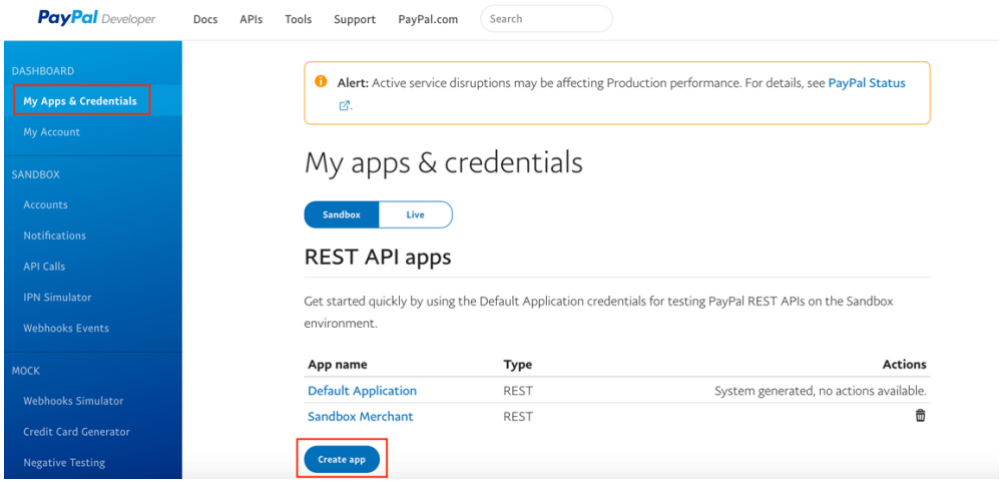
PayPal Advance Credit Card Integration Guide

This document will show the details of how to integrate the Advance Credit Card (ACC) payment and 3Ds. ACC generally supports the latest SDK (Restful API) to process payment, you may need to upgrade the API if the merchant already integrated PayPal products with the Classic API (NVP/SOAP). You can also take the official documentation as the reference.

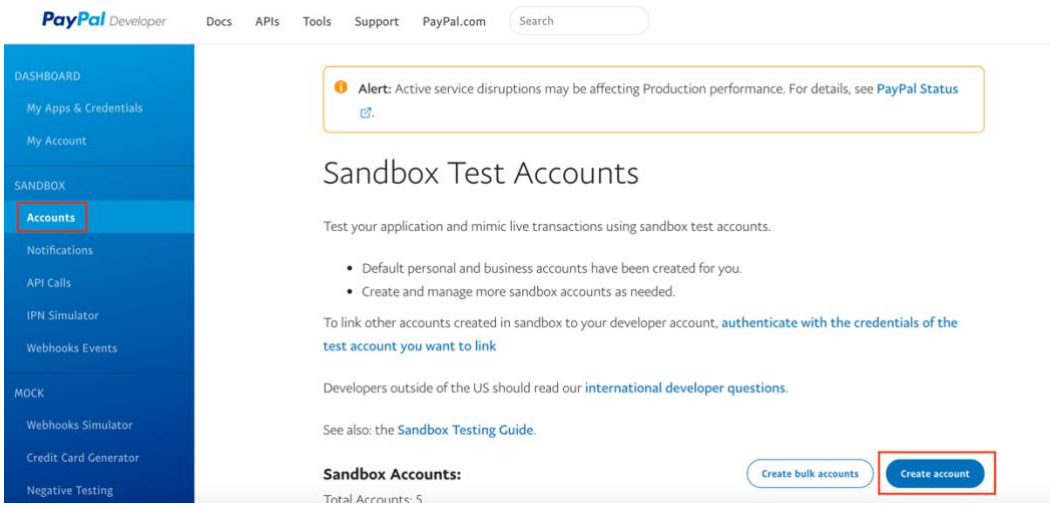
Account Preparation

Before the integration, you may need to fulfil the following criteria:

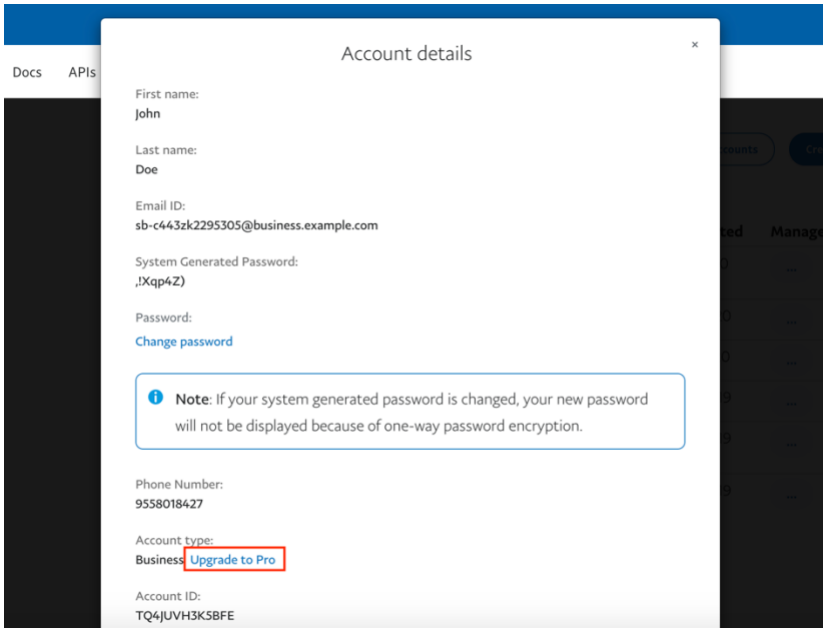
1. Create a new application from the sandbox account:
   Go to the PayPal developer website and login the your PayPal sandbox account (Hong Kong). Go to "My Apps & Credentials" from the left column, create a new application by clicking "Create App" and enter your custom app name.



2. Enable the Payment-Pro feature from the business account:
   Go to "Accounts" from the left column, create a new business account by clicking "Create Account" and select the correct country.



After created the account, edit the account information and upgrade the account type to Business-Pro.



ACC Integration

In this solution, it requires the customer information including card number, card expiry date and CVV to finish the payment process. You can customize the billing address information from the checkout page and input the data.

1. Get the access token from the API credentials:
   You may need to get the access token by using the client ID and secret from your application created in your sandbox account. This document will show you the details of API call and how to set up in the Postman.

```bash
curl -v POST https://api.sandbox.paypal.com/v1/oauth2/token \
  -H "Accept: application/json" \
  -H "Accept-Language: en_US" \
  -u "CLIENT_ID:SECRET" \
  -d "grant_type=client_credentials"
```

API Sandbox URL: https://api.sandbox.paypal.com/v1/oauth2/token
API Production URL: https://api.paypal.com/v1/oauth2/token

You can also follow the Postman settings and get the “access_token” from the result.

2. Generate the client data token:
   After retrieving the access token, a client data token is needed from the JavaScript SDK. The following shows the details of the API call:

```
curl -X POST https://api.sandbox.paypal.com/v1/identity/generate-token \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <ACCESS-TOKEN>' \
-H 'Accept-Language: en_US' \
-d '{
 "customer_id": "customer_1234"
}'
```

   API Sandbox URL: https://api.sandbox.paypal.com/v1/identity/generate-token
   API Production URL: https://api.paypal.com/v1/identity/generate-token

   You can customize your "customer_id", but make sure that it is the unique alphanumeric value and it does not change when the user is the same.

3. Add the PayPal Javascript SDK and card form in the website
   You can add the PayPal JavaScript SDK and the card form to the checkout page of the website. The following code shows the example of how the card form works in HTML format, and here is the link of "cardfields.css".

   Before rendering the page, you may also need to create a new order from the server side and retrieve the order ID. After that, fill the order ID in the SDK library. If you want to know how to create a new order, please refer to here.

   For your reference, you can try the live demo here.

```
<html>
<head>

 <meta charset="utf-8"/>
 <meta name="viewport" content="width=device-width, initial-scale=1"> <!-- Optimal rendering on mobile devices. -->
 <meta http-equiv="X-UA-Compatible" content="IE=edge" /> <!-- Optimal Internet Explorer compatibility -->
 <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
 <link rel="stylesheet" type="text/css" href="cardfields.css"/>

</head>
<body>

<!-- JavaScript SDK -->
 <script src="https://www.paypal.com/sdk/js?components=hosted-fields,buttons&client-id=<YOUR-CLIENT-ID>" data-client-token="<YOUR-CLIENT-TOKEN>"></script>
```

```
  <!-- Advanced credit and debit card payments form -->
  <div class='card_container'>
   <form id='my-sample-form'>
    <label for='card-number'>Card Number</label><div id='card-number' class='card_field'></div>
    <div>
     <label for='expiration-date'>Expiration Date</label><div id='expiration-date' class='card_field'></div>
    </div>
    <div>
     <label for='cvv'>CVV</label><div id='cvv' class='card_field'></div>
    </div>
    <label for='card-holder-name'>Name on Card</label><input type='text' id='card-holder-name' name='card-
holder-name' autocomplete='off' placeholder='card holder name'/>
    <div>
     <label for='card-billing-address-street'>Billing Address</label><input type='text' id='card-billing-address-
street' name='card-billing-address-street' autocomplete='off' placeholder='street address'/>
    </div>
    <div>
     <label for='card-billing-address-unit'>&amp;nbsp;</label><input type='text' id='card-billing-address-unit'
name='card-billing-address-unit' autocomplete='off' placeholder='unit'/>
    </div>
    <div>
     <input type='text' id='card-billing-address-city' name='card-billing-address-city' autocomplete='off'
placeholder='city'/>
    </div>
    <div>
     <input type='text' id='card-billing-address-state' name='card-billing-address-state' autocomplete='off'
placeholder='state'/>
    </div>
    <div>
     <input type='text' id='card-billing-address-zip' name='card-billing-address-zip' autocomplete='off'
placeholder='zip / postal code'/>
    </div>
    <div>
     <input type='text' id='card-billing-address-country' name='card-billing-address-country' autocomplete='off'
placeholder='country code' />
    </div>
    <br><br>
    <button value='submit' id='submit' class='btn'>Pay</button>
   </form>
  </div>

  <!-- Implementation -->
  <script>
   // Eligibility check for advanced credit and debit card payments
   if (paypal.HostedFields.isEligible()) {
    paypal.HostedFields.render({
     createOrder: function () {return "order-ID";}, // replace order-ID with the order ID
     styles: {
      'input': {
       'font-size': '17px',
       'font-family': 'helvetica, tahoma, calibri, sans-serif',
       'color': '#3a3a3a'
      },
      ':focus': {
       'color': 'black'
```

```
        }
       },
       fields: {
        number: {
         selector: '#card-number',
         placeholder: 'card number'
        },
        cvv: {
         selector: '#cvv',
         placeholder: 'card security number'
        },
        expirationDate: {
         selector: '#expiration-date',
         placeholder: 'mm/yy'
        }
       }
     }).then(function (hf) {
       $('#my-sample-form').submit(function (event) {
        event.preventDefault();
        hf.submit({
         // Cardholder Name
         cardholderName: document.getElementById('card-holder-name').value,
         // Billing Address
         billingAddress: {
          streetAddress: document.getElementById('card-billing-address-street').value,     // address_line_1 -
street
          extendedAddress: document.getElementById('card-billing-address-unit').value,      // address_line_2 -
unit
          region: document.getElementById('card-billing-address-state').value,          // admin_area_1 - state
          locality: document.getElementById('card-billing-address-city').value,         // admin_area_2 - town / city
          postalCode: document.getElementById('card-billing-address-zip').value,          // postal_code -
postal_code
          countryCodeAlpha2: document.getElementById('card-billing-address-country').value   // country_code -
country
         }
        }).then((result) => {
           //TODO
           // Fetch the API call from backend to capture the order
        });
       });
     });
    }
    else {
     $('#my-sample-form').hide();  // hides the advanced credit and debit card payments fields if merchant isn't
eligible
    }
  </script>

  </body>
  </html>
```

4. Capture the orders from the server-side API calls
   After the form is submitted, you can pass the order ID to the backend and capture the payment order. Once after the payment is captured, you can return the status to

the checkout page through the API call.

The following code shows the example of how to capture the order:

```
// Note: This is example code. Each server platform and programming language has a different way of
handling requests, making HTTP API calls, and serving responses to the browser.

// 1. Set up your server to make calls to PayPal

// 1a. Add your client ID and secret
PAYPAL_CLIENT = 'PAYPAL_SANDBOX_CLIENT';
PAYPAL_SECRET = 'PAYPAL_SANDBOX_SECRET';

// 1b. Point your server to the PayPal API
PAYPAL_OAUTH_API = 'https://api.sandbox.paypal.com/v1/oauth2/token/';
PAYPAL_ORDER_API = 'https://api.sandbox.paypal.com/v2/checkout/orders/';

// 1c. Get an access token from the PayPal API
basicAuth = base64encode(`${ PAYPAL_CLIENT }:${ PAYPAL_SECRET }`);
auth = http.post(PAYPAL_OAUTH_API {
  headers: {
    Accept:       `application/json`,
    Authorization: `Basic ${ basicAuth }`
  },
  data: `grant_type=client_credentials`
});

// 2. Set up your server to receive a call from the client
function handleRequest(request, response) {

  // 2a. Get the order ID from the request body
  orderID = request.body.orderID;

  // 3. Call PayPal to capture the order
  capture = http.post(PAYPAL_ORDER_API + orderID + '/capture', {
    headers: {
      Accept:       `application/json`,
      Authorization: `Bearer ${ auth.access_token }`
    }
  });

  // 4. Save the capture ID to your database
  if (!capture.error) {
    captureID = capture.purchase_units[0]
       .payments.captures[0].id;
    database.saveCaptureID(captureID);
  }

  // 5. Handle any errors from the call
  if (capture.error) {
    console.error(capture.error);
    return response.send(500);
  }
```

```
  // 6. Return a successful response to the client
  response.send(200);
}
```

From the capture API result, please make sure that you get the correct payment status:

```
"result": {
      "id": "6N283131LV7966709",
      "intent": "CAPTURE",
      "purchase_units": [
       {
        "reference_id": "666a97cddb3245789e23d3299ad9e7a0",
        "amount": {
         "currency_code": "USD",
         "value": "2.00"
        },
        "payee": {
         "email_address": "xxx@xxx.com”,
         "merchant_id": "P2FFKES9RYGKJ"
        },
        "description": "xxxxxxxx",
        "custom_id": "xxxxxx",
        "soft_descriptor": "PAYPAL *PEPPER HK",
        "payments": {
         "captures": [
          {
           "id": "0CX03331WX398261F",
           "status": "COMPLETED",
           "amount": {
            "currency_code": "USD",
            "value": "2.00"
           },
           "final_capture": true,
           "seller_protection": {
            "status": "ELIGIBLE",
            "dispute_categories": [
             "ITEM_NOT_RECEIVED",
             "UNAUTHORIZED_TRANSACTION"
            ]
           },
           "custom_id": "xxxxxx",
           "links": [
            {
             "href": "https://api.paypal.com/v2/payments/captures/0CX03331WX398261F",
             "rel": "self",
             "method": "GET"
            },


……
```

3D-S Integration

3D Secure is to authenticate the card holders from the card issuer and improve transaction performance by preventing frauds. It can shift liability for fraudulent chargebacks from the merchant to the card issuer when the 3D Secure authentication is made. You can check the details from the documentation, and the following steps will show you how to do the integration:

1. Pass parameter to enable 3D Secure:
   Add the parameter “data-enable-3ds” to the SDK script tag:

```HTML
<script src="https://paypal.com/sdk/js?client-id=YOUR_CLIENT_ID" data-client-token="eyJicmFpbnRyZWUiOnsiYXV0aG9ya==" data-enable-3ds></script>
```

2. Include a contingency for the 3D Secure:
   Add the following code to the checkout page:

```
<div id="payments-sdk__contingency-lightbox"></div>
```

3. Update the ACC code:
   You may need to add the “contingency” parameter to the submit the 3D Secure authentication.

```
// Check eligibility for advanced credit and debit card payments
if (paypal.HostedFields.isEligible()) {
  // render the card fields
  paypal.HostedFields.render({

    // sample function to return the order ID
    createOrder: () => {
      // add logic to return an order ID from your server
    },
    fields: {
      number: {
        selector: '#card-number',
        placeholder: 'card number'
      },
      cvv: {
        selector: '#cvv',
        placeholder: 'CVV',
      },
      expirationDate: {
        selector: '#expiration-date',
        placeholder: 'mm/yyyy'
```

```
        }
      }
    }).then(function (hf) {

      document.querySelector('#my-sample-form').addEventListener('submit', (event) => {
        event.preventDefault();

        hf.submit({

          // Trigger 3D Secure authentication
          contingencies: ['3D_SECURE']

        }).then(function (payload) {

          /** sample payload
          * {
          * "orderId": "0BS14434UR665304G",
          * "liabilityShifted":      true,
          * "authenticationStatus":   "YES",
          * "authenticationReason":   "SUCCESSFUL"
          * }
          * possible value:
          * liabilityShifted - true, false, undefined
          * authenticationStatus - "YES", "NO", "ERROR", undefined
          * authenticationReason - "SUCCESSFUL", "ATTEMPTED", "BYPASSED", "UNAVAILABLE", "ERROR",
"CARD_INELIGIBLE", "SKIPPED_BY_BUYER", "FAILURE", undefined
          */

          // Needed only when 3D Secure contingency applied

          if (payload.liabilityShifted === undefined) {
            // Handle no 3D Secure contingency passed scenario
          }

          if (payload.liabilityShifted) {
            // Handle Buyer confirmed 3D Secure successfully
          }

          if (payload.authenticationReason === 'SKIPPED_BY_BUYER') {
            // Handle buyer skipped 3D Secure use-case
          }
        });
      });
    });
}
else {
  /*
   * Handle experience when advanced credit and debit card payments
   * card fields are not eligible
   */
}
```

From the result, there is a few status in the parameters of liabilityShifted, authenticationStatus, and authenticationReason, which represents different scenario:

Liability shifted	Authentication Status	Authentication Reason	Description	Next Steps
undefined	undefined	undefined	You have not required 3D Secure for the buyer or the card network did not require a 3D Secure	You can continue with authorization and assume liability. If you prefer not to assume liability, ask the buyer for another card
true	YES	SUCCESSFUL	Buyer successfully authenticated using 3D secure	Buyer authenticated with 3DS and you can continue with the authorization
false	ERROR	ERROR	An error occurred with the 3D Secure authentication system	Prompt the buyer to re-authenticate or request for another form of payment
false	NO	SKIPPED_BY_BUYER	Buyer was presented the 3D Secure challenge but chose to skip the authentication	Do not continue with current authorization. Prompt the buyer to re-authenticate or request buyer for another form of payment
false	NO	FAILURE	Buyer may have failed the challenge or the device was not verified	Do not continue with current authorization. Prompt the

				buyer to re-authenticate or request buyer for another form of payment
false	NO	BYPASSED	3D Secure was skipped as authentication system did not require a challenge	You can continue with the authorization and assume liability. If you prefer not to assume liability, ask the buyer for another card
false	NO	ATTEMPTED	Card is not enrolled in 3D Secure. Card issuing bank is not participating in 3D Secure	Continue with authorization as authentication is not required
False	NO	UNAVAILABLE	Issuing bank is not able to complete authentication	You can continue with the authorization and assume liability. If you prefer not to assume liability, ask the buyer for another card
False	NO	CARD_INELIGIBLE	Card is not eligible for 3D Secure authentication	Continue with authorization as authentication is not required