

- 1. Mobile SDK Android Guide ..... 2
  - 1.1 Mobile SDK Intro ..... 3
  - 1.2 Mobile SDK API description ..... 4
  - 1.3 Mobile API SDK interaction ..... 14
  - 1.4 Mobile SDK customisation ..... 17

# Mobile SDK Android Guide

- [Mobile SDK Intro](#)
- [Mobile SDK API description](#)
- [Mobile API SDK interaction](#)
- [Mobile SDK customisation](#)

# Mobile SDK Intro

Unlimint mobile SDK for Android (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

Android minSdkVersion 19

## Wise notice

Unlimint mobile SDK just reminds you that the device is rooted. Please take a look at our [security approach](#). You can accept working on Rooted devices or reject a payment.

For integration with MobileSdk you have to do the following steps:

1. Add Mobile Sdk .aar to your project and next dependencies to your Android project

```
repositories {
    google()
    jcenter()

    maven { url "https://repos.unlimint.io/repository/mobile-sdk/" }
}

implementation 'com.unlimint.sdk:mobile-sdk:$version'
```

2. Call this method "MobileSdk.bindNewCardForResult(...)" for card binding or "MobileSdk.paymentForResult(...)" for making a mobile payment
3. For receiving the result of card binding or mobile payment you have to implement onActivityResult(...) method
4. After sending a card data (using the "MobileSdk.bindNewCardForResult(...)" or "MobileSdk.paymentForResult(...)" methods) and before 3DS verification procedure SDK receives "transaction\_id" parameter,
5. Mobile application should send this parameter to your Server part.
6. Please, create an instance of Service and wait for the transaction\_id

```
LocalBroadcastManager.getInstance(this).registerReceiver(listener, IntentFilter(MobileSdk.TransactionData.TRANSACTION_ACTION))
val transactionId = intent?.getStringExtra(MobileSdk.TransactionData.TRANSACTION_ID)
```

7. Add style to your styles.xml

```
<style name="MobileSdkStyle" parent="Theme.MobileSdk.Light">
```

8. Add <activity> to your manifest.xml with your added theme

For card binding

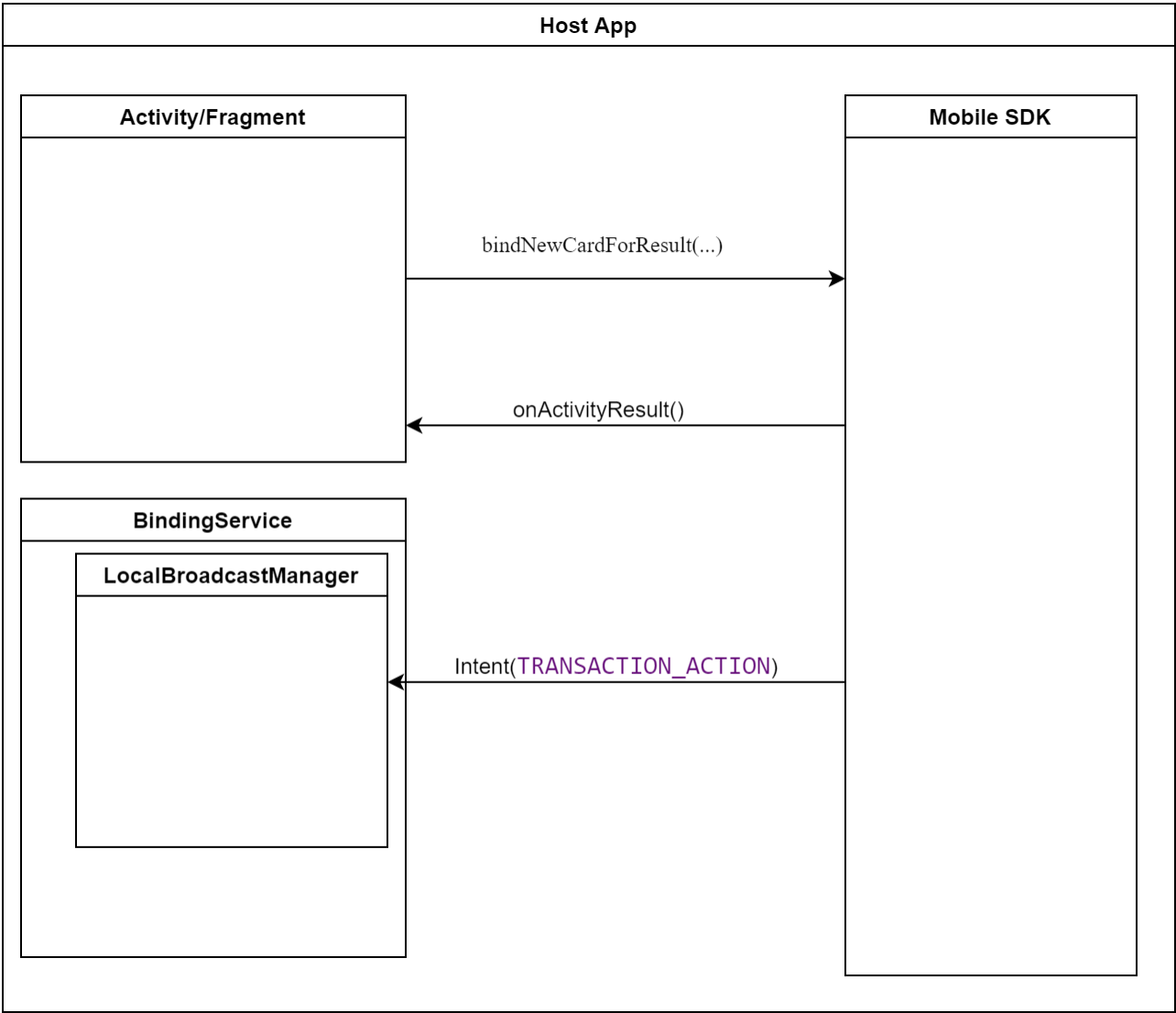
```
<activity
    android:name=".sdk.scenario.binding.view.BindActivity"
    android:launchMode="singleInstance"
    android:screenOrientation="portrait"
    android:theme="@style/MobileSdkStyle"
    android:windowSoftInputMode="adjustResize" />
```

For mobile payment

```
<activity
    android:name=".sdk.scenario.payment.view.PaymentActivity"
    android:launchMode="singleInstance"
    android:screenOrientation="portrait"
    android:theme="@style/MobileSdkStyle"
    android:windowSoftInputMode="adjustResize" />
```

# Mobile SDK API description

Card binding API description:

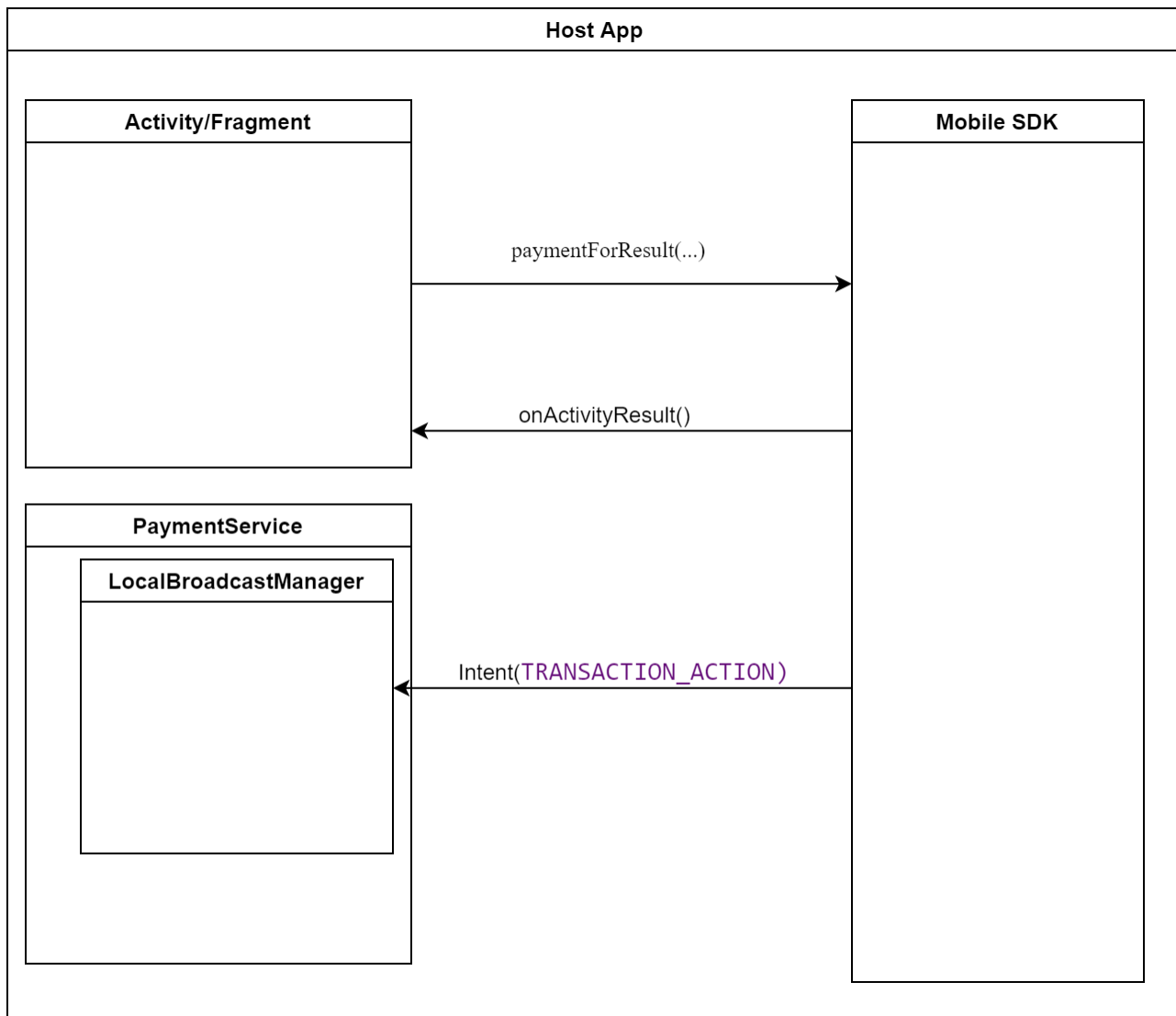


```

/**
 * Binds bank card. If bank card was bound, rewrites it
 *
 * *Mandatory params*
 * @param activity for calling our Activity
 * @param mobileAuthorizationToken Mobile token received from Unlimint (authentication mobile token)
 * @param bindingMethodData needed information for binding
 * @param requestCode code for startActivityResult
 *
 * @return You have to catch the answer from onActivityResult with your *requestCode*
 * if resultCode == Activity.RESULT_OK then get result data from Intent with MobileSdk.ARG_LAST_4_DIGITS
flag
 * if resultCode == Activity.RESULT_CANCEL then you may get *Exception* from Intent with MobileSdk.
EXCEPTION flag
 * data: Intent. If data is empty, then the user just closed the SDK
 *
 * Variety of errors at onActivityResult()
 * @exception MobileSdkServiceUnavailableException some io errors
 * @exception MobileSdkUnauthorizedException try to refresh your *mobileToken*
 * @exception MobileSdkIllegalStateException some business errors, look at the message
 * @exception MobileSdkBindingDeclineException Acquirer rejected map binding
 * @exception MobileSdkSecurityException security error
 *
 * Use LocalBroadcast for catching security error with Intent([MobileSdk.TransactionData.
TRANSACTION_ACTION])
 * and [MobileSdk.TransactionData.INSECURE_EVENT] key
 *
 * After filling bank card requisites, you have to get TRANSACTION_ID from LocalBroadcast
 * with Intent(UnlimintSdk.TransactionData.TRANSACTION_ACTION). Get the data from received Intent with name
MobileSdk.TransactionData.TRANSACTION_ID
 * Then send it to your server for checking.
 */
fun bindNewCardForResult(
    activity: AppCompatActivity,
    mobileAuthorizationToken: String,
    bindingMethodData: Binding.Data,
    requestCode: Int
)

```

Mobile payment API description:



```

/**
 * Make a payment
 *
 * *Mandatory params*
 * @param activity for calling our Activity
 * @param mobileAuthorizationToken Mobile token received from Unlimint (authentication mobile token)
 * @param paymentMethodData needed information for payment
 * @param requestCode code for startActivityResult
 *
 * @return You have to catch the answer from onActivityResult with your *requestCode*
 * if resultCode == Activity.RESULT_OK then get result data from Intent with MobileSdk.ARG_LAST_4_DIGITS
flag if user's decided save the card for next payment or empty Intent
 * if resultCode == Activity.RESULT_CANCEL then you may get *Exception* from Intent with MobileSdk.
EXCEPTION flag
 * data: Intent. If data is empty, then the user just closed the SDK
 *
 * Variety of errors at onActivityResult()
 * @exception MobileSdkServiceUnavailableException some io errors
 * @exception MobileSdkIllegalStateException some business errors, look at the message
 * @exception MobileSdkPaymentDeclineException Payment was rejected
 * @exception MobileSdkSecurityException security error
 *
 * Use LocalBroadcast for catching security error with Intent([MobileSdk.TransactionData.
TRANSACTION_ACTION])
 * and [MobileSdk.TransactionData.INSECURE_EVENT] key

```

```

*
* After filling bank card requisites, you have to get TRANSACTION_ID from LocalBroadcast
* with Intent(MobileSdk.TransactionData.TRANSACTION_ACTION). Get the data from received Intent with name
MobileSdk.TransactionData.TRANSACTION_ID
* Then send it to your server for checking.
*/
fun paymentForResult(
    activity: AppCompatActivity,
    mobileAuthorizationToken: String,
    paymentMethodData: Payment.Data,
    requestCode: Int
)

/**
* Make a payment with the token of the saved card
*
* *Mandatory params*
* @param activity for calling our Activity
* @param mobileAuthorizationToken Mobile token received from Unlimint (authentication mobile token)
* @param tokenPaymentMethodData needed information for payment with saved bankcard token
* @param requestCode code for startActivityForResult
*
* @return You have to catch the answer from onActivityResult with your *requestCode*
* if resultCode == Activity.RESULT_OK then get result data from empty Intent
* if resultCode == Activity.RESULT_CANCEL then you may get *Exception* from Intent with MobileSdk.
EXCEPTION flag
* data: Intent. If data is empty, then the user just closed the SDK
*
* Variety of errors at onActivityResult()
* @exception MobileSdkServiceUnavailableException some io errors
* @exception MobileSdkIllegalStateException some business errors, look at the message
* @exception MobileSdkPaymentDeclineException Payment was rejected
* @exception MobileSdkSecurityException security error
*
* Use LocalBroadcast for catching security error with Intent([MobileSdk.TransactionData.
TRANSACTION_ACTION])
* and [MobileSdk.TransactionData.INSECURE_EVENT] key
*
* After filling bank card requisites, you have to get TRANSACTION_ID from LocalBroadcast
* with Intent(MobileSdk.TransactionData.TRANSACTION_ACTION). Get the data from received Intent with name
MobileSdk.TransactionData.TRANSACTION_ID
* Then send it to your server for checking.
*/
fun paymentForResult(
    activity: AppCompatActivity,
    mobileAuthorizationToken: String,
    tokenPaymentMethodData: TokenPayment.Data,
    requestCode: Int
)

```

## Description of the API data classes

```

object Binding {
    data class Data(
        /**
         * the currency for the lowest payment from card for binding
         */
        val currency: Currency,
        /**
         * Customer data
         */
        val customer: Customer,
        /**
         * Merchant order data
         */
        val merchantOrder: MerchantOrder? = null,
    )
}

```

```

    /**
     * Card account data
     */
    val cardAccount: CardAccount? = null
)

data class CardAccount(
    /**
     * Billing Address
     */
    val billingAddress: BillingAddress?
) : Serializable
}

object Payment {
    data class Data(
        /**
         * The name of the merchant
         */
        val merchantName: String,
        /**
         * Customer data
         */
        val customer: Customer,
        /**
         * Merchant order data
         */
        val merchantOrder: MerchantOrder,
        /**
         * Payment data
         */
        val paymentData: PaymentData,
        /**
         * Card account data
         */
        val cardAccount: CardAccount? = null
    )

    data class CardAccount(
        /**
         * Billing Address
         */
        val billingAddress: BillingAddress?
    ) : Serializable
}

object TokenPayment {
    data class Data(
        /**
         * The name of the merchant
         */
        val merchantName: String,
        /**
         * Customer data
         */
        val customer: Customer,
        /**
         * Merchant order data
         */
        val merchantOrder: MerchantOrder,
        /**
         * Payment data
         */
        val paymentData: PaymentData,
        /**
         * Card account data
         */
        val cardAccount: CardAccount
    )

    data class CardAccount(

```



```

    /**
     * Card token data (used instead of card information)
     */
    val tokenData: TokenData,
    /**
     * Billing Address
     */
    val billingAddress: BillingAddress?
) : Serializable

data class TokenData(
    /**
     * Card token value, used instead of card information, except card.security_code (it's mandatory)
     */
    val token: String,
    /**
     * Card pan last four digits (Needed to be displayed on the screen)
     */
    val last4PanDigits: String
) : Serializable
}

data class ShippingAddress(
    /**
     * 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
     */
    val 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.
     */
    val addrLine2: String?,
    /**
     * Delivery city. May include whitespaces, hyphens, apostrophes, commas and dots
     */
    val city: String,
    /**
     * 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
     */
    val country: String,
    /**
     * Valid customer_layout phone number
     */
    val phone: 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
     */
    val state: String?,
    /**
     * Delivery postal code
     * For BANKCARD payment method - max length 12
     */
    val zip: String
) : Serializable

data class PaymentData(
    /**
     * The total transaction amount in selected currency with dot as a decimal separator, must be less than 100
     millions
     */
    val amount: Amount,
    /**
     * Short description of the service or product, must be enabled by Unlimit manager to be used.
     * For Visa cards: maximum length 25 symbols, for MasterCard cards - 22 symbols.
     */
    val dynamicDescriptor: String?,
    /**

```

```

    * Note about the transaction that will not be displayed to customer_layout
    */
    val note: 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.
     */
    val transType: String?,
    /**
     * Array of items (in the shopping cart)
     */
    val items: List<Item>?,
    /**
     * Shipping Address
     */
    val shippingAddress: ShippingAddress?
) : Serializable

data class Amount(
    /**
     * The value of the total transaction amount in selected currency with dot as a decimal separator, must be
    less than 100 millions
     */
    val value: BigDecimal,
    /**
     * ISO 4217 currency code
     */
    val currency: Currency
) : Serializable

data class Item(
    /**
     * The count of product / service, provided to the customer_layout. Any positive number
     */
    val count: Int?,
    /**
     * The description of product / service, provided to the customer_layout
     */
    val description: String?,
    /**
     * The name of product / service, provided to the customer_layout
     */
    val name: String,
    /**
     * Price of product / service with dot as a decimal separator, must be less than a million
     */
    val price: Double?
) : Serializable

data class BillingAddress(
    /**
     * First line of the street address or equivalent local portion of the Cardholder billing address
    associated with the card used for this purchase. Should include street and house number
     * May include whitespaces, hyphens, apostrophes, commas, quotes, dots, slashes and semicolons.
     * Required (if available) unless market or regional mandate restricts sending this information.
     * 1-PA: Required unless market or regional mandate restricts sending this information.
     * 02-NPA: Required (if available) unless market or regional mandate restricts sending this information.
     */
    val 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.
     */
    val addrLine2: String?,
    /**
     * Billing city. May include whitespaces, hyphens, apostrophes, commas and dots

```

```

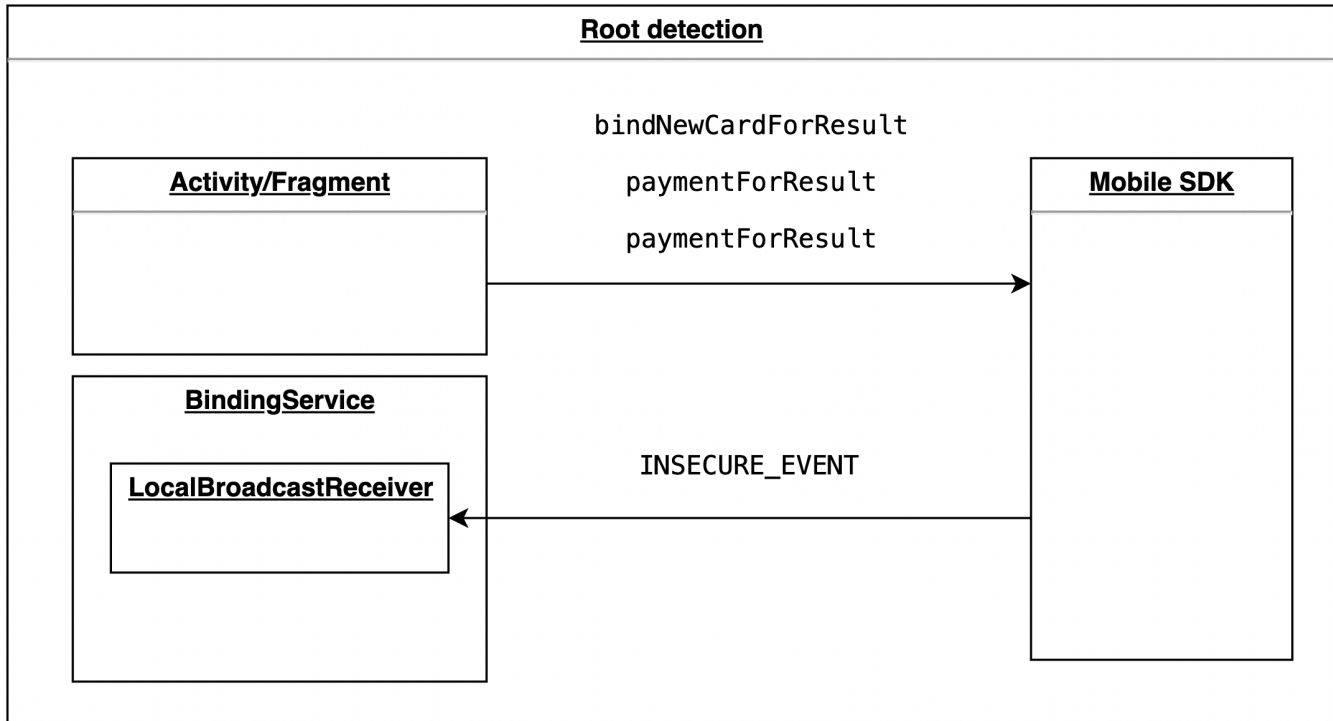
    */
    val city: String,
    /**
     * ISO 3166-1 code of billing country: 2 or 3 latin letters or numeric code
     */
    val country: String,
    /**
     * The state or province of the billing address associated with the card being used for this purchase.
     * It's recommended to send in following format: The country subdivision code defined in ISO 3166-2.
     * May include whitespaces, hyphens, apostrophes, commas and dots
     */
    val state: String?,
    /**
     * Billing postal code
     */
    val zip: String
) : Serializable

data class Customer(
    /**
     * 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.
     */
    val id: String,
    /**
     * Customer's e-mail address
     * Optional for wallets where setting in PM "May omit customer_layout email" is enabled
     */
    val email: String,
    /**
     * Customer's IPv4
     * Mandatory only for S2S mode
     */
    val ip: String? = null,
    /**
     * 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
     */
    val locale: String? = null,
    /**
     * 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_layout email" is enabled and
     customer_layout.email isn't presented in request
     */
    val phone: String? = null,
    /**
     * 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)
     */
    val homePhone: String? = null,
    /**
     * 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)
     */
    val workPhone: String? = null
) : Serializable

```

```
data class MerchantOrder(
    /**
     * Description of product/service being sold
     */
    val description: String,
    /**
     * Order ID used by the merchant's shopping cart
     */
    val id: String
) : Serializable
```

Root detection:



Close SDK:

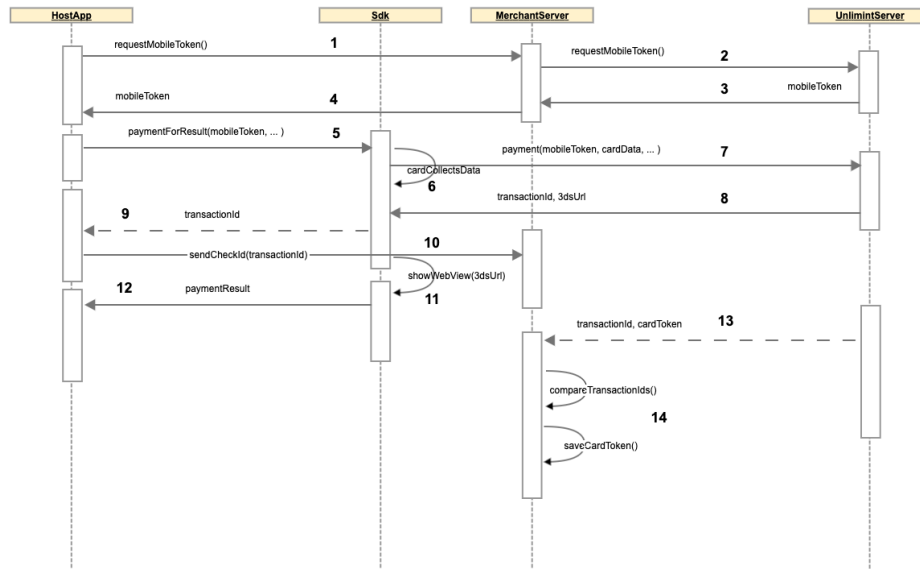
SDK can be closed via **close()** method (for example, if **INSECURE\_EVENT** is coming with **true**)

```
/**
 * Close our SDK Activity
 *
 * Invoke it, when you want urgently close SDK, for example, after catching security error
 * with Intent([MobileSdk.TransactionData.TRANSACTION_ACTION]) and [MobileSdk.TransactionData.
INSECURE_EVENT] key
 */
fun close(context: Context)
```



# Mobile API SDK interaction

Mobile SDK interaction sequence diagram (mobile payment)

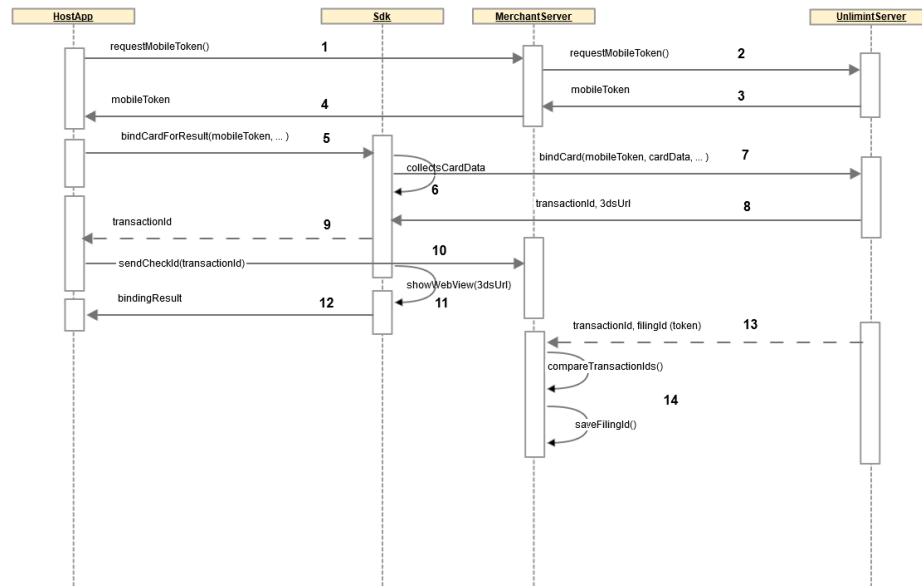


User scenario for mobile payment process:

Step	Requirement text
1	Host (merchant) mobile application sends request for getting mobile token to merchant backend (requestMobileToken)
2	<p>Merchant backend sends POST request (JSON) with valid access token for getting mobile token to API v3 endpoint <a href="https://cardpay.com/api/mobile/token">https://cardpay.com/api/mobile/token</a></p> <p>Header of request: valid access_token of merchant</p> <p>Parameters of request:</p> <p>request.id - request ID, should be unique for time period of 30 minutes  request.time - request attempt date and time up to milliseconds in ISO 8601 format (milliseconds is optional part)(example of format - yyyy-MM-dd'T'HH:mm:ss.SSS'Z')</p>
3	<p>API v3 returns response to merchant backend with requested mobile token:</p> <ul style="list-style-type: none"> <li>- mobile_token (string, unique identifier, 128 symbols)</li> <li>- expires ( date and time of mobile token expiration in ISO 8601 format, example of format - yyyy-MM-dd'T'HH:mm:ss.SSS'Z')</li> </ul> <p>Lifetime of mobile token is (see below):</p> <p>'lifetime of mobile token' &lt;= 5 min and &lt; 'Access_token' ('Bearer' token) life time.</p> <p>Granted LT of access token (for mobile token creation) should be less or equal 5 min but more or equeal than 4 min</p> <p>4 min &lt;= access token&lt;= 5 min</p>
4	Merchant backend sends mobile token to the host application
5	Host application calls paymentForResult(mobileToken) method of the mobile SDK
6	Customer fills in card data in card data form (in SDK)
7	Mobile SDK sends request with customer, card data, received mobile token (JSON) for making a payment to API v3 endpoint <a href="https://cardpay.com/api/mobile/*">https://cardpay.com/api/mobile/*</a> (here is presented masked endpoint)
8	API v3 sends a payment response with redirect URL and transaction id (for 3DS verification) to the mobile SDK
9	Mobile SDK returns transaction id to the host application, host application resend it to the merchant backend
10	Host application sends transaction id to the merchant server

11	Mobile SDK presents webview with 3dsUrl to the customer for 3-D Secure verification
12	3-D Secure verification procedure passes and customer redirects to success or decline url (paymentResult)
13	Unlimint API v3 sends callback to the merchant backend (with transaction id and card token (if it was requested by customer)
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:

Step	Requirement text
1	Host (merchant) mobile application sends request for getting mobile token to merchant backend (requestMobileToken)
2	Merchant backend sends POST request (JSON) with valid access token for getting mobile token to API v3 endpoint <a href="https://cardpay.com/api/mobile/token">https://cardpay.com/api/mobile/token</a>  Header of request: valid access_token of merchant.  Parameters of request:  request.id - request ID, should be unique for time period of 30 minutes request.time - request attempt date and time up to milliseconds in ISO 8601 format (milliseconds is optional part)(example of format - yyyy-MM-dd'T'HH:mm:ss.SSS'Z')
3	API v3 returns response to merchant backend with requested mobile token:  - mobile_token (string, unique identifier, 128 symbols)  - expires ( date and time of mobile token expiration in ISO 8601 format, example of format - yyyy-MM-dd'T'HH:mm:ss.SSS'Z')
4	Merchant backend sends mobile token to the host application
5	Host application calls bindCardForResult(mobileToken) method of the mobile SDK
6	Customer fills in card data in card data form (in SDK)
7	Mobile SDK sends request with customer, card data, received mobile token (JSON) for card binding to API v3 endpoint <a href="https://cardpay.com/api/mobile/*">https://cardpay.com/api/mobile/*</a> (here is presented masked endpoint)
8	API v3 sends a card binding response with redirect URL and transaction id (for 3DS verification) to the mobile SDK
9	Mobile SDK returns transaction id to the host application, host application resend it to the merchant backend
10	Host application sends transaction id to the merchant server
11	Mobile SDK presents webview with 3dsUrl to the customer for 3-D Secure verification

12	3-D Secure verification procedure passes and customer redirects to success or decline url (bindingResult)
13	Unlimint API v3 sends callback to the merchant backend (with transaction id and filing id)
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)



# Mobile SDK customisation

This page shows you the different ways that you can customize the UI in SDK

You can customize several UI styles and elements look:

- texts (hints and information), titles styles
- buttons style
- input layouts styles
- payment messages styles
- status bar, bottom area, background, text colors
- card binding elements customization
- title and regular fonts

## 1. Texts (hints and information), titles styles

You can customize hint style as mentioned in image

```
<attr name="cp_sdk_hint_style" format="reference" />
```

The image displays two screenshots of a mobile payment application. The top screenshot shows a card payment form titled 'TestTest' with a red close button. It includes input fields for 'Card number' (containing '4000 0000 0000 0002'), 'Expiry date' (containing '12 / 22'), and 'CVV2/CVC2' (containing three dots). A checkbox labeled 'Save for future use' is checked. The bottom screenshot shows a payment summary screen with 'Total 1,000.00 USD' and 'Order #some id'. It features a green 'Pay' button and logos for Mastercard, VISA, JCB, and MIR. Below the logos is a numeric keypad with letters for each digit (e.g., 1, 2 ABC, 3 DEF) and a green checkmark button.

Also you can customize information and title styles for each info text or each title

```
<attr name="cp_sdk_info_style" format="reference" />
```

10:59

16%

Merchant name



Card number

Expiry date

CVV2/CVC2

☒ Save for future use

Total 9,700.00 USD

Pay

Order #21513-216



VISA



MP

1

2

3

-

4

5

6

⌋

7

8

9

⌫

,

0

.

→

<attr name="cp\_sdk\_title\_style" format="reference" />

10:59

16%

Merchant name



Card number

Expiry date

CVV2/CVC2

☒ Save for future use

Total 9,700.00 USD

Order #21513-216

Pay




VISA



You can customize list title or list item style



```
<attr name="cp_sdk_list_title_style" format="reference" />  
<attr name="cp_sdk_list_item_style" format="reference" />
```

Very long merchant name  
with some numbers 15632

 Decline

#### Payment details

Order number	21513-216
Card number	2568
Card type	Visa
Total amount	1,000,000.00 EUR

  
  
[Back to the shop](#)




VISA



MIF

## Very long merchant name with some numbers 15632

 Decline

### Payment details

Order number	21513-216
Card number	... 2568
Card type	Visa
Total amount	1,000,000.00 EUR

Back to the shop



VISA



МИР


2. Buttons styles: you can customize style of buttons

```
<attr name="cp_sdk_button_style" format="reference" />
```

10:59

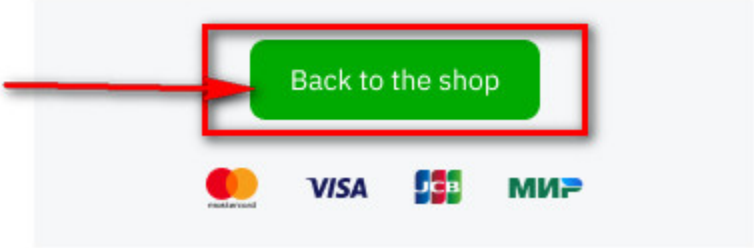
16%

Very long merchant name  
with some numbers 15632

 Decline

### Payment details

Order number	21513-216
Card number	... 2568
Card type	Visa
Total amount	1,000,000.00 EUR



Back to the shop



VISA



MIF

### 3. Customization of input layouts styles

You can customize input layout style, fonts, elements look

```
<attr name="cp_sdk_text_input_layout_style" format="reference" />
```

TestTest



Card number

4000 0000 0000 0002

Expiry date

12 / 22

CVV2/CVC2

...

☒ Save for future use

Total 1,000.00 USD

Order #some id

Pay



VISA



MIR

1 2 ABC 3 DEF

4 GHI 5 JKL 6 MNO

7 PQRS 8 TUV 9 WXYZ

0

✓

You can customize input layout edit text

```
<attr name="cp_sdk_text_input_edittext_style" format="reference" />
```

TestTest



Card number

4000 0000 0000 0002

Expiry date

12 / 22

CVV2/CVC2

...

☒ Save for future use

Total 1,000.00 USD

Order #some id

Pay



VISA



MIR

1 2 ABC 3 DEF

4 GHI 5 JKL 6 MNO

7 PQRS 8 TUV 9 WXYZ

0

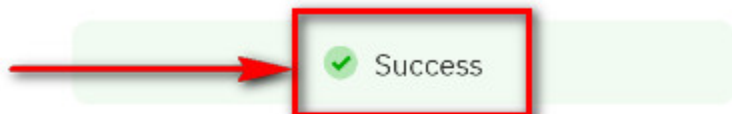
#### 4. Payment messages styles

You can customize payment messages success and decline styles

```
<attr name="cp_sdk_payment_message_success_style" format="reference" />
```



## Merchant name



## Payment details

Order id	111-2222
Card number	... 0002
Card type	Mastercard
Total amount	1,000.00 USD

[Back to the shop](#)



VISA



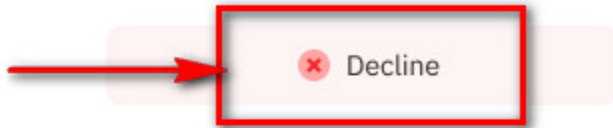
MWP

<attr name="cp\_sdk\_payment\_message\_decline\_style" format="reference" />

10:59

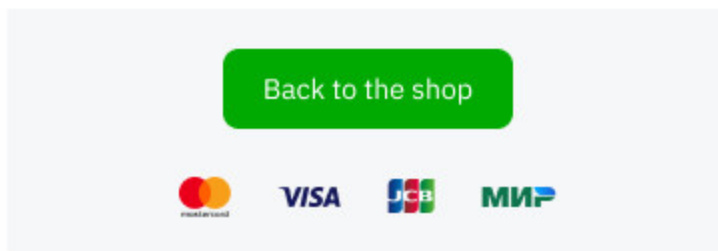
16%

Very long merchant name  
with some numbers 15632



### Payment details

Order number	21513-216
Card number	... 2568
Card type	Visa
Total amount	1,000,000.00 EUR



5. Status bar, bottom area, background, text colors

You can customize status bar color

```
<attr name="cp_sdk_status_bar_color" format="color" />
```



Merchant name

✓ Success

### Payment details

Order id	111-2222
Card number	... 0002
Card type	Mastercard
Total amount	1,000.00 USD

Back to the shop

Also you can customize bottom and backgroud area colors  
<attr name="cp\_sdk\_bottom\_area\_color" format="color" />

<attr name="cp\_sdk\_background" format="color" />

10:59

16%

## Add card



[Terms and conditions](#)

Confirm



VISA



МИР



1

2

3

-

4

5

6

⌋

<

7

8

9

⌫



,

0

.

→

And you can customize title and text colors in general  
<attr name="cp\_sdk\_title\_color" format="color" />

10:59

16%

Merchant name



Card number

Expiry date

CVV2/CVC2

☒ Save for future use

Total 9,700.00 USD

Order #21513-216

Pay



VISA



МИР

<attr name="cp\_sdk\_text\_color" format="color" />

10:59

16%

## Add card



Card number

Expiration date

CVV2/CVC2

[Terms and conditions](#)

Confirm



VISA



1

2

3

-

4

5

6

\_



7

8

9

✕



,

0

.

→

### 6. Card binding elements customization

```
<attr name="cp_sdk_binding_success_drawable" format="reference" />
```

10:59

16%

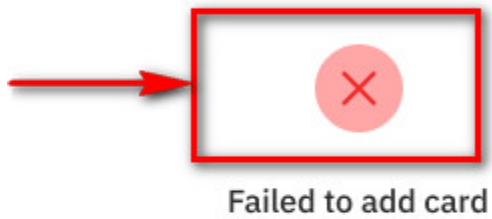


Back to the shop

<attr name="cp\_sdk\_binding\_fail\_drawable" format="reference" />

10:59

16%



Back to the shop

7. Title and regular fonts customization

```
<attr name="cp_sdk_title_font" format="reference" />
```



10:59

16%

Add card



Card number

Expiration date

CVV2/CVC2

[Terms and conditions](#)

Confirm



VISA



MIR



1

2

3

-



4

5

6

\_



7

8

9

✕

,

0

.

→

<attr name="cp\_sdk\_regular\_font" format="reference" />

10:59

16%

## Add card



Card number

Expiration date

CVV2/CVC2

[Terms and conditions](#)

Confirm



VISA



MAIF

1

2

3

-

4

5

6

⌋

7

8

9

⌫

,

0

.

→

### Change the look of each component

If you want to customize the components beyond what is specified in the theme, find the component that you want to customize and look at the **layout** and **values** directories of the Support SDK. These directories contain all of the source for the SDK layouts and styles. To find these directories:

1. Include our SDK in your **build.gradle** file
2. Build your project.
3. View the project structure in Project view in Android Studio. This means using the file tree pane on the left side and selecting **Project** from the top drop-down menu, which defaults to **Android**.

Project ▾

1: Project

Resource Manager

Build Variants

Gradle: androidx.viewpager:viewpager:1.0.0@aar

Gradle: artifacts:mobile-sdk:unspecified@jar

classes.jar library root

res library root

color

drawable

drawable-anydpi-v24

drawable-hdpi-v4

drawable-ldpi-v4

drawable-mdpi-v4

drawable-xhdpi-v4

drawable-xxhdpi-v4

font

layout

values

values.xml

values-v23

AndroidManifest.xml

Gradle: com.google.android.material:material:1.2.0@aar

Gradle: com.google.code.findbugs:jsr305:2.0.1@jar

Gradle: com.google.code.gson:gson:2.8.6@jar

Gradle: com.squareup.okhttp3:logging-interceptor:4.7.2@jar

Gradle: com.squareup.okhttp3:okhttp:4.7.2@jar

Gradle: com.squareup.okio:okio:2.6.0@jar

Gradle: com.squareup.retrofit2:converter-gson:2.4.0@jar

Gradle: com.squareup.retrofit2:retrofit:2.9.0@jar

Gradle: com.squareup.javawriter:2.1.1@jar

Gradle: io.mockk:mockk:1.10.0@jar

values.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <attr format="color" name="cp_sdk_background"/>
  <attr format="reference" name="cp_sdk_binding_fail_drawable"/>
  <attr format="reference" name="cp_sdk_binding_success_drawable"/>
  <attr format="color" name="cp_sdk_bottom_area_color"/>
  <attr format="reference" name="cp_sdk_button_style"/>
  <attr format="reference" name="cp_sdk_hint_style"/>
  <attr format="reference" name="cp_sdk_info_style"/>
  <attr format="reference" name="cp_sdk_list_item_style"/>
  <attr format="reference" name="cp_sdk_list_title_style"/>
  <attr format="reference" name="cp_sdk_payment_message_decline_style"/>
  <attr format="reference" name="cp_sdk_payment_message_success_style"/>
  <attr format="reference" name="cp_sdk_regular_font"/>
  <attr format="color" name="cp_sdk_status_bar_color"/>
  <attr format="color" name="cp_sdk_text_color"/>
  <attr format="reference" name="cp_sdk_text_input_edittext_style"/>
  <attr format="reference" name="cp_sdk_text_input_layout_style"/>
  <attr format="color" name="cp_sdk_title_color"/>
  <attr format="reference" name="cp_sdk_title_font"/>
  <attr format="reference" name="cp_sdk_title_style"/>
  <color name="color_area">#F5F7F8</color>
  <color name="color_background_light">#fff</color>
  <color name="color_button">#00AA00</color>
  <color name="color_button_disabled">#E2E6EA</color>
  <color name="color_button_pressed">#1C961C</color>
  <color name="color_button_text">#fff</color>
  <color name="color_button_text_disabled">#BCBFC1</color>
  <color name="color_decline">#FEF4F4</color>
  <color name="color_edittext_border_default">#E2E6EA</color>
  <color name="color_edittext_border_error">#EF2929</color>
  <color name="color_edittext_border_focused">#1476E9</color>
  <color name="color_edittext_success">#1476E9</color>
```