Integrate card payments with JS SDK for direct merchants

PayPal Checkout plus customized card fields

SDK CURRENT ADVANCED Last updated: December 15th 2023, @ 12:11:35 pm

> ⚠ **Important:** This is version 2 of the JavaScript SDK integration guide for direct merchants. Version 1 is a legacy integration.

How it works

Advanced Checkout lets you offer the PayPal payment button and custom credit and debit card fields. This guide covers integrating the PayPal button, card payments, and customized credit card fields that handle input from the following payment fields:

- Card number
- CVV
- Expiration date
- Optional: Cardholder name

You can find more ways to extend your integration in our customization doc.

After you integrate advanced Checkout, you can also offer options like Pay Later, Venmo, and other payment methods with some additional configuration.

Visit the GitHub repo to download a sample integration.

Which integration to use

There are 2 versions of card payment integrations for direct merchants using the JavaScript SDK:

- **Recommended:** Version 2 uses the PayPal-hosted `CardFields` component to accept and save cards without handling card information. PayPal handles all security and compliance issues associated with processing cards. The `CardFields` component is the recommended integration method and receives ongoing enhancements and improvements.
- Version 1 is a legacy integration that uses the `HostedFields` component. This integration is no longer under active development and won't receive further updates.

Other elements of the JavaScript SDK integration for direct merchants remain the same.

Get up and running in GitHub Codespaces

GitHub Codespaces are cloud-based development containers where you can code and test your PayPal integrations. Learn more

Open in Codespaces

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.
    - Dashboard
    - Read the guide
- You need the following PayPal tools for this integration:
  - JavaScript SDK: Adds PayPal-supported payment methods.
  - Orders v2 REST API: Create, update, retrieve, authorize, and capture orders.
  - You can 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 risk 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 3D Secure in your app.

2. Initialize JavaScript SDK

Add the JavaScript SDK to your web page and include the following:

- Your app's client ID.
- A `div` to render the PayPal buttons.
- A `div` to render each of the card fields.

In the included JavaScript file, there are reference routes on the server that you'll add in the next step.

> **Tip:** Test your PayPal button transactions in the developer dashboard by creating sandbox accounts.

3. Add PayPal buttons and card fields

This integration includes a full-stack Node.js example. The `/client/checkout.ejs`, `/public/app.js`, and `/server/server.js` file samples show how to render the PayPal buttons and the card fields component:

- Use PayPal buttons to process PayPal payments.
- Use card fields to process card payments.

You'll need to:

- Save the `checkout.ejs` file in a folder named `/client`.
- Save the `app.js` file in a folder named `/public`.
- Save the `server.js` file in a folder named `/server`.

/client/checkout.ejs | /public/app.js | /server/server.js

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <!-- To be replaced with your own stylesheet -->
    <link
      rel="stylesheet"
      type="text/css"
      href="https://www.paypalobjects.com/webstatic/en_US/developer/docs/css/cardfields.css"
    />
    <!-- Express fills in the clientId and clientToken variables -->
    <script src="https://www.paypal.com/sdk/js?components=buttons,card-fields&client-id=<%= clientId %>"></script
  </head>
  <body>
    <div id="paypal-button-container" class="paypal-button-container"></div>
    <div id="checkout-form">
      <div id="card-name-field-container"></div>
      <div id="card-number-field-container"></div>
      <div id="card-expiry-field-container"></div>
      <div id="card-cvv-field-container"></div>
      <button id="card-field-submit-button" type="button">
        Pay now with Card Fields
      </button>
    </div>
    <script src="./public/app.js"></script>
  </body>
</html>
```

Modify the code

This section explains how to customize the PayPal buttons and card fields for your integration.

PayPal buttons

1. Copy a complete set of sample integration code from the GitHub repo.
2. The CSS file in the `head` section is a sample for demo purposes. Instead, you should use styles that align with your brand using the CSS properties supported by this integration.
3. Optional: Customize JavaScript configurations, such as `currency` and `intent`.
4. Optional: Change the layout, width, height, and outer styling of the PayPal buttons, such as `border`, `box-shadow`, and `background`.

Card fields

1. Copy and paste both examples of card field style objects into your existing `/client/checkout.ejs` and `/public/app.js` files.
2. Include the required card form elements: `card number`, `security code`, and `expiration date`. To learn about the available card form elements, see Card fields.
3. A complete set of sample integration code is available from the GitHub repo.
4. Optional: Change the layout, width, height and outer styling of the card fields, such as `border`, `box-shadow`, and `background`. You can modify the elements you supply as containers.

4. Call Orders API for PayPal buttons and card fields

Create API endpoints on your server that communicate with the Orders v2 API to create an order and capture payment for an order.

> ⚠ **Important:** If you process payments that require Strong Customer Authentication, you need to provide additional context with payment indicators.

Server-side example (Node.js)

The `paypal-api.js` and `/server/server.js` code samples show how to integrate back-end code for advanced payments. The code adds routes to an Express server to create orders and capture payments using the Orders v2 API.

You'll need to:

- Save the `paypal-api.js` file in your app's main folder.
- Save the `server.js` file in a folder named `/server`.

/server/server.js | paypal-api.js

```
import * as paypal from "./paypal-api.js";

// create order
app.post("/api/orders", async (req, res) => {
  const order = await paypal.createOrder(req.body.paymentSource);
  res.json(order);
});
```

5. Capture order

Set up your server-side code to capture the order when a payer uses a credit or debit card. The `paypal-api.js` and `/server/server.js` files show how using server-side code prevents exposing your access token on the client side.

You'll need to:

- Save the `server.js` file in a folder named `/server`.
- Save the `paypal-api.js` file in your app's main folder.

/server/server.js | paypal-api.js

```
// capture payment
app.post("/api/orders/:orderID/capture", async (req, res) => {
  const { orderID } = req.params;
  const captureData = await paypal.capturePayment(orderID);
  res.json(captureData);
});
```