Checkout

Standard
Advanced
Integrate
Android SDK
iOS SDK
Customize
Card Decline Errors
Copy and Paste
Pay Later Offers
Pay with Venmo
Save Payment Methods
Alternative Payment Methods
Payment Methods

Checkout / Advanced / Android SDK

Integrate card payments in Android apps

SDK CURRENT ADVANCED Last updated: December 7th 2023, @ 10:43:54 am

Accept PayPal, credit, and debit card payments in a web or native experience using the PayPal Mobile Android SDK. Use customizable PayPal buttons with your custom checkout UI to align with your business branding. For more implementation details, see the PayPal GitHub repository.

Know before you code

You need a developer account to get sandbox credentials:

- PayPal uses REST API credentials which you can get from the developer dashboard.
- Client ID: Authenticates your account with PayPal and identifies an app in your sandbox.
- Client secret: Authorizes an app in your sandbox. Keep this secret safe and don't share it.

Read Get started with PayPal APIs for more information.

You need a combination of PayPal and third-party tools:

- Android SDK: Adds PayPal-supported payment methods for Android.
- Orders REST API: Create, update, retrieve, authorize, and capture orders.

Use Postman to explore and test PayPal APIs.

Run in Postman

1. Before you begin your integration

Check your account setup for advanced card payments

This integration requires a sandbox business account with the Advanced Credit and Debit Card Payments capability. Your account should automatically have this capability.

To confirm that Advanced Credit and Debit Card Payments are enabled for you, check your sandbox business account as follows:

1. Log into the **PayPal Developer Dashboard**, toggle **Sandbox**, and go to **Apps & Credentials**.
2. In **REST API apps**, select the name of your app.
3. Go to **Features** > **Accept payments**.
4. Select the **Advanced Credit and Debit Card Payments** checkbox and select **Save Changes**.

> **Note:** If you created a sandbox business account through sandbox.paypal.com, and the advanced credit and debit card payments status for the account is disabled, complete the sandbox onboarding steps.

Check 3D Secure requirements

Add 3D Secure to reduce the chance of fraud and improve the payment experience by authenticating a cardholder through their card issuer.

Visit our 3D Secure page to see if 3D Secure is required in your region and learn more about implementing 3DS in your app.

2. Integrate the SDK into your app

The PayPal Mobile SDK is available through Maven Central. Add the `mavenCentral` repository to the `build.gradle` file of your project root:

```
allprojects {
  repositories {
    mavenCentral()
  }
}
```

Snapshot builds

You can also use snapshot builds to test upcoming features before release. To include a snapshot build:

1. Add snapshots repository

1. Add snapshots repository

Add the snapshots repository to the `build.gradle` file of your project root.

```
allprojects {
  repositories {
    mavenCentral()
    maven {
      url 'https://oss.sonatype.org/content/repositories/snapshots/'
    }
  }
}
```

2. Add snapshot to dependencies

Then, add a snapshot build by adding `-SNAPSHOT` to the current dependency version. For example, if you want to add a snapshot build for `CardPayments`, add the following:

```
dependencies {
  implementation 'com.paypal.android:card-payments:CURRENT-VERSION-SNAPSHOT'
}
```

3. Payment integrations

Integrate 3 different types of payments using the PayPal Mobile SDK:

- **Card payments:** Add card fields that align with your branding.
- **PayPal native payments:** Launch a checkout page within your app, instead of a popup.
- **PayPal web payments:** A lighter integration that launches a checkout page in a browser within your app.

Card | Native payments | Web payments

Integrate with card payments

Build and customize the card fields to align with your branding.

1. Add card payments module to your app

Add the `card-payments` package dependency in your app's `build.gradle` file:

```
dependencies {
  implementation "com.paypal.android:card-payments:CURRENT-VERSION"
}
```

2. Create CardClient

A `CardClient` helps you attach a card to a payment.

In your Android app:

1. Use the `CLIENT_ID` to construct a `CoreConfig`.
2. Construct a `CardClient` using your `CoreConfig` object.

```
val config = CoreConfig("CLIENT_ID", environment = Environment.SANDBOX)

val cardClient = CardClient(config)
```

3. Get Order ID

On your server:

1. Create an `ORDER_ID` by using the Orders v2 API.
2. Pass your `ACCESS_TOKEN` in the `Authorization` header. To get an `ACCESS_TOKEN`, use the Authentication API.

> **Note:** This access token is only for the sandbox environment. When you're ready to go live, request a live access token by changing the request sandbox endpoint to https://api-m.paypal.com/v1/oauth2/token.

3. Pass the `intent`. You'll need to pass either `AUTHORIZE` or `CAPTURE` as the `intent` type. This type must match the

3. Pass the `intent`. You'll need to pass either `AUTHORIZE` or `CAPTURE` as the `intent` type. This type must match the `/authorize` or `/capture` endpoint you use to process your order.

Sample request | Sample response

```
curl --location --request POST 'https://api-m.sandbox.paypal.com/v2/checkout/orders/' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer ACCESS_TOKEN' \
  --data-raw '{
    "intent": "CAPTURE|AUTHORIZE",
    "purchase_units": [
      {
        "amount": {
          "currency_code": "USD",
          "value": "5.00"
        }
      }
    ]
  }'
```

When a buyer starts a payment, send the `ORDER_ID` from your server to your client app.

4. Create card request

A `CardRequest` object:

- Attaches a card to an `ORDER_ID`.
- Launches 3D Secure when a payment requires additional authentication.

1. Collect card payment details

Build a `card` object with the buyer's card details:

```
val card = Card(
  number = "4005519200000004",
  expirationMonth = "01",
  expirationYear = "2025",
  securityCode = "123",
  billingAddress = Address(
    streetAddress = "123 Main St.",
    extendedAddress = "Apt. 1A",
    locality = "Anytown",
    region = "CA",
    postalCode = "12345",
    countryCode = "US"
  )
)
```

Collecting a billing address can reduce the number of authentication challenges to customers.

2. Build CardRequest

Build a `CardRequest` with the `card` object and your `ORDER_ID`:

```
val cardRequest  = CardRequest(
  orderID = "ORDER_ID",
  card = card,
  returnUrl = "myapp://return_url", // custom URL scheme needs to be configured in AndroidManifest.xml
  sca = SCA.SCA_ALWAYS // default value is SCA.SCA_WHEN_REQUIRED
)
```

3D Secure is supported for all card payments to comply with the Second Payment Services Directive (PSD2). PSD2 is a European Union regulation that introduces Strong Customer Authentication (SCA) and other security requirements.

Select your SCA launch option type using the `sca` parameter in the `CardRequest` initializer:

- `SCA.SCA_WHEN_REQUIRED` launches an SCA challenge when applicable. This is enabled by default.
- `SCA.SCA_ALWAYS` requires an SCA challenge for all card transactions.

3. Set up your app for browser switching

The `sca` challenge launches in a browser within your application. Your app needs to handle the browser switch between the `sca` challenge and the checkout page. Set up a return URL that returns to your app from the browser.

4. Create a return URL

Provide a `returnUrl` so the browser returns to your application after the `sca` challenge finishes.

The `returnUrl` should have the following format: