fields
    private val cardNumberValidator by lazy { CardNumberValidator(this) }
    private val cardExpiryValidator by lazy { CardExpiryValidator(this) }
    private val cardCodeValidator by lazy { CardCodeValidator(this) }
    private val cardHolderValidator by lazy { CardHolderValidator(this) }
    private val orderNumberValidator by lazy { OrderNumberValidator(this) }
    private val sdkCore by lazy { SDKCore(context = this) }

    override fun onCreate(savedInstanceState: Bundle?) {
        // installation of validators on the card information entry fields
        cardNumberInput.setupValidator(cardNumberValidator)
        cardExpiryInput.setupValidator(cardExpiryValidator)
        cardCodeInput.setupValidator(cardCodeValidator)
        cardHolderInput.setupValidator(cardHolderValidator)
        mdOrderInput.setupValidator(orderNumberValidator)

        // creation of an object and initialization of fields for a new card
        val params = NewPaymentMethodCardParams(
            pan = cardNumberInput.text.toString(),
            cvc = cardCodeInput.text.toString(),
            expiryMMYY = cardExpiryInput.text.toString(),
            cardHolder = cardHolderInput.text.toString(),
            pubKey = pubKeyInput.text.toString()
        )
        // method call to get the cryptogram for a new card
        sdkCore.generateWithConfig(SDKCoreConfig(params))

        // Creation of an object and initialization of fields for the linked
card
        val params = NewPaymentMethodStoredCardParams(
            storedPaymentId = "storedPaymentMethodId",
            cvc = "123",
            pubKey = pubKeyInput.text.toString()
        )
        // method call to get the cryptogram for the linked card
        sdkCore.generateWithConfig(SDKCoreConfig(params))
    }
}
```

Модели

NewPaymentMethodCardParams

Название свойства	Тип данных	Значение по умолчанию	Необязательно	Описание
pan	String	-	Нет	Номер карты
cvc	String	-	Нет	Секретный код карты
expiryMMYY	String	-	Нет	Срок действия карты
cardHolder	String	-	Нет	Имя и фамилия держателя карты
pubKey	String	-	Нет	Открытый ключ

NewPaymentMethodStoredCardParams

Название свойства	Тип данных	Значение по умолчанию	Необязательно	Описание
storedPaymentId	String	-	Нет	Номер связки для карты
cvc	String	-	Нет	Секретный код карты
pubKey	String	-	Нет	Открытый ключ

TokenResult

Название свойства	Тип данных	Значение по умолчанию	Необязательно	Описание
token	String	-	Нет	Токен как строка
errors	Map	-	Нет	Ошибка при генерации токена

Ошибки валидации полей

ParamField	Ошибка	Описание
PAN	required	Указано пустое поле
	invalid	Некорректное значение
	invalid-format	Используются недопустимые символы. Доступны только цифры.
CVC	required	Указано пустое поле
	invalid	Некорректное значение
EXPIRY	required	Указано пустое поле
	invalid	Некорректное значение
	invalid-format	Формат не соответствует шаблону MM/YY
CARDHOLDER	required	Указано пустое поле
	invalid	Некорректное значение
	invalid-format	Используются недопустимые символы. Доступны только буквы и пробелы.
PUB_KEY	required	Указано пустое поле
STORED_PAYMENT_ID	required	Указано пустое поле
	invalid	Некорректное значение

Работа с API

Внешние зависимости

Для генерации токена необходимо установить открытый ключ.

```
val publicKey: String =
    "-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoIITqh9xlGx4tWA+aucb0V0YuFC9aXzJ
b0epdioSkq3qzNdRSZIxe/dHqcbMN2SyhzvN6MRVl3xyjGAV+lwk8poD4BRW3VwPUkT8xG/P/YLz
i5N8lY6ILlfw6WCtRPK5bKGGnERcX5dqL60LhOPRDSYT5NHbbp/J2eFWyLigdU9Sq7jvz9ixOLh6
xD7pgNgHtnOJ3Cw0Gqy03r3+m3+CBZwrzcp7ZFs41bit7/t1nIqgx78BCTPugap88Gs+8ZjdfDvu
DM+/3EwwK0UVTj0SQOv0E5KcEHENL9QQg3ujmEi+zAavulPqXH5907q21lwQeemzkTJH4o2RCCVe
YO+YrQIDAQAB-----END PUBLIC KEY-----"
```

Метод генерации токена

```
// TokenResult with CardParams
val cardParams: CardParams = CardParams(
    mdOrder = "mdOrder",
    pan = "4111111111111111",
    cvc = "123",
    expiryMMYY = "12/28",
    cardHolder = "TEST CARDHOLDER",
    pubKey = "publicKey"
)
val tokenResult = sdkCore.generationWithConfig(paymentCardParams =
cardParams)

// TokenResult with BindingParams
val bindingParams: BindingParams = BindingParams(
    mdOrder = "mdOrder",
    bindingID = "das",
    cvc = "123",
    pubKey = "publicKey"
)

val tokenResult = sdkCore.generationWithConfig(paymentCardParams =
bindingParams)
```

React Native

Для реализации интеграции Mobile SDK Core в React Native-приложение необходимо настроить

мост — обёртку на Kotlin/Java или Swift/Objective-C, которая будет принимать вызовы из JavaScript и передавать их в нативный код SDK. З Мосты в React Native позволяют вашему JS-коду вызывать методы нативных библиотек так же просто, как обычные асинхронные функции: вы объявляете нативный модуль с методами, которые возвращают результат через Promise, React Native автоматически связывает эти методы с JavaScript, и дальше вы работаете с ними в приложении без глубоких знаний платформенных API.

IOS

1. Интегрируйте SDKCore в ваш проект в соответствии с разделом iOS-интеграция.

2. Проверьте, что для фреймворка `DKCore.xcframework` в колонке **Embed** выбрано **Embed & Sign**.

3. Создайте Swift-модуль `RadarSdkBridge.swift` со следующим содержимым:

```
import Foundation
import React
import SDKCore

@objc(RadarSdk)
class RadarSdkBridge: NSObject {
    private let sdkCore = SdkCore()

    @objc
    func generateNewCardToken(
        _ pan: String,
        cvc: String,
        expiryMMYY: String,
        cardHolder: String,
        mdOrder: String,
        pubKey: String,
        resolver resolve: RCTPromiseResolveBlock,
        rejecter reject: RCTPromiseRejectBlock
    ) {
        do {
            let params = CardParams(
                pan: pan,
                cvc: cvc,
                expiryMMYY: expiryMMYY,
                cardholder: cardHolder,
                mdOrder: mdOrder,
                pubKey: pubKey
            )
            let config = SDKCoreConfig(paymentMethodParams:
.cardParams(params: params))
            let result = try sdkCore.generateWithConfig(config: config)
            resolve(result.token)
        } catch {
```

```
            reject("TOKEN_ERROR", error.localizedDescription, error)
        }
    }

    @objc
    func generateStoredCardToken(
        _ storedPaymentId: String,
        cvc: String,
        pubKey: String,
        mdOrder: String,
        resolver resolve: RCTPromiseResolveBlock,
        rejecter reject: RCTPromiseRejectBlock
    ) {
        do {
            let params = BindingParams(
                pubKey: pubKey,
                bindingId: storedPaymentId,
                cvc: cvc,
                mdOrder: mdOrder
            )
            let config = SDKCoreConfig(paymentMethodParams:
.bindingParams(params: params))
            let result = try sdkCore.generateWithConfig(config: config)
            resolve(result.token)
        } catch {
            reject("TOKEN_ERROR", error.localizedDescription, error)
        }
    }

    @objc
    static func requiresMainQueueSetup() -> Bool {
        return false
    }
}
```

4. Реализуйте Objective-C мост в файле `RadarSdkBridge.m`:

```
#import <React/RCTBridgeModule.h>

@interface RCT_EXTERN_MODULE(RadarSdk, NSObject)

RCT_EXTERN_METHOD(generateNewCardToken:
  (NSString *)pan
  cvc:(NSString *)cvc
  expiryMMYY:(NSString *)expiryMMYY
  cardHolder:(NSString *)cardHolder
  mdOrder:(NSString *)mdOrder
  pubKey:(NSString *)pubKey
  resolver:(RCTPromiseResolveBlock)resolve
  rejecter:(RCTPromiseRejectBlock)reject
)
```

```
RCT_EXTERN_METHOD(generateStoredCardToken:
  (NSString *)storedPaymentId
  cvc:(NSString *)cvc
  pubKey:(NSString *)pubKey
  mdOrder:(NSString *)mdOrder
  resolver:(RCTPromiseResolveBlock)resolve
  rejecter:(RCTPromiseRejectBlock)reject
)

@end
```

Android

1. Скачайте файлы `sdk_core-release.aar` и `sdk_logs-release.aar` и скопируйте их в папку `android/app/libs/` согласно разделу Android-интеграция .

2. Добавьте плоский репозиторий и зависимости в файл `android/app/build.gradle`:

```
android {
  // ... остальная конфигурация вашего проекта ...
}

repositories {
    flatDir {
        dirs 'libs'
    }
}

dependencies {
  // ... прочие зависимости ...
    implementation(name: 'sdk_core-release', ext: 'aar')
    implementation(name: 'sdk_logs-release', ext: 'aar')
}
```

Реализация Native Module для Kotlin

1. Создайте файл RadarSdkModule.kt в директории `android/app/src/main/java/…/radar/`:

```
package com.mobilesdkdemo.radar

import com.facebook.react.bridge.Promise
import com.facebook.react.bridge.ReactApplicationContext
import com.facebook.react.bridge.ReactContextBaseJavaModule
import com.facebook.react.bridge.ReactMethod
import net.payrdr.mobile.payment.sdk.core.SDKCore
import net.payrdr.mobile.payment.sdk.core.model.SDKCoreConfig
import net.payrdr.mobile.payment.sdk.core.model.CardParams
```

```
import net.payrdr.mobile.payment.sdk.core.model.BindingParams

class RadarSdkModule(reactContext: ReactApplicationContext) :
ReactContextBaseJavaModule(reactContext) {
    override fun getName(): String = "RadarSdk"

    private val sdkCore by lazy { SDKCore(context = reactApplicationContext)
}

    @ReactMethod
    fun generateNewCardToken(
        pan: String,
        cvc: String,
        expiryMMYY: String,
        cardHolder: String,
        mdOrder: String,
        pubKey: String,
        promise: Promise
    ) {
        try {
            val params = CardParams(
                pan = pan,
                cvc = cvc,
                expiryMMYY = expiryMMYY,
                cardHolder = cardHolder,
                mdOrder = mdOrder,
                pubKey = pubKey
            )
            val config = SDKCoreConfig(paymentCardParams = params)
            val result = sdkCore.generateWithConfig(config)
            promise.resolve(result.token)
        } catch (e: Exception) {
            promise.reject("TOKEN_ERROR", e)
        }
    }

    @ReactMethod
    fun generateStoredCardToken(
        storedPaymentId: String,
        cvc: String,
        pubKey: String,
        mdOrder: String,
        promise: Promise
    ) {
        try {
            val params = BindingParams(
                bindingID = storedPaymentId,
                mdOrder = mdOrder,
                cvc = cvc,
                pubKey = pubKey
            )
```

```
            val config = SDKCoreConfig(paymentCardParams = params)
            val result = sdkCore.generateWithConfig(config)
            promise.resolve(result.token)
        } catch (e: Exception) {
            promise.reject("TOKEN_ERROR", e)
        }
    }
}
```

2. Создайте пакет `RadarSdkPackage.kt` в той же директории:

```
package com.mobilesdkdemo.radar

import com.facebook.react.ReactPackage
import com.facebook.react.bridge.NativeModule
import com.facebook.react.bridge.ReactApplicationContext
import com.facebook.react.uimanager.ViewManager

class RadarSdkPackage : ReactPackage {
    override fun createNativeModules(reactContext: ReactApplicationContext):
List<NativeModule> =
        listOf(RadarSdkModule(reactContext))

    override fun createViewManagers(
        reactContext: ReactApplicationContext
    ): List<ViewManager<*, *>> = emptyList()
}
```

3. Зарегистрируйте созданный пакет в `MainApplication.kt`:

```
package com.mobilesdkdemo

import android.app.Application
import com.facebook.react.PackageList
import com.facebook.react.ReactApplication
import com.facebook.react.ReactNativeHost
import com.facebook.react.ReactHost
import com.facebook.react.defaults.DefaultNewArchitectureEntryPoint.load
import com.facebook.react.defaults.DefaultReactNativeHost
import com.facebook.soloader.SoLoader
import com.mobilesdkdemo.radar.RadarSdkPackage

class MainApplication : Application(), ReactApplication {

    override val reactNativeHost: ReactNativeHost =
        object : DefaultReactNativeHost(this) {
            override fun getPackages(): List<ReactPackage> =
```

```
                PackageList(this).packages.apply {
                    add(RadarSdkPackage()) // Регистрируем пакет
                }

            override fun getJSMainModuleName(): String = "index"
            override fun getUseDeveloperSupport(): Boolean =
BuildConfig.DEBUG
            override val isNewArchEnabled: Boolean =
BuildConfig.IS_NEW_ARCHITECTURE_ENABLED
            override val isHermesEnabled: Boolean =
BuildConfig.IS_HERMES_ENABLED
        }

    override val reactHost: ReactHost
        get() = getReactNativeHost(applicationContext, reactNativeHost)

    override fun onCreate() {
        super.onCreate()
        SoLoader.init(this, false)
        if (BuildConfig.IS_NEW_ARCHITECTURE_ENABLED) {
            load()
        }
    }
}
```

Использование в React Native

Используйте методы из JavaScript/TypeScript следующим образом:

```
import { NativeModules } from 'react-native';
const { RadarSdk } = NativeModules;
// Генерация токена для новой карты
const token = await RadarSdk.generateNewCardToken(
  '4111111111111111', // PAN
  '123',              // CVC
  '12/25',            // Expiry MM/YY
  'CARDHOLDER NAME',  // Имя держателя карты
  'mdOrder',          // Номер, полученный при регистрации заказа
  '-----BEGIN PUBLIC KEY------...' // Публичный ключ из /payment/se/keys.do
);
// Генерация токена для привязанной карты
const bindToken = await RadarSdk.generateStoredCardToken(
  'bindingId',        // ID сохранённой карты
  '123',              // CVC
  '-----BEGIN PUBLIC KEY-----…', // Публичный ключ из /payment/se/keys.do
  'mdOrder'           // Номер, полученный при регистрации заказа
);
```