

Customer Service Center and Back Office System

Ohio River Bridges

PayPal (SDD)

Version 1.0

March 17, 2022

COPYRIGHT

Copyright © 2022 Electronic Transaction Consultants LLC (ETC). All rights reserved.

The software described in this document, if furnished under a license agreement or nondisclosure agreement, may be used or copied only in accordance with the terms of those agreements.

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Summary of Changes |
|  | 1.0 | ETC | Initial submittal |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



Table of Contents

[1. General 1](#_Toc101140940)

[1.1. Document Purpose 1](#_Toc101140944)

[1.2. Audience 1](#_Toc101140945)

[1.3. Software Detailed Design 1](#_Toc101140948)

[2. Reference Documents 2](#_Toc101140949)

[3. PayPal 3](#_Toc101140950)

[3.1. Functional Requirements 3](#_Toc101140951)

[3.2. PayPal Workflow 3](#_Toc101140952)

[3.3. Technical Design 4](#_Toc101140953)

[3.3.1. UI Design 4](#_Toc101140954)

[3.3.2. PayPal Integration 6](#_Toc101140955)

[3.3.3. Mendix Changes 9](#_Toc101140956)

[3.3.4. DB Changes 14](#_Toc101140957)

[3.3.5. Java Changes 15](#_Toc101140958)

Table of Figures

[Figure 1: Goolge Pay Workflow 3](#_Toc99038463)

Table of Tables

[Table 1: Reference Documents 2](#_Toc99038474)

[Table 2: Google Pay Functional Requirements 3](#_Toc99038475)

Logo

Description automatically generated

# General

This document provides detailed design information for the PayPal Payment processing functionality identified as new addition to the Ohio River Bridges (ORB).

## Document Purpose

The ORB PayPal Software Design Document (SDD) is a formal project document to manage, communicate, and define extensions for the Core features of the Back Office system functionalities per the requirements.

## Audience

The SDD is intended to explain to Ohio River Bridges about incorporating PayPal with existing payment methods.

## Software Detailed Design

The following system components/configurations will be modified to support ORB’s PayPal Payment method requirements:

* Payment check-out screens
* Checkout Microflows
* Java Classes handling the Transaction
* Update credit\_card\_types

# Reference Documents

The table below lists the reference documents for the ORB PayPal SDD.

Table 1: Reference Documents

| **Document Title** | **Description** |
| --- | --- |
| PayPal ICD | Interface control Document |
| PayPal Developer Guide | <https://developer.paypal.com/docs/checkout/> |
| PayPal Checkout Integration Video | <https://youtu.be/QhcyRUZmEm4> |
| PayPal Checkout SDK | https://github.com/paypal/Checkout-Java-SDK |

# PayPal

## Functional Requirements

The functional requirements are listed in the table below.

Table 2: PayPal Functional Requirements

| Requirement ID | Description | Associated Assumption # |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## PayPal Workflow

This section includes workflow of PayPal payment processing.



Figure 1: PayPal Workflow

#### PayPal Workflow Description

The PayPal Process Flow is as follows:

1. Customer clicks on PayPal button.
2. PayPal’s Orders API - Create Order - is called to create a new Order. This includes details about the transaction amount and currency type. The Order API returns the Order Id (JSON element: id) which is used in future API calls.
3. Paypal Checkout Experience popup appears.
4. User approves payment by clicking Pay Now button. The PayPal Checkout popup closes.
5. Payment cart sent and PayPal’s Orders API – Capture Order - is called to capture the payment which returns the transaction details. And transfers money to ORB’s PayPal account.
6. Payment applied and response to browser.
7. Refund call made from BOS system.

## Technical Design

The following sections describe the technical design for PayPal.

### UI Design

User enters the payment checkout screen, user can see the PayPal button.

Graphical user interface, application

Description automatically generated

*Figure 3: PayPal Button – Directed Payments*

Graphical user interface, application

Description automatically generated

*Figure 4: PayPal popup - Login*

Graphical user interface, application

Description automatically generated

*Figure 5: PayPal popup – Payment Choice and Pay Now(Approval)*

Graphical user interface, application

Description automatically generated

*Figure 6: PayPal popup – Pay Now*

Graphical user interface, application

Description automatically generated

*Figure 7: After Approval, PayPal Payment Cart Added and submitted automatically to System*

### PayPal Integration

Steps to integrate PayPal. Please refer to <https://developer.paypal.com/docs/checkout/advanced/integrate>

1. Setup PayPal business account. Get client id from business account website.
2. Add PayPal Javascript snippet(from Javascript SDK) into applicable Mendix page. Include client id in Javascript snippet.
3. createOrder() function calls BOS server to call PayPal’s Orders REST API – Create Order.
4. Modify onApprove() function so that BOS server makes webservice call to PayPal’s Orders REST API - Capture.
5. Modify BOS server code so that on successful Capture response – PayPal values are stored.



Add PayPal Javascript SDK:

*//Include the PayPal JavaScript SDK; replace "test" with your own sandbox Business account app client ID*

*//Add to HTML snippet or index.html*

<script src="https://www.paypal.com/sdk/js?**client-id=test**&currency=USD"></script>

*//Set up a container element for the button – Add to HTML snippet where button should be located.*

<div id="paypal-button-container"></div>

Javascript SDK integration:

Add to Javascript snippet:

paypal.Buttons({

// Order is created on the server and the order id is returned

**createOrder**: (data, actions) => {

//Create PayPalCreateOrderDTO

mx.data.create({

entity: "**Payment.PayPalCreateOrderDTO**",

callback: function (object) {

object.set(“amount”,<TotalAmountString>) ***//Dynamically assign TotalAmount***

},

error: function (error) {

}

});

*//Call MF\_SaveBillingMethodDTO microflow – Adds Payment Method/Cart*

mx.data.action({

params: {

actionname: "**Payment.MF\_PayPalCreateOrder**",

origin: this.mxform

},

callback: function(obj) {

return obj.get("orderId"));

},

error: function(error) {

//alert(error.description);

}

},

// Finalize the transaction on the server after payer approval

**onApprove**: (data, actions) => {

//Create BillingMethodDTO

mx.data.create({

entity: "**Payment.BillingMethodDTO**",

callback: function (object) {

object.set(“PaypageRegistrationId”, **data.orderID)**;

object.set(“RiteCardTypeId”, **“7”**); ***//Value for PayPal credit card type***

object.set(“ExpirationDate”, **“12/99”**);

},

error: function (error) {

}

});

*//Call MF\_SaveBillingMethodDTO microflow – Adds Payment Method/Cart*

mx.data.action({

params: {

actionname: "**Payment.MF\_SaveBillingMethodDTO**",

origin: this.mxform

},

callback: function(obj) {

// expect single MxObject

//alert(obj.get("manufacturer"));

},

error: function(error) {

//alert(error.description);

}

});

}

}).render('#paypal-button-container');

### Mendix Changes

#### Create Order

| Component | Component  Type | Purpose |
| --- | --- | --- |
| MF\_PayPalCreateOrder | Microflow | Microflow that will make a REST call to RiteOnlineGateway. Use MF\_MakeCallRestPayment as a guide. |
| PayPalCreateOrderDTO | Entity | Mendix DTO Entity that only contains amount string attribute. |
| Export\_PayPalCreateOrder | Mapper | Mapper to export PayPalCreateOrderDTO JSON in MF\_ MF\_PayPalCreateOrder. |
| PayPalCreateOrderResponseDTO | Entity | Mendix Response DTO that contains String fields orderId, amount, status. |
| Import\_PayPalCreateOrder | Mapper | Mapper to import PayPalCreateOrderResponseDTO JSON in MF\_ MF\_PayPalCreateOrder. |

* Create MF\_PayPalCreateOrder microflow. This microflow will make a REST call to RiteOnlineGateway.
* Create a new constant called UserLogin.OnlineGatewayPayPalCreateOrder with /paypalCreateOrder as path/value.
* Add this new constant as 3rd parameter in new Rest Call in MF\_PayPalCreateOrder microflow.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

#### Capture Order

**Pages to be modified: –** Add the PayPal Button/Javascript in all the Checkout screens

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Module** | **Name** | **Notes** |
| Page | Payment | Payment | Add a Javascript snippet to add the PayPal integration scripts into it.  On createOrder event: Create PayPalCreateOrderDTO and store TotalAmount. Call MF\_PayPalCreateOrder in the script to get OrderId from PayPal API.  Create BillingMethodDTO and store the orderId received from PayPal as PaypageregistrationID/TokenId. The Payment Form should be updated with credit card type.  Call MF\_SaveBillingMethodDTO in the script once the Pay Now button from PayPal is clicked. |
| Page | Payment | Payment\_Phone |
| Page | Payment | Payment\_Membership\_PayFee |
| Page | Payment | Payment\_Membership\_PayFee\_Phone |
| Page | Payment | Payment\_NoLogin |
| Page | Payment | Payment\_NoLogin\_Phone |
| Page | Payment | Payment\_Apply\_To\_Balance |
| Page | Payment | Payment\_Apply\_to\_Balance\_Phone |
| Page | Payment | Payment\_Membership\_AddNewVehicle |
| Page | Payment | Payment\_Membership\_AddNewVehicle\_Phone |

**Mendix Microflow related changes:**

1. MF\_SaveBillingMethodDTO – Update the Payment object based on the details captured in BillingMethodDTO in Javascript snippet. (if CreditCardType=’PayPal’)
   1. MF\_GetVantivToken is not needed for PayPal processing.
   2. DisplayBillingType should be updated with only the RiteCardTypeDesc - what is displayed in UI Payment cart item.

These changes will impact the /payment JSON value. The values related to PayPal will be sent to the Payment Gateway.

1. Modify the calculate microflow MF\_GetRiteCreditCardTypeDesc with “PayPal” in the Expression.

Graphical user interface, application

Description automatically generated

**if** $CC\_types != **empty** **then**  
$CC\_types/ShortDescription  
**else** **if** $BillingMethodDTO/VantivCardType = 'PP' **then**  
'PAYPAL'  
**else** **if** $BillingMethodDTO/VantivCardType = 'MC' **then**  
'MASTERCARD'  
**else** **if** $BillingMethodDTO/VantivCardType = 'VI' **then**  
'VISA'  
**else** **if** $BillingMethodDTO/VantivCardType = 'AX' **then**  
'AMERICAN EXPRESS'  
**else** **if** $BillingMethodDTO/VantivCardType = 'DI' **then**  
'DISCOVER'  
**else** **empty**

Diagram

Description automatically generated

*Figure: MF\_SaveBillingMethodDTO*

Graphical user interface

Description automatically generated

For DisplayBillingType in Change ‘NewPayments’ Activity adjust value to:

**if** $UserBillingMethodDropdown/CreditCardTypeId = -99 **then**  
$BillingMethodDTO/RiteCardTypeDesc + ' (...' + $BillingMethodDTO/CCLastFour + ')'  
**else** **if** $BillingMethodDTO /CreditCardTypeId = 7<PayPal**>**) **then**  
$BillingMethodDTO/RiteCardTypeDesc  
**else** $UserBillingMethodDropdown/DisplayBillingType

Also add a new field to the same activity(This will assign the orderId from PayPal Create Order response to the PaymentToken that is submitted to Payment Gateway. The orderId is required for PayPal Capture Order API call):

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, diagram

Description automatically generated

*Figure: MF\_SaveBillingMethodDTO*

Modify decision so that PayPal payments don’t retrieve Vantiv Token.

$UserBillingMethodDropdown/CreditCardTypeId = -99  
**and**  
$BillingMethodDTO /CreditCardTypeId != 7<PayPal**>**

Graphical user interface, text, application, chat or text message

Description automatically generated

Add CC\_Type\_ID = 7, code = ‘PP’ and ShortDescription and LongDescription = ‘PayPal’ to AccountView CC\_types. Might not be required after DeDup implementation.

### DB Changes

1. Update the credit\_card\_types table to add PayPal as a credit card type. Should use Credit card PaymentForm id for PayPal also during implementation.

insert into tag\_owner.credit\_card\_types (CREDIT\_CARD\_TYPE\_ID, CODE, LEDGER\_ACCOUNT\_NUMBER, ABBREVIATION, SHORT\_DESCRIPTION, LONG\_DESCRIPTION, IS\_ACTIVE, IS\_DEFAULT, SORT\_ORDER, RECORD\_VERSION, DATE\_CREATED, DATE\_MODIFIED, MODIFIED\_BY, CREATED\_BY, IS\_RPT\_UI\_DISP)

values (7, 'P', '60005', 'PP', 'PAYPAL', 'PAYPAL', 'Y', 'N', 10, 1, sysdate, sysdate, 'TAG\_OWNER', 'TAG\_OWNER', 'Y');

1. Please review TAG\_OWNER.PMT\_APPLIANCE\_API.authorize\_cc\_payment\_ws\_call, TAG\_OWNER.PMT\_POSTING\_XML and TAG\_OWNER.PMT\_POSTING\_IMPL.

### Java Changes

#### Create Order

| Component | Component  Type | Purpose |
| --- | --- | --- |
| PayPalController | Rite-online-gateway Java class | Controller that sends kafka message. Use PaymentController as guide. |
| PayPalCreateOrderDTO | Rite-online-gateway Java class | DTO object. Only has amount string field.Use PaymentDTO as guide. Used by PayPalController. |
| PayPalCreateOrderResponseDTO | Rite-online-gateway Java class | Response DTO that contains String fields orderId, amount, status. Used by PayPalController. |
| RiteOnlinePayPalConsumer | Mendix-Adapter Java class | Kafka consumer. Calls RiteOnlinePayPalConsumerProcess. Use RiteOnlinePaymentConsumer as guide. |
| RiteOnlinePayPalConsumerProcess | Mendix-Adapter Java class | Kafka consumer processor. Calls payment-appliance.PANManagerImpl.createPayPalOrder(amount). Use RiteOnlinePaymentConsumerProcess as guide. |
| PayPalCreateOrderWrapperDTO | Mendix-Adapter Java class | Same class as PayPalCreateOrderDTO. Use CartMakePaymentWrapperDTO as a guide. Used by RiteOnlinePayPalConsumerProcess. |
| PayPalCreateOrderResponseDTO | rite-paymentappliance java class | Response DTO that contains String fields orderId, amount, status. Used in payment-appliance.PANManagerImpl.createPayPalOrder and CCProcessor.createPayPalOrder |
| PayPalCreateOrderResponseWrapperDTO | Mendix-Adapter Java class | Response DTO that contains String fields orderId, amount, status. Used by RiteOnlinePayPalConsumerProcess. |
| CCProcessorPayPal | rite-paymentappliance java class | Interface to PayPal API. PayPal Checkout SDK will be used. |

Create objects listed in above table.

Add PayPal Checkout API SDK to rite-paymentappliance pom.xml:

<dependency>

<groupId>com.paypal.sdk</groupId>

<artifactId>checkout-sdk</artifactId>

<version>1.0.5</version>

</dependency>

Create kafka topics:

topic-paypal-create-order-riteonline-request

topic-paypal-create-order-riteonline-reply

group-paypal-create-order-riteonline-reply-gateway

Add topics to application.properties in rite-online-gateway.

* Copy all contents in rite-paymentappliance\payment\_core\payment-ws\src\main\java\com\etcc\paymentappliance\ccprocessing\**litle** directory and paste into new directory named rite-paymentappliance\payment\_core\payment-ws\src\main\java\com\etcc\paymentappliance\ccprocessing\**paypal**
* Rename any references to Litle to Paypal.
* Add the following in PaymentWSApplicationContext.xml:

<bean id="payPalProcessor"

class="com.etcc.paymentappliance.ccprocessing.paypal.wsclient.CCProcessorPayPal" />

Review <https://github.com/paypal/Checkout-Java-SDK> documentation to implement Create Order, Capture Order and Refund API calls to PayPal from rite-paymentappliance.CCProcessorPayPal. This SDK requires Java 8.

Add the following method to rite-online-gateway.PayPalController:

@ApiOperation(value = "Create Order for PayPal")

@PostMapping(value = "/paypalCreateOrder", produces = MediaType.APPLICATION\_JSON\_VALUE)

**public** ResponseEntity<?> makePayment(@RequestBody @Nonnull PayPalCreateOrderDTO paypalCreateOrder)

In Mendix-Adapter.RiteOnlinePayPalConsumerProcess include the following code to make call to rite-paymentappliance.PANManagerImpl:

PANManagerImpl panManagerImpl = **null**;

PANManagerServiceLocator serviceLocator = **new** PANManagerServiceLocator();

String panmananagerurl = getSysParamValue("PCI\_APPLIANCE\_WS\_URL");

panManagerImpl = serviceLocator.getPANManagerPort(**new** URL(panmananagerurl));

PaypalCreateOrderResponseDTO response = panManagerImpl.createPayPalOrder(amount);

Add the following method to rite-paymentappliance.PANManagerImpl.java:

@WebMethod()

**public** PaypalCreateOrderResponseDTO createPayPalOrder(@WebParam(name = "amount") String amount) {

This method will call rite-paymentappliance.CCProcessorPayPal.createPayPalOrder(amount):

**public** PaypalCreateOrderResponseDTO createPayPalOrder(String amount) **throws** IOException {

OrdersCreateRequest request = **new** OrdersCreateRequest();

request.header("prefer","return=representation");

request.requestBody(buildRequestBody(amount));

HttpResponse<Order> response = client().execute(request); //client is PayPalClient

//set PaypalCreateOrderResponseDTO values

**return** responseDTO;

}

**private** OrderRequest buildRequestBody(String amount) {

OrderRequest orderRequest = **new** OrderRequest();

orderRequest.checkoutPaymentIntent("CAPTURE");

ApplicationContext applicationContext = **new** ApplicationContext()

.cancelUrl("https://www.example.com").returnUrl("https://www.example.com");

orderRequest.applicationContext(applicationContext);

List<PurchaseUnitRequest> purchaseUnitRequests = **new** ArrayList<>();

PurchaseUnitRequest purchaseUnitRequest = **new** PurchaseUnitRequest()

.amountWithBreakdown(**new** AmountWithBreakdown().currencyCode("USD").value(amount)); //amount set here

purchaseUnitRequests.add(purchaseUnitRequest);

orderRequest.purchaseUnits(purchaseUnitRequests);

**return** orderRequest;

}

##### Sample Create Order request:

POST https://api-m.sandbox.paypal.com/v2/checkout/orders

-H "Content-Type: application/json"

-H "Authorization: Bearer Access-Token"

-d '{

"intent": "CAPTURE",

"purchase\_units": [

{

"amount": {

"currency\_code": "USD",

"value": "100.00"

}

}

]

}'

**Sample Create Order response:**

{

"id": "5O190127TN364715T", //This Order Id value is needed for Capture request.

"status": "CREATED",

"links": [

{

"href": "https://api-m.paypal.com/v2/checkout/orders/5O190127TN364715T",

"rel": "self",

"method": "GET"

},

{

"href": "https://www.paypal.com/checkoutnow?token=5O190127TN364715T",

"rel": "approve",

"method": "GET"

},

{

"href": "https://api-m.paypal.com/v2/checkout/orders/5O190127TN364715T",

"rel": "update",

"method": "PATCH"

},

{

"href": "https://api-m.paypal.com/v2/checkout/orders/5O190127TN364715T/capture",

"rel": "capture",

"method": "POST"

}

]

}

#### Capture Order

Modify rite-paymentappliance.PANManagerImpl.processTransaction:

Graphical user interface, text, application, email

Description automatically generated

Modify CCProcessorPayPal.generatePaymentRequest() so that PayPal Capture Order API call is made:

**private** CCProcessorResponse processCreditPaymentRequests(CCProcessorRequest request) {

OrdersCaptureRequest request = **new** OrdersCaptureRequest(request.getTokenId()); //payPal orderId

request.requestBody(**new** OrderRequest());

HttpResponse<Order> response = client().execute(request); //client is PayPalClient

ccResponse.setCcProcessorTransId(String.valueOf(response.result().purchaseUnits().get(0).payments.get(0).id())); //captureId needed for refund

ccResponse.setCcProcessorStatusCode(response.statusCode());

ccResponse.setStatusMessage(response.result().status());

ccResponse.setCcProcessorResponseString(**new** JSONObject(**new** Json().serialize(response.result())).toString());

**return** ccResponse;

}

**Sample Capture request:**

POST https://api-m.sandbox.paypal.com/v2/checkout/orders/5O190127TN364715T/capture

-H "Content-Type: application/json"

-H "Authorization: Bearer Access-Token"

**Sample Capture response:**   
{

"id": "5O190127TN364715T",

"status": "COMPLETED",

"payer": {

"name": {

"given\_name": "John",

"surname": "Doe"

},

"email\_address": "customer@example.com",

"payer\_id": "QYR5Z8XDVJNXQ"

},

"purchase\_units": [

{

"reference\_id": "d9f80740-38f0-11e8-b467-0ed5f89f718b",

"shipping": {

"address": {

"address\_line\_1": "2211 N First Street",

"address\_line\_2": "Building 17",

"admin\_area\_2": "San Jose",

"admin\_area\_1": "CA",

"postal\_code": "95131",

"country\_code": "US"

}

},

"payments": {

"captures": [

{

"id": "3C679366HH908993F", //Capture Id. Store this for future refund/partial refund.

"status": "COMPLETED",

"amount": {

"currency\_code": "USD",

"value": "100.00"

},

"seller\_protection": {

"status": "ELIGIBLE",

"dispute\_categories": [

"ITEM\_NOT\_RECEIVED",

"UNAUTHORIZED\_TRANSACTION"

]

},

"final\_capture": true,

"disbursement\_mode": "INSTANT",

"seller\_receivable\_breakdown": {

"gross\_amount": {

"currency\_code": "USD",

"value": "100.00"

},

"paypal\_fee": {

"currency\_code": "USD",

"value": "3.00"

},

"net\_amount": {

"currency\_code": "USD",

"value": "97.00"

}

},

"create\_time": "2018-04-01T21:20:49Z",

"update\_time": "2018-04-01T21:20:49Z",

"links": [

{

"href": "https://api-m.paypal.com/v2/payments/captures/3C679366HH908993F",

"rel": "self",

"method": "GET"

},

{

"href": "https://api-m.paypal.com/v2/payments/captures/3C679366HH908993F/refund",

"rel": "refund",

"method": "POST"

}

]

}

]

}

}

],

"links": [

{

"href": "https://api-m.paypal.com/v2/checkout/orders/5O190127TN364715T",

"rel": "self",

"method": "GET"

}

]

}

#### Credit/Refund

Modify rite-paymentappliance.CCProcessorPayPal.processRefundRequests method so that PayPal Refund API call is made:

**private** CCProcessorResponse processRefundRequests(CCProcessorRequest request) **throws** Exception {

logger.info("Start : processRefundRequests ");

CapturesRefundRequest request = **new** CapturesRefundRequest(request.setCcProcessorTransId); //PayPal CaptureId

request.prefer("return=representation");

request.requestBody(**new** RefundRequest());

HttpResponse<Refund> response = client().execute(request); //client is PayPalClient

ccResponse.setCardTransactionId(Long.valueOf(response.getId()));

ccResponse.setCcProcessorTransId(String.valueOf(response.result().id()));

ccResponse.setCcProcessorStatusCode(response.statusCode());

ccResponse.setStatusMessage(response.result().status());

ccResponse.setCcProcessorResponseString(**new** JSONObject(**new** Json().serialize(response.result())).toString(0))

logger.info("End : processRefundRequests ");

**return** ccResponse;

}

**Sample refund request:**

POST https://api-m.sandbox.paypal.com/v2/payments/captures/3C679366HH908993F/refund

-H "Content-Type: application/json"

-H "Authorization: Bearer Access-Token"

**Sample refund response:**

{

"id": "1JU08902781691411",

"status": "COMPLETED",

"links": [

{

"rel": "self",

"method": "GET",

"href": "https://api-m.paypal.com/v2/payments/refunds/1JU08902781691411"

},

{

"rel": "up",

"method": "GET",

"href": "https://api-m.paypal.com/v2/payments/captures/2GG279541U471931P"

}

]

}