PayPal - REST API V2

Please find a dedicated section to help you create your Production or Sandbox Paypal account : Create PayPal Production and sandbox accounts - REST API V2

PayPal is one of the world's most popular digital wallets, enabling users to pay quickly and securely online using their stored payment methods. With over 400 million users globally, PayPal supports transactions in multiple currencies and provides a trusted checkout experience across devices and platforms.

supports PayPal through all major integration types – giving merchants the flexibility to design their ideal checkout experience while benefiting from PayPal's global reach.

Get started

Before integrating PayPal with , merchants must first complete the necessary setup within their PayPal account. These steps are essential for linking your PayPal merchant profile with .

Steps for PayPal Sandbox (Test) Environment

1. **Create PayPal developer account** Create Paypal developer account if you do not have one.
2. **Create a PayPal Sandbox Account**
   Follow PayPal's guide to create a sandbox business account.
3. **Grant permissions to**
   Fetch the Email ID and password of your sandbox business account to login and Authorize to process transactions on your behalf.
4. **Retrieve Your Payer ID (Account ID)**
   You can find the Payer ID also known as Account ID in your developer account under the specific sandbox business account.
5. **Send Payer ID to**
   Once you have your Account ID, contact and request activation of PayPal in the Test environment.

Steps for PayPal Live Environment

1. Signup to create a live PayPal business account
2. Activate your business account.
3. Grant permissions to by logging in with your account credentials and follow the steps.
4. Retrieve the live Payer ID also known as Account ID from your business account
5. Contact to enable PayPal in the Live environment and provide the Payer ID

Integration options

supports PayPal across all three integration types:

Hosted Payment Page

Use Create checkout session to redirect customers to a Hosted Payment Page where PayPal will appear as a payment option.

Hosted forms

For a redirect-style integration, call Create payment with:

```
{
  ...
  "paymentMethods": {
    "type": "PAYPAL",
    "integrationType": "HOSTED"
  }
}
```

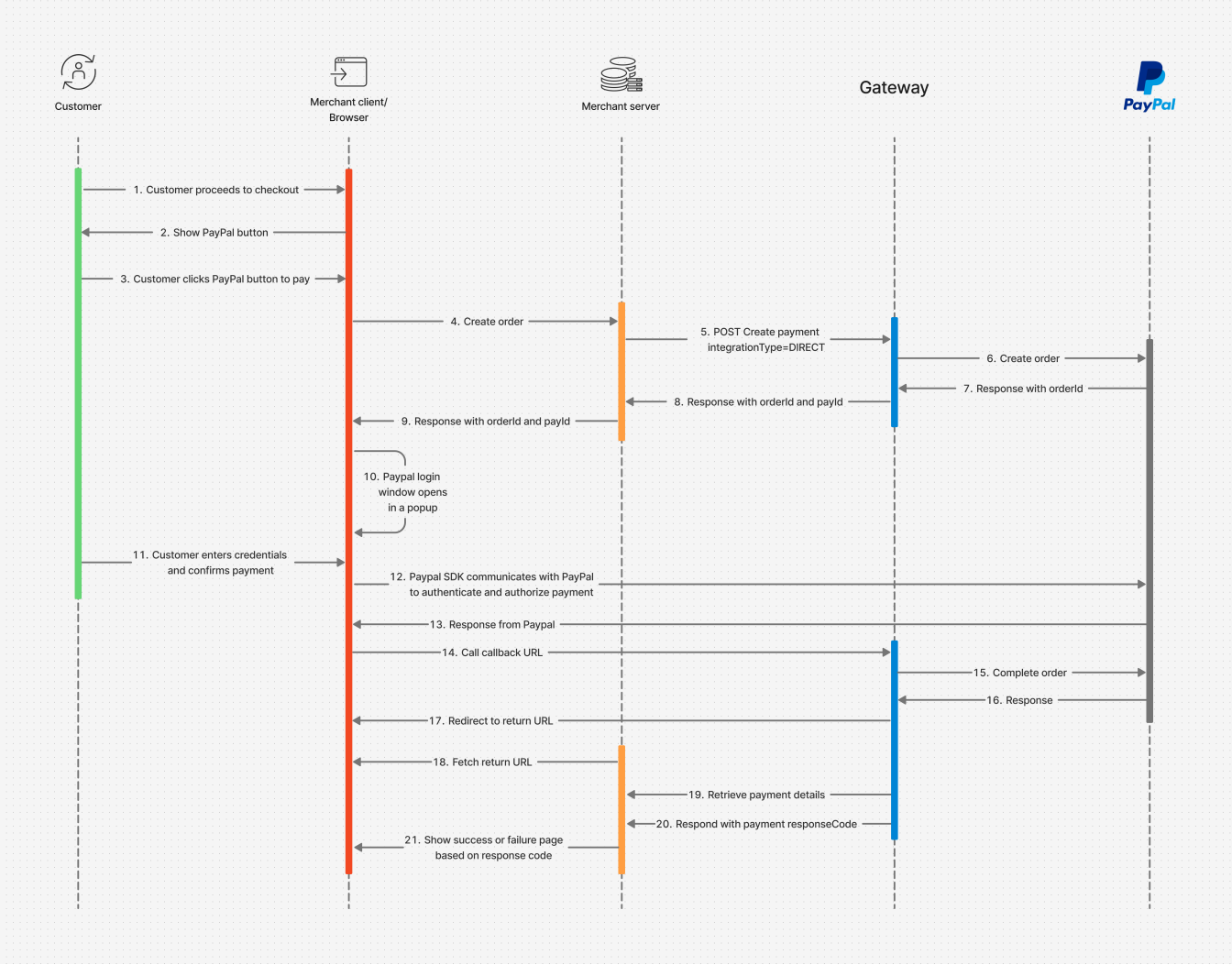
.

Customers will be redirected to PayPal to complete the payment.

Direct integration – PayPal Smart Button

Unlike typical Direct integration, PayPal Smart Button follows a custom integration flow. Refer to the detailed instructions below to implement it correctly.

Process flow



1. Customer proceeds to checkout.
2. Your frontend loads the PayPal Smart Button. See below sample code for your reference to embed the Paypal's Smart Button on your checkout page.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
</head>
<body>
  <!-- Set up a container element for the button -->
  <div id="paypal-button-container"></div>

  <!-- Include the PayPal JavaScript SDK -->
  <!-- Replace <PayerID> with your value, available under your PayPal account -->
  <!-- Replace <Currency> with the currency which should be used, e.g., EUR -->
  <!-- When going live, replace client-id with the live client-id provided by Computop -->
  <!-- When going live, replace data-partner-attribution-id with the live data-partner-attribution-id
```

```
provided by Computop -->
  <script type="text/javascript" src="https://www.paypal.com/sdk/js?client-
id=ARCsDK7xBFxa5pnGxk8qvB0STB07fyi_yHDRrb5al6gxahj73Pxg9X2l7onP9J2IN-LqcVJojys94FLK&merchant-
id=<PayerID>¤cy=<Currency>&disable-funding=giropay,sofort,sepa,card&intent=capture" data-partner-
attribution-id="Computop_PSP_PCP_Test"></script>
  <!-- Initialize and show PayPal button -->
  <script type="text/javascript">
    let mid = "YOUR MERCHANTID";
    let orderId = null;
    let payId = null;

    // Render the PayPal button into #paypal-button-container
    paypal.Buttons({
      createOrder: function (data, actions) {
        // This is a sample endpoint path. Replace '/api/create-paypal-order' with your own backend
URL.
        // When the PayPal button is clicked, this frontend function calls your backend.
        // Your backend should then:
        // 1. Call Computop Paygate to initiate a PayPal transaction,
        // 2. Retrieve and return the OrderId and PayId generated by Computop,
        // 3. Respond to this frontend with that data.
        return fetch('/api/create-paypal-order', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ merchantId: mid })
        })
        .then(res => res.json())
        .then(response => {
          orderId = response.orderId;
          payId = response.payId;
          return orderId; // Return this to PayPal
        });
      },
      onApprove: function (data, actions) {
        buildAndSubmitForm();
      },
      onCancel: function (data, actions) {
        buildAndSubmitForm("cancel");
      },
      onError: function (data, actions) {
        buildAndSubmitForm();
      }
    }).render('#paypal-button-container');

    // Call cbPayPal.aspx to continue sequence. The endpoint will then forward to urls.return from
the REST call
    function buildAndSubmitForm(userAction) {
      var requestData = "MerchantId=" + mid + "&PayId=" + payId + "&OrderId=" + orderId;

      // Build an invisible form and submit it to prevent preflight requests
      const form = document.createElement('form');
      form.style.display = 'none';
      form.method = 'POST';
      form.action = "https://www.computop-paygate.com/cbPayPal.aspx?rd=" + window.btoa(requestData) +
"&token=" + orderId;

      if (userAction === "cancel") {
        form.action += "&ua=cancel";
      }

      document.body.appendChild(form);
      form.submit();
    }
  </script>
</body>
</html>
```

Additional configuration options

Update the below line of code as per your additional configuration needs

```
<script type="text/javascript" src="https://www.paypal.com/sdk/js?client-
id=ARCsDK7xBFxa5pnGxk8qvB0STB07fyi_yHDRrb5al6gxahj73Pxg9X2l7onP9J2IN-LqcVJojys94FLK&merchant-
id=<PayerID>¤cy=<Currency>&disable-funding=giropay,sofort,sepa,card&intent=capture" data-partner-
attribution-id="Computop_PSP_PCP_Test"></script>
```

- **Funding sources**: Customize allowed or disabled payment sources via `disable-funding` or `enable-funding` parameters.
- **Pay Later support**: Add `enable-funding=paylater` to show PayPal Pay Later.
- **Intent parameter**: Must align with capture strategy:
  - `intent=capture` for auto capture
  - `intent=authorize` for manual capture

3. Customer clicks PayPal button to pay.
4. Your frontend javascript code initiates the createOrder function to call your server.
5. Your server makes a Create payment call with:

```
{
  ...
  "paymentMethods": {
    "type": "PAYPAL",
    "integrationType": "DIRECT"
  }
}
```

6. initiates a Create order call with PayPal.
7. PayPal returns `orderId` in the response.
8. returns `orderId` along with the `payId` of the transaction.
9. Your server returns `orderId` and `payId` to your frontend.
10. Javascript code on your frontend initializes PayPal SDK with `orderId` to open the PayPal login window in a popup.
11. Customer enters credentials and confirms payment.
12. PayPal SDK communicates with PayPal to authenticate and authorize the payment.
13. Paypal responds to the request.
14. Javascript code on your frontend calls 's callback URL.
15. intiates a Complete order call to PayPal.
16. PayPal responds to the request.
17. Customer is redirected to the return URL (urls.return) that was submitted by you in step 5. The return URL is suffixed with `payId` in the query parameters.
18. The browser triggers an HTTP GET request to the return URL on your server, including the `payId` in the query string. Your server uses this p `ayId` to identify the payment session and proceed with retrieving the payment result.
19. Your server makes a Retrieve payment details by Payment ID call to retrieve the responseCode of the payment.
20. responds with the responseCode of the payment along with other parameters.
21. Your server returns a success or a failure page based on the responseCode of the payment.

Express checkout

PayPal Express checkout is a streamlined, customer-friendly payment experience that enables quick purchasing by allowing users to leverage their PayPal-stored address and payment details. This flow reduces friction by pre-filling shipping details, making it ideal for repeat customers or those who want a fast, no-hassle checkout experience.

Express checkout is available on both Hosted Forms and Direct integration (PayPal Smart Button).

To support PayPal Express checkout, submit the following in your initial Create payment request:

```
{
  ...
  "paymentMethods": {
    "type": "PAYPAL",
    "integrationType": "DIRECT", // HOSTED - for Hosted forms
    "payPal": {
      "expressCheckout": true
    }
  }
}
```

The process flow for Express checkout is an extension to standard flow:

- Once the initial payment request completes successfully responds with status of the payment as `AUTHORIZATION_REQUEST`. Based on this you should show a confirmation page to the customer.

- Here customer can perform some actions like updating their address. These need to be communicated to PayPal in a subsequent Update payment details call with `eventToken=UPDATE_ORDER_DETAILS`
- It is possible to update the amount when finalizing the transaction. However, there are some thresholds that you should not exceed. You can contact your PayPal account manager for more details on this topic. The response comes with the status `OK` if successful.

Reverse payment

Reverse an authorized but not yet captured PayPal transaction by calling Reverse payment with the original `payId`.

Capture payment

You can capture a PayPal payment manually using the Capture payment endpoint.

Refund payment

To issue a refund for a captured Paypal payment, use the Refund payment endpoint.