



PayPal Credit Messaging Ocap Integration Guide

FOR SALESFORCE COMMERCE CLOUD



Table of Contents

1. Summary	1
2. Component Overview	1
2.1 Supported Features.....	1
2.1.1 Smart Payment Button	1
2.1.2 Billing Agreement and Reference Transactions	1
2.1.3 Alternative Payment Methods	1
2.1.4 Transactions post processing	2
2.1.5 PayPal-provided Billing Address and Phone Number	2
2.1.6 PayPal credit messaging banners functionality	2
2.1.7 Webhook Support	2
2.2 Privacy and Payment	2
3. What is OCAPI?	2
3.1 Overview	2
3.2 How is OCAPI used?	3
4. Implementation Guide	3
4.1 Overview	3
4.2 Installation	3
4.2.1 Import metadata archive	3
4.2.2 Add cartridge to cartridge path.....	4
4.2.3 Adding API Credentials	4
4.2.4 Papal Transaction and PayPal Styles Configuration Business Manager modules installation	6
4.2.5 Custom cache configuration	6
4.3 Configuration.....	7
4.3.1 Updating Cartridge Custom Site Preferences	7
4.3.2 Job configuration for removal outdated transaction.....	8
4.3.3 Static values configuration	8
4.3.4 Disable funding property.....	8
4.3.5 Enable funding property	9
4.3.6 Alternative Payment Methods Configuration	9
4.3.7 Service Profile Configuration.....	9
4.4 Webhook support	9
4.4.1 Payment authorization voided	10
4.4.2 Payment capture refunded	10
4.4.3 Payment capture completed	10
4.5 Open commerce API Settings	10
5. Operations and Maintenance	15
5.1 Data Storage.....	15
5.2 Logs.....	15
5.3 HTTP Service Availability	16
5.4 Testing	16
5.5 Support.....	17
6. User Guide	17
6.1 Business Manager modules	17
6.1.1 PayPal Transactions	17
6.1.2 PayPal Styles Configuration	23
6.2 Storefront Functionality	25

7. Multi-site and multi-credential support	26
7.1 General info	26
8. Open commerce API Hooks	26
8.1 Ocapi Functionality (Ocapi Hooks)	26
8.1.1 Basic Variable's description.	26
8.1.2 A Registered and an unregistered user authentication.	27
8.1.3 Get banner configuration for category page.	30
8.1.4 Get banner configuration for product page.	31
8.1.5 Create billing agreement token.	31
8.1.6 Add billing agreement.	32
8.1.7 Display a registered user's payment instruments (billing agreements). ..	33
8.1.8 Delete a registered user's payment instrument (billing agreement)	35
9. An order placing due a PayPal payment method by using OCAPI	35
9.1.1 Create a basket	35
9.1.2 The basket's flashes	37
9.2 Add an item (product) to the basket.....	38
9.3 Sets the billing address of a basket	40
9.4 Update a shipment of basket	41
9.5 Sets a customer email for an existing basket.....	43
9.6 Add a payment instrument to the basket	44
9.7 Get basket (get banner config for cart / minicart / billing pages)	48
9.7.1 Url's query parameters.....	49
9.7.2 PayPal buttons config objects creation	49
9.7.3 Get banner config for cart / minicart / billing pages.....	49
9.7.4 A billing agreement token(c_paypalToken) and a purchase unit creation	49
9.7.5 A billing agreement and order details object creating	51
9.8 Place order	51
9.9 Additional Ocapi resources	54

Copyright Information

© 2020 PayPal, Inc. All rights reserved.

PayPal is a registered trademark of PayPal, Inc. The PayPal logo is a trademark of PayPal, Inc. Other trademarks and brands are the property of their respective owners. The information in this document belongs to PayPal, Inc. It may not be used, reproduced, or disclosed without the written approval of PayPal, Inc.

Copyright © PayPal. All rights reserved. PayPal (Europe) S.à r.l. et Cie, S.C.A., Société en Commandite par Actions. Registered office: 22-24 Boulevard Royal, L-2449, Luxembourg, R.C.S. Luxembourg B 118 349

Consumer advisory: The PayPal™ payment service is regarded as a stored value facility under Singapore law. As such, it does not require the approval of the Monetary Authority of Singapore. You are advised to read the terms and conditions carefully.

Notice of Non-Liability

PayPal, Inc. is providing the information in this document to you “AS-IS” with all faults. PayPal, Inc. makes no warranties of any kind (whether express, implied or statutory) with respect to the information contained herein. PayPal, Inc. assumes no liability for damages (whether direct or indirect), caused by errors or omissions, or resulting from the use of this document or the information contained in this document or resulting from the application or use of the product or service described herein. PayPal, Inc. reserves the right to make changes to any information herein without further notice.

1. Summary

This Implementation Guide describes how to integrate **paypal_credit_messaging_ocapi**, **bm_paypal** and **bm_paypal_configuration** cartridges version 22.2.0 into your site.

Cartridges includes:

- PayPal Checkout from cart and payment pages
- Billing Agreement creation in the checkout process
- PayPal credit messaging banners functionality
- Review and post-process PayPal transactions in the Business Manager
- Dynamic Smart Button styling management from the Business Manager

PayPal integration uses PayPal REST Order API for checkout and Payment API for transaction post processing in Business Manager.

For more information, contact your PayPal support manager.

2. Component Overview

2.1 Supported Features

2.1.1 Smart Payment Button

PayPal Checkout with Smart Payment Buttons gives your buyers a simplified and secure checkout experience. PayPal intelligently presents the most relevant payment types to your shoppers, automatically, making it easier for them to complete their purchase using methods like Pay with Venmo, PayPal Credit, credit card payments.

[Button Demo](#)

2.1.2 Billing Agreement and Reference Transactions

The PayPal Billing Agreement (Reference Transactions) feature helps customers to pay more quickly. If a buyer accepts Billing Agreements with your site, they can check out without redirection to PayPal, both from the Cart and Billing page.

To learn more, see [Reference transactions overview](#).

NOTE: Reference transaction needs to be enabled in a merchant account setting.
Please contact your PayPal support manager enable this feature on your account

2.1.3 Alternative Payment Methods

Alternative payment methods allow customers across the globe to pay with their bank accounts, wallets, and other local payment methods. Relevant alternative payment methods are automatically presented with Smart Payment Buttons.

To learn more, see [Alternative Payment Methods overview](#).

2.1.4 Transactions post processing

Orders paid with integration can be post-processed from separate Business Manager module. Post-processing includes transaction capture, refund and void. Also, you can create a new transaction using existed billing agreement id.

Check [Business Manager](#) chapter for more details.

2.1.5 PayPal-provided Billing Address and Phone Number

To retrieve a buyer's billing address and phone number from the PayPal Checkout flow, please contact your PayPal representative or PayPal Support to enable them.

Important: Retrieve billing address and phone number should be enabled in merchant account before PayPal button will be enabled on a cart page.

2.1.6 PayPal credit messaging banners functionality

PayPal Credit is a revolving line of credit that gives your customers the flexibility to buy now and pay overtime. Consumers can use PayPal Credit on purchases at thousands of stores, like yours, that accept PayPal by selecting PayPal Credit at checkout.

PayPal Credit messaging is available for merchants who wish to promote the availability of special financing offers, which help increase sales.

For more details check [Credit Messaging Overview](#)

Credit Message feature is not available with enabled billing agreement. Please make sure you disable billing agreements in the custom preference section before cartridge installation.

2.1.7 Webhook Support

The purpose of the webhooks is to reflect changes which was done manually or automatically on PayPal console level.

For now only the next hooks are supported: **Payment authorization voided, Payment capture refunded and Payment capture completed.**

2.2 Privacy and Payment

This integration requires access to the following customer data elements: Shipping Address, Order Details, Customer Profile.

3. What is OCAPI?

3.1 Overview

The Salesforce Commerce Cloud Open Commerce API (OCAPI) is an API that allows other applications to securely access the platform's resources externally from the platform itself. An API, or Application Programming interface, is the preferred method for two applications to communicate

in real-time. Simply stated, an API allows a server to return or accept information that is requested or sent by another application. The calling application can then interpret the received data and present it to the user.

3.2 How is OCAPI used?

The OCAPI interface grants other applications access to Salesforce Commerce Cloud resources such as baskets, customers, orders, and products. While these resources are usually consumed through the client site storefront that customers see, the interface provides access to those resources for other applications to read, create, or update through service calls.

The general information about OCAPI you can find here: [OCAPI documentation](#)

4. Implementation Guide

4.1 Overview

Three cartridges support this integration:

1. `paypal_credit_messaging_ocapi` - The cartridge makes possible to connect PayPal solution to your site by using OCAPI.
2. `bm_paypal` - Business Manager extension “PayPal Transactions”
3. `bm_paypal_configuration` - Business Manager extension “PayPal Styles Configuration”

4.2 Installation

4.2.1 Import metadata archive

1. Upload and import meta.zip from the metadata folder. To do so, go to **Business Manager > Administration > Site Development > Site Import & Export**. Upload archive using Local option in the Upload Archive section. After upload choose metadata.zip in the list and click on import button.

Site Import & Export

This page allows you to export the current configuration of your organization including all of its sites. To download an archive, just click its file name.

Import

Upload Archive:

☒ Local ☐ Remote

No file selected.

SELECT	Name	Location	File Size	Last Modified
<input checked="" type="radio"/>	metadata.zip	local	6.08 KB	8/27/20 11:26:08 am
<input type="radio"/>	SiteGenesis Demo Site			
<input type="radio"/>	Storefront Reference Architecture Demo Sites			

Figure 1. metadata.zip archive import

4.2.2 Add cartridge to cartridge path

Add paypal_credit_messaging_ocapi into the cartridge path of a target site. To do so, go to **Business Manager > Administration > Sites > Manage Sites > Your Target Site > Settings** and insert paypal_credit_messaging_ocapi before your cartridges record, as shown in Figure 2.

RefArch - Settings

Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

Instance Type:	Sandbox/Development
<small>Deprecated. The preferred way of configuring HTTP and HTTPS hostnames is by using new features of the site aliases configuration ("SEO > Aliases Configuration"). The HTTP/HTTPS hostname values set in this section will be used if no hostnames are defined by aliases configuration and are intended only to support an older configuration style.</small>	
HTTP Hostname:	<input type="text"/>
HTTPS Hostname:	<input type="text"/>
Instance Type:	All
Cartridges:	paypal_credit_messaging_ocapi:app_storefront_base

Figure 2. Storefront cartridge path

4.2.3 Adding API Credentials

Services:

- int_paypal.http.rest - is used for the main plugin logic (Smart Button, manipulations with transactions and etc.).
- int_paypal.http.token.service - is used for Webhooks features.

To access credentials go to **Business Manager > Administration > Operations > Services > Credentials**.

You need to fill the next credential with the same data:

- Paypal_Sandbox_Credentials
SB URL: <https://api.sandbox.paypal.com/>

Prod URL: <https://api.paypal.com/>

User: Client Id

Pass: Secret

- Paypal_Sandbox_Connect_Credentials

SB URL: <https://api-m.sandbox.paypal.com/v1/>

Prod URL: <https://api-m.paypal.com/v1/>

User: Client Id

Pass: Secret

For both credentials use Client Id as **User** and Secret as **Password**. Client Id and Secret can be obtained in the app details at the PayPal developer portal.

For more details about REST apps and credentials please visit [Get credentials](#) page at the PayPal Developer Portal.

NOTE: Current integration supports only one credentials usage at a time. If you wish to use multiple credentials at one sandbox you should customize cartridge at PaypalRestService.js file

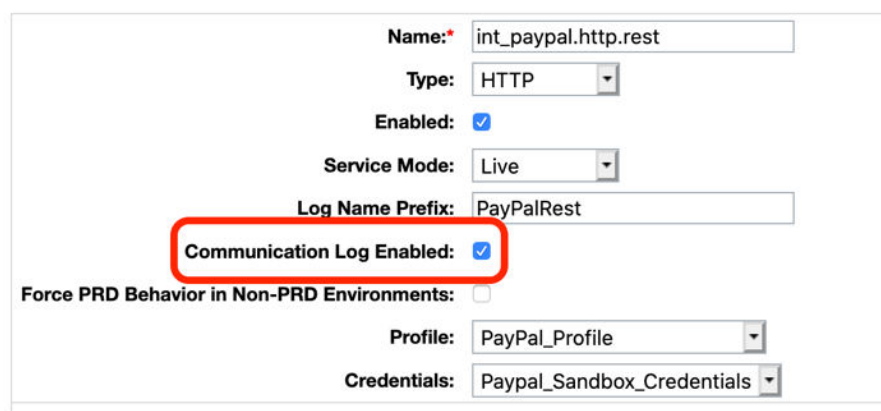
(Optional) For non-production activities we recommend enabling the communication log for the `int_paypal.http.rest` service. The communication log will log every request and response to the log files. To do this navigate to the **Business Manager > Administration > Operations > Services** and click on `int_paypal.http.rest`. Check Communication Log Enabled checkbox (Figure 4)

PLEASE NOTE: By enabling the communication log, some details such as customer's personal information including address, phone, and email will be logged in cleartext. This should only be used for debugging purposes. We recommend, if possible, only use this in a sandbox; however, when used in production, ensure you are disabling / unchecking the communication log box immediately after you are done with your debugging. Further, some credentials will also be saved in cleartext.

[Administration](#) > [Operations](#) > [Services](#) > `int_paypal.http.rest` - Details

`int_paypal.http.rest` 

Fields with a red asterisk (*) are mandatory. Click **Apply** to save the details. Click **Reset** to revert to the last saved state.



Name:	*	int_paypal.http.rest
Type:		HTTP
Enabled:		<input checked="" type="checkbox"/>
Service Mode:		Live
Log Name Prefix:		PayPalRest
Communication Log Enabled:		<input checked="" type="checkbox"/>
Force PRD Behavior in Non-PRD Environments:		<input type="checkbox"/>
Profile:		PayPal_Profile
Credentials:		Paypal_Sandbox_Credentials

Figure 3. Service Example

4.2.4 Papal Transaction and PayPal Styles Configuration Business Manager modules installation

Add **bm_paypal** into the record of the Business Manager cartridge path. Go to **Business Manager > Administration > Sites > Manage Sites > Manage the Business Manager Site > Settings** and add **bm_paypal** and **bm_paypal_configuration** to the input as shown in Figure 4.

[Administration](#) > [Sites](#) > [Manage Sites](#) > Business Manager - Settings

Settings Cache Hostnames

Business Manager - Settings

Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

Instance Type:

Deprecated. Up to two instance specific hostname aliases for Business Manager can be configured here.

HTTP Hostname:

HTTPS Hostname:

Instance Type: All

Cartridges:

Effective Cartridge Path: app_business_manager:plugin_apple_pay:plugin_facebook:plugin_payments:plugin_pinterest_commerce:plugin_web_payments:bc_imp

Figure 4. Business Manager Cartridge Path

Grant access to Business Manager modules:

- Go to **Business Manager > Administration > Organization > Roles & Permissions** and select the target role that needs to have access to PayPal Transaction management.
- Select the **Business Manager Modules** tab in the drop-down list on top. You must select your target site as the active context.
- Scroll to the **PayPal Transactions** module. Grant access to it by selecting the checkbox and clicking **Update**. Repeat same action for **PayPal Styles Configuration**.

Once you do this, every Business Manager user with that role can manage PayPal Transactions and Credit Banner styling by selecting **Business Manager > Merchant Tools > Ordering > PayPal Transactions** and change Smart Button styling under **Business Manager > Merchant Tools > Site Preferences > PayPal Styles Configuration**.

4.2.5 Custom cache configuration

Custom cache required for fast and stable work on the integration. Navigate to **Business Manager > Administration > Operations > Custom Caches** and check checkbox **Enable Caching**.

There is custom cache objects used by the integration:

1. paypalRestOAuthToken - OAUTH token for communication with PayPal REST API. Out of the box, the integration supports single site usage and token retrieved from PayPal is cached. For multiple credentials usage we recommend removing cache definition from caches.json file in the cartridge.

NOTE: Custom Cache could be disabled for developing or testing purposes. But we don't recommend disabling it on production and staging to avoid performance issue and http calls quota violation.

4.3 Configuration

4.3.1 Updating Cartridge Custom Site Preferences

Go to **Business Manager > Merchant Tools > Site Preferences > Custom Site Preferences**. You'll see Custom Site Preference Group called PayPal Configuration (Figure 5.1.)



ID	Name	Description	Preferences	View Across Sites
Storefront Configs	Storefront Configurations		5	View
Paypal_Checkout	Paypal Configuration		10	View

Figure 5.1. PayPal Custom Site Preferences

Click **View** for PayPal Configuration. Detailed description about each option is available under preference name on the page.

NOTE: Show PayPal button on the cart page requires customer shipping address. Review [PayPal-provided Billing Address and Phone Number](#) before enabling this preference.

NOTE: Review section [Billing Agreement and Reference Transactions](#) section before enable Billing Agreement feature.

Also go to **Business Manager > Merchant Tools > Site Preferences > Custom Site Preferences**. You'll see Custom Site Preference Group called PayPal Credit Messaging (Figure 5.2.).

ID	Name	Description	Preferences	View Across Sites
Storefront Configs	Storefront Configurations		14	View
Paypal_Branding	PayPal Branding (NVP)		9	View
Paypal_Express_Checkout	PayPal Express Checkout (NVP)		15	View
Paypal_Checkout	Paypal Configuration		7	View
Paypal_Credit_Messaging	Paypal Credit Messaging		3	View
Paypal_Credit_Financial_O...	PayPal Credit Financing Options and Marketi...		4	View

5.2. PayPal Credit Messaging custom preferences group

Click **View** for PayPal Credit Messaging.

Preferences in the group allow you to toggle visibility of the credit message for PDP, category and cart page.

4.3.2 Job configuration for removal outdated transaction

To record transaction created from Business Manager cartridge use custom object feature. After the year transaction became outdated and to remove it job with recurring interval was imported with metadata.

Job has site context and default site id is RefArch. If you have different site id, navigate to **Business Manager > Administration > Operations > Jobs >**

RemoveOutdatedPayPalTransaction > Job Steps and click on RefArch mark near Scope and select new site to run.

By default, job will run every 2 months. If you don't create transaction from Business Manager or you want to change a job interval, navigate to **Business Manager > Administration > Operations > Jobs > RemoveOutdatedPayPalTransaction > Schedule and History**. Uncheck Enable checkbox if you want to disable the job or change an interval in the menu below.

4.3.3 Static values configuration

All configuration option available in `cartridge/config/sdkConfig.js` file. You could change

1. Allowed currencies. By default, all currencies are allowed. Currency list available under allowedCurrencies variable. More details [Currency](#)
2. Disable funding property. By default, only alternative payment methods are disabled. More details [Disable funding](#) and [Alternative Payment Methods](#)
3. Enable funding property is for adding additional buttons using PayPal SDK. More details [Enable funding](#)
4. Static image for PayPal button. Will be rendered on a page in use cases where quick checkout is available, and load of the Smart Payment Button is not required. List of images for usage - [Available images](#)

4.3.4 Disable funding property

To add additional values to the SDK disable-funding property, navigate to **Business Manager > Merchant Tools > Site Preferences > Custom Site Preferences -> PayPal Configuration**. Add payment method id to the list and click Add (Figure 6.1.).

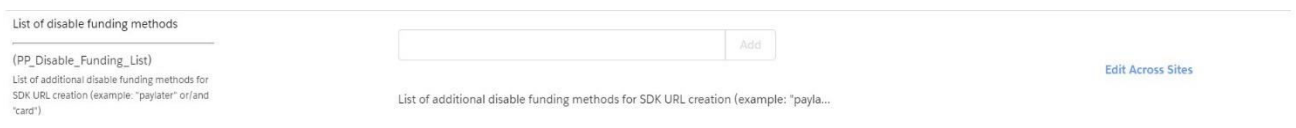


Figure 6.1. List of

disable funding methods

4.3.5 Enable funding property

To add additional values to the SDK enable-funding property, navigate to **Business Manager > Merchant Tools > Site Preferences > Custom Site Preferences -> PayPal Configuration**. Add payment method id to the list and click Add (Figure 6.2.).

List of enable funding methods

(PP_Enable_Funding_List)

List of additional enable funding methods for SDK URL creation (example: "paypal...")

Add

Edit Across Sites

Figure 6.2. List of enable funding methods

4.3.6 Alternative Payment Methods Configuration

By default, all payment methods are disabled for Smart Button. To enable payment method, navigate to **Business Manager > Merchant Tools > Site Preferences > Custom Site Preferences -> PayPal Configuration**. Add payment method id to the list and click Add (Figure 6.3.).

List of IDs can be found [here](#).

Will be displayed on the PayPal page if billing agreement feature is enabled. ...

Merchant ID

(PP_API_Merchant_Id)

(String)

The merchant for whom you are facilitating a transaction.

The merchant for whom you are facilitating a transaction.

Available Alternative Payment Methods

(PP_API_APM_methods)

List of IDs of the enabled alternative payment methods.

List of IDs of the enabled alternative payment methods.

Add

mybank

Figure 6.3. Alternative Payment Methods configuration

NOTE: Alternative Payment Methods are available only on the billing page and will appears only if *Capture funds immediately* preference set to Yes. For registered user also required to disable *Billing Agreement Enabled* preference.

4.3.7 Service Profile Configuration

By default, integration doesn't have any service profile configuration. We recommend putting limitation and timeout values according to a storefront traffic to prevent fraud or potential attack. You can read more about configuration values [here](#).

4.4 Webhook support

The purpose of the webhooks is to reflect changes which was done manually or automatically on PayPal console level.

To configure webhook support, follow [this](#) PayPal guide.

After covering configurations mentioned in PayPal guide - get Webhook ID from PayPal dev console (from SANDBOX WEBHOOKS section). And enter in to the Custom Site Preference with ID: *PP_WH_Authorization_And_Capture_Id* (which located inside *Paypal Configuration Custom Site Pref Group*).

NOTE: you may subscribe only on events listed below.

4.4.1 Payment authorization voided

Triggers when payment has been voided.

4.4.2 Payment capture refunded

Triggers when a merchant refunds a payment capture.

4.4.3 Payment capture completed

Triggers when a payment capture completes.

4.5 Open commerce API Settings

Here's how:

Go to **Business Manager > Administration > Open Commerce API Settings**

Open Commerce API Settings

This page allows you to make client application-specific configurations of Open Commerce API resources, i.e. manage resource access privileges, at (organization-wide). Please note that due to caching, changes may take up to three minutes to become effective. You can browse the Open Commerce API here [API Explorer](#).



Then choose: **Select Type - Shop, Select Content - your site id.**

And paste the following JSON:

```
{
  "_v": "22.4",
  "clients": [{
    "client_id": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
    "allowed_origins": [],
    "resources": [
      {
        "resource_id": "/customers",
        "methods": ["post"],
        "read_attributes": "(**)",
```

```

        "write_attributes": "(**)"
    },
    {
        "resource_id": "/customers/auth",
        "methods": ["post"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/customers/{customer_id}",
        "methods": ["get"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id":
"/customers/{customer_id}/payment_instruments/{payment_instrument_id}",
        "methods": ["get", "delete"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/customers/{customer_id}/payment_instruments",
        "methods": ["post"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets",
        "methods": ["post"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    }

```

```

    },

    {
        "resource_id": "/baskets/{basket_id}",
        "methods": ["get"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets/{basket_id}/items",
        "methods": ["post"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets/{basket_id}/shipments",
        "methods": ["post"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets/{basket_id}/billing_address",
        "methods": ["put"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets/{basket_id}/payment_instruments",
        "methods": ["post"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
}

```



```

{
    "resource_id": "/categories/{category_id}",
    "methods": ["get"],
    "read_attributes": "(**)",
    "write_attributes": "(**)"
},
{
    "resource_id": "/products/{product_id}",
    "methods": ["get"],
    "read_attributes": "(**)",
    "write_attributes": "(**)"
},
{
    "resource_id": "/orders",
    "methods": ["post"],
    "read_attributes": "(**)",
    "write_attributes": "(**)"
},
{
    "resource_id": "/baskets/{basket_id}/shipments/{shipment_id}",
    "methods": ["patch"],
    "read_attributes": "(**)",
    "write_attributes": "(**)"
},
{
    "resource_id": "/customers/{customer_id}/payment_instruments",
    "methods": ["get"],
    "read_attributes": "(**)",
    "write_attributes": "(**)"
},
{

```

```

        "resource_id":
"/customers/{customer_id}/payment_instruments/{payment_instrument_id}",
        "methods": ["get"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets/{basket_id}/customer",
        "methods": ["put"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets/{basket_id}/payment_methods",
        "methods": ["get"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/baskets/{basket_id}",
        "methods": ["patch", "delete"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/product_search",
        "methods": ["get"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    },
    {
        "resource_id": "/products/{product_id}",

```

```

        "methods": ["get"],
        "read_attributes": "(**)",
        "write_attributes": "(**)"
    }
}
]
}
}

```

5. Operations and Maintenance

5.1 Data Storage

The paypal_credit_messaging_ocapi integration requires System Objects Extension to store payment-related information.

OrderPaymentInstrument custom properties:

- **paypalOrderId** - Result of a payment action (Auth, Order, Sale).
- **currentPaypalEmail** - The reference ID for PayPal API calls.
- **PP_API_ActiveBillingAgreement** - The PayPal Payer ID in the PayPal service.

Profile custom properties:

- **PP_API_billingAgreement** - Array of saved billing agreements. Each billing agreement is an object with balID, email, default keys

Order custom properties:

- **paypalPaymentMethod** - Property to differentiate PayPal related orders.

Customer Payment Instrument custom properties:

- **paypalBillingAgreementEmail** - Property to differentiate PayPal related payment instruments (Billing Agreement Account Email).
- **paypalBillingAgreementID** - Property to differentiate PayPal related payment instruments (Billing Agreement ID).
- **paypalBillingAgreementToken** - Property to differentiate PayPal related payment instruments (Billing Agreement Token).

5.2 Logs

This integration introduces three new custom logs:

1. **PayPal Storefront Custom logs** - starts with prefix custom-PayPal-blade2-2-appserver-20150722.log. This Log file contains all errors related information in the **paypal_credit_messaging_ocapi** cartridge.
2. **PayPal Business manager custom logs** - starts with prefix custom- PayPal-BM-blade. This Log file contains all errors related information in the **bm_paypal** cartridge.

3. **Service communication logs** - starts with service-PayPalRest. These logs contain every request and response to the PayPal endpoint. To enable these logs, check [Adding API Credentials](#) **Optional** section.

5.3 HTTP Service Availability

You can track availability and downtime by service status in the Commerce Cloud Business Manager. Go to **Administration > Operations > Service Status > int_paypal.http.rest**.

You can configure options for HTTP calls to REST API related to this PayPal integration via the Commerce Cloud Service Profile Interface. To do this, go to **Administration > Operations > Services > Service Profiles -> PayPal_Default_Profile**. You can set a timeout for all requests, enable the Circuit Breaker mechanism, and adjust the Rate Limit.

Administration > Operations > Services > Service Profiles > PayPal_Default_Profile - Details

PayPal_Default_Profile

Fields with a red asterisk (*) are mandatory. Click **Apply** to save the details. Click **Reset** to revert to the last saved state.

This profile is used by 1.00 service.

Name: *	PayPal_Default_Profile
Timeout (ms):	60,000
Enable Circuit Breaker:	<input type="checkbox"/>
Max Circuit Breaker Calls:	0
Circuit Breaker Interval (ms):	0
Enable Rate Limit:	<input type="checkbox"/>
Max Rate Limit Calls:	0
Rate Limit Interval (ms):	0

Figure 7. PayPal Service Profile Settings

5.4 Testing

You must obtain your own test account on the [PayPal Developer Portal](#). If you can't create your own PayPal Sandbox account, contact PayPal support.

5.5 Support

To get help and support from PayPal:

- **PayPal Business support** – Go to PayPal's [Contact Us](#) and log in to your PayPal account.

- **Technical Support** – Go to the [Merchant Technical Support Help Center](#).

6. User Guide

6.1 Business Manager modules

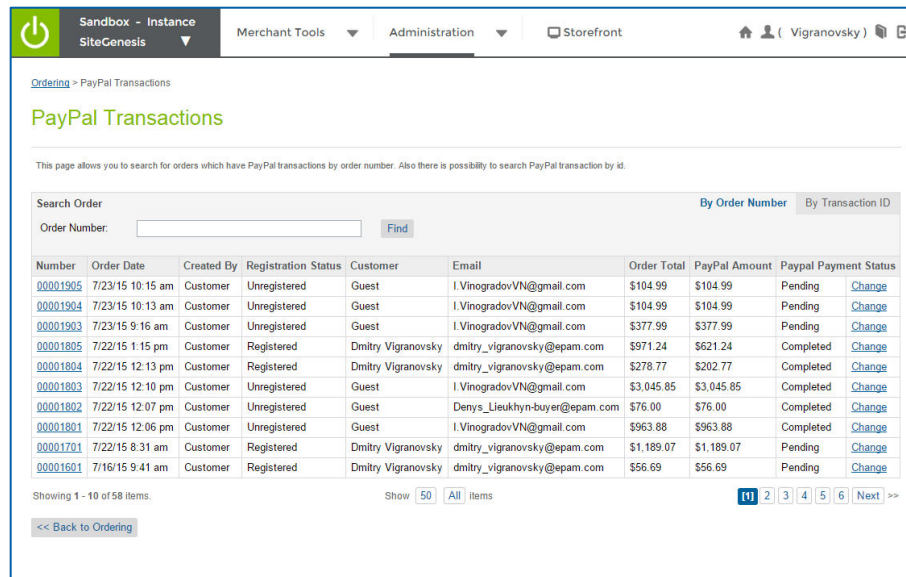
6.1.1 PayPal Transactions

This integration introduces a new Business Manager function, PayPal Transactions. See [Papal Transaction Business Manager module configuration](#) for a detailed description of how to grant access to the PayPal Transactions page.

The PayPal Transactions extension allows you to manage PayPal transactions assigned to Commerce Cloud Orders by the Commerce Cloud Business manager. PayPal Transactions module supports reauthorization (after 3 days), capture and partial capture, void, refund.

The main PayPal Transactions page displays all orders in a site that were paid or partially paid by PayPal. You can access this page in the Ordering menu by selecting **Business Manager > Merchant Tools > Ordering > PayPal Transactions**.

This page displays various information from the order record:



Number	Order Date	Created By	Registration Status	Customer	Email	Order Total	PayPal Amount	Paypal Payment Status
00001905	7/23/15 10:15 am	Customer	Unregistered	Guest	I.VinogradovVN@gmail.com	\$104.99	\$104.99	Pending Change
00001904	7/23/15 10:13 am	Customer	Unregistered	Guest	I.VinogradovVN@gmail.com	\$104.99	\$104.99	Pending Change
00001903	7/23/15 9:16 am	Customer	Unregistered	Guest	I.VinogradovVN@gmail.com	\$377.99	\$377.99	Pending Change
00001805	7/22/15 1:15 pm	Customer	Registered	Dmitry Vigranovsky	dmitry_vigranovsky@epam.com	\$971.24	\$621.24	Completed Change
00001804	7/22/15 12:13 pm	Customer	Registered	Dmitry Vigranovsky	dmitry_vigranovsky@epam.com	\$278.77	\$202.77	Completed Change
00001803	7/22/15 12:10 pm	Customer	Unregistered	Guest	I.VinogradovVN@gmail.com	\$3,045.85	\$3,045.85	Completed Change
00001802	7/22/15 12:07 pm	Customer	Unregistered	Guest	Denys_Lieukhyn-buyer@epam.com	\$76.00	\$76.00	Completed Change
00001801	7/22/15 12:06 pm	Customer	Unregistered	Guest	I.VinogradovVN@gmail.com	\$963.88	\$963.88	Completed Change
00001701	7/22/15 8:31 am	Customer	Registered	Dmitry Vigranovsky	dmitry_vigranovsky@epam.com	\$1,189.07	\$1,189.07	Pending Change
00001601	7/16/15 9:41 am	Customer	Registered	Dmitry Vigranovsky	dmitry_vigranovsky@epam.com	\$56.69	\$56.69	Pending Change

Figure 8. PayPal Transactions Business Manager Main Page Interface

You can search Orders placed within PayPal order number (as shown in Figure 9) or by PayPal Transaction ID (as shown in Figure 10).

The screenshot shows the 'PayPal Transactions' page in a Commerce Cloud environment. The top navigation bar includes 'Sandbox - Instance SiteGenesis', 'Merchant Tools', 'Administration', and 'Storefront'. The user is logged in as 'Vigranovsky'. The page title is 'PayPal Transactions'. Below the title, a message states: 'This page allows you to search for orders which have PayPal transactions by order number. Also there is possibility to search PayPal transaction by id.' There are two tabs: 'By Order Number' (selected) and 'By Transaction ID'. The search form has a 'Search Order' label and an input field for 'Order Number' containing '00001803', with a 'Find' button. Below the search form is a table with the following data:

Number	Order Date	Created By	Registration Status	Customer	Email	Order Total	PayPal Amount	Paypal Payment Status
00001803	7/22/15 12:10 pm	Customer	Unregistered	Guest	I.VinogradovVN@gmail.com	\$3,045.85	\$3,045.85	Completed Change

Below the table, it says 'Showing 1 - 1 of 1 items.' and there is a '<< Back to Ordering' button. A small blue box with '1' is in the bottom right corner.

Figure 9. Search Orders with PayPal Transaction by Commerce Cloud Order Number

The screenshot shows the 'PayPal Transactions' page with the 'By Transaction ID' tab selected. The search form has a 'Search Order' label and an input field for 'Transaction ID' containing '18550924TK545642L', with a 'Find' button. Below the search form is a table with the following data:

Number	Order Date	Created By	Registration Status	Customer	Email	Order Total	PayPal Amount	Paypal Payment Status
00001803	7/22/15 12:10 pm	Customer	Unregistered	Guest	I.VinogradovVN@gmail.com	\$3,045.85	\$3,045.85	Completed Change

Below the table, it says 'Showing 1 - 1 of 1 items.' and there is a '<< Back to Ordering' button. A small blue box with '1' is in the bottom right corner.

Figure 10. Search Orders with PayPal Transaction by Transaction ID

You can see Transaction Details from the PayPal Payment Transaction of the actual order by clicking **Order Number** or the **Change** link on the right. If an order has more than one related transaction, you'll see a selection box with all the transactions that are related to the current order (as shown in Figure 11).

ins by order number. Also there is possibility to search PayPal transaction by id.

Find

Transactions of order: 00015207

Transaction ID: 7AE52170XT8052505 Other transactions: 7AE52170XT8052505

Name: Ivan C Vinogradov

Transaction Email: I.VinogradovVN@gmail.com

Amount: 244.63 USD Captured: 244.63 Refunded: 243.63

Shipping: 0.00 USD

Invoice ID: 00015207

Order Date: 7/22/20 1:45 pm

Payment Status: CAPTURED

Shipping Address: Ezio Auditore
Rome, Rome
Rome, US 12345
DE

	Total	PayPal Amc
Customer	29	\$153.29
Customer	03	\$778.03
Customer	6.90	\$2,066.90
Merchant		\$1.00
Customer	3.04	\$1,093.04
Customer	34	\$218.34
Customer	3.03	\$2,143.03
Customer	63	\$244.63
Customer	63	\$244.63
Customer	63	\$244.63

Figure 11. Transaction Details Popup

order number. Also there is possibility to search PayPal transaction by id.

Find

Transactions of order: 00015207

Transaction ID: 7AE52170XT8052505 Other transactions: 7AE52170XT8052505

Name: Ivan C Vinogradov

Transaction Email: I.VinogradovVN@gmail.com

Amount: 244.63 USD Captured: 244.63 Refunded: 243.63

Shipping: 0.00 USD

Invoice ID: 00015207

Order Date: 7/22/20 1:45 pm

Payment Status: CAPTURED

Shipping Address: Ezio Auditore
Rome, Rome
Rome, US 12345
DE

7AE52170XT8052505
Captured: 7GX80784LM544511C
Captured: 3D495099BU1827429

	Total	PayPal An
Customer	29	\$153.29
Customer	03	\$778.03
Customer	6.90	\$2,066.90
Merchant		\$1.00
Customer	3.04	\$1,093.04
Customer	34	\$218.34
Customer	3.03	\$2,143.03
Customer	63	\$244.63
Customer	63	\$244.63
Customer	63	\$244.63

Figure 12. PayPal Transactions Selected for Their Relationships to Commerce Cloud Order Transactions

Depending on the transaction type and status, the order may have the following action buttons

- Capture (Figure 13)
- Void (Figure 13 and Figure 14)
- Issue Refund (Figure 15)
- Reauthorize (Figure 14)

After click on a button pop-up with details will appear

- Capture Form (Figure 17)
- Void Form (Figure 16)
- Issue Refund From (Figure 18)
- Reauthorize From (Figure 18)

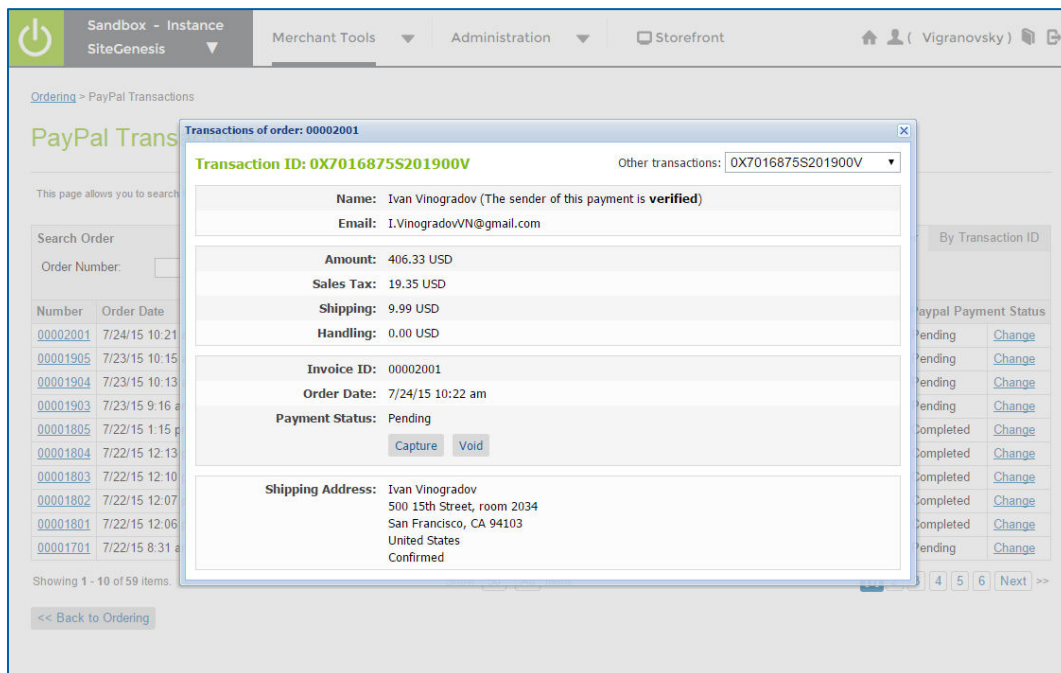


Figure 13. Authorized transaction payment actions

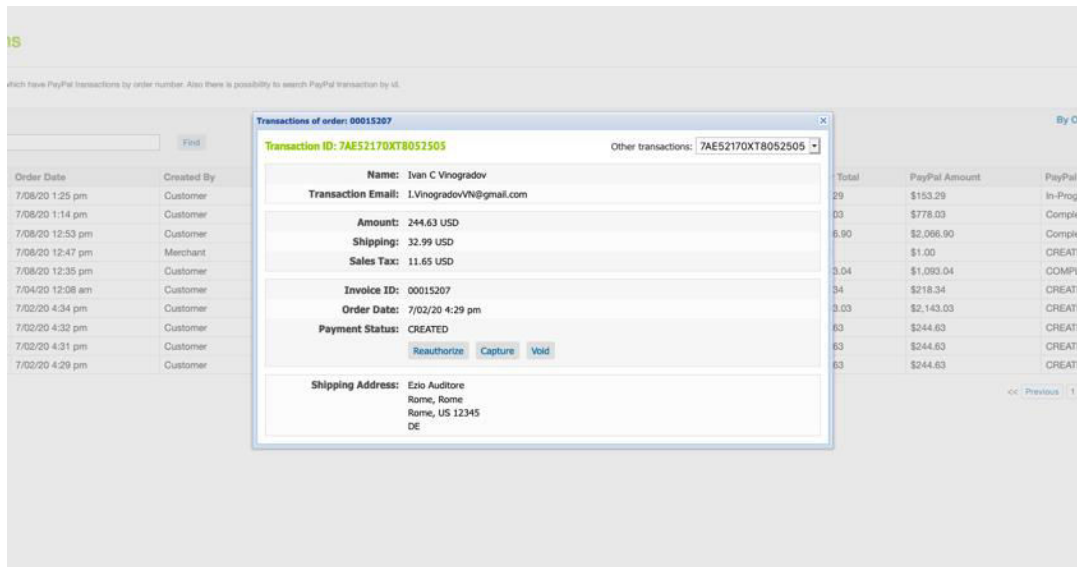


Figure 14. Authorized transaction payment actions (after 3 days)

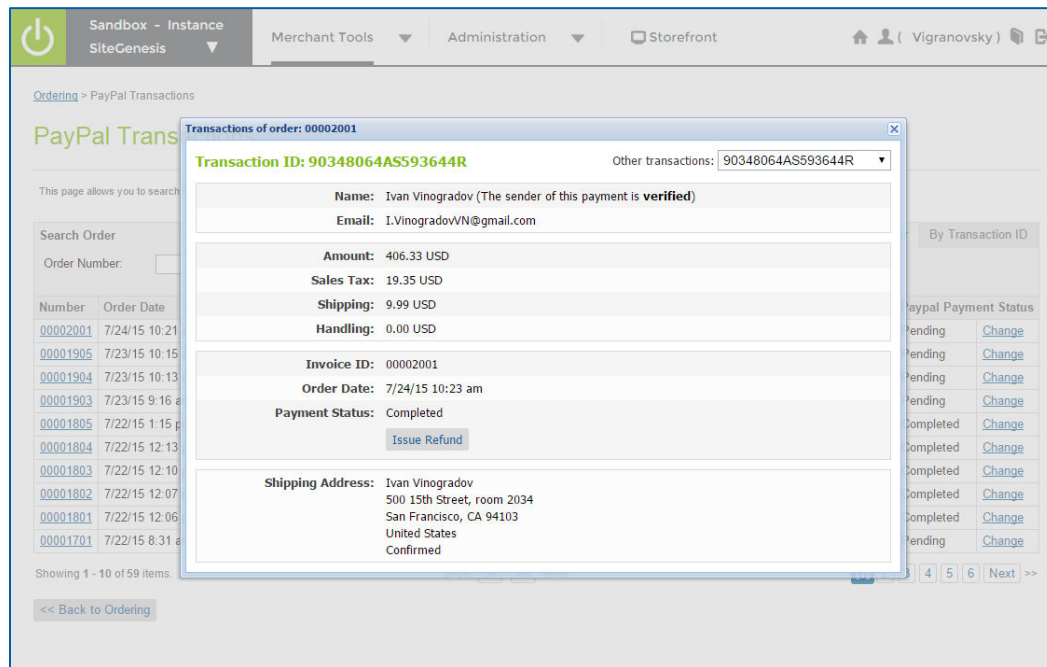


Figure 15. Captured transaction payment actions

The screenshot shows a merchant interface with a 'PayPal Transactions' page. A modal window titled 'Void Authorization' is open, displaying the following information:

- Authorization ID:** 0X7016875S201900V
- Name:** Ivan Vinogradov
- Email:** I.VinogradovVN@gmail.com
- Amount to Void:** 406.33 USD
- Note to Buyer:** A text area containing 'lorem ipsum lor...' with a '236 characters left' indicator.

At the bottom of the modal, there is a 'Submit' button and a 'Cancel' button. The background shows a list of transactions and a 'Paypal Payment Status' table.

Figure 16. Void Authorization Form

The screenshot shows the same merchant interface with a 'Capture Funds' modal window open. The information displayed is:

- Authorization ID:** 0X7016875S201900V
- Name:** Ivan Vinogradov
- Email:** I.VinogradovVN@gmail.com
- Authorization Amount:** 406.33 USD
- Capture Amount:** A text input field containing '406.33' followed by 'USD'.
- Note to merchant:** A paragraph of text explaining the capture process and recommending a note to the buyer.
- Note to Buyer:** A text area with '(optional)' and a '255 characters left' indicator.

The modal includes 'Submit' and 'Cancel' buttons at the bottom. The background interface remains the same as in Figure 16.

Figure 17. Capture Form

Issue Refund

Transaction ID: 90348064A5593644R

Name: Ivan Vinogradov

Email: I.VinogradovVN@gmail.com

Original payment: 406.33 USD

Refund amount: 406.33 USD

Invoice ID: 00002001

Note to Buyer: (optional)

255 characters left

Submit Cancel

Figure 18. Issue Refund Form

Reauthorize Payment

Transaction ID: 7AE52170XTB052505

Name: Ivan C Vinogradov

Transaction Email: I.VinogradovVN@gmail.com

Authorization ID: 7AE52170XTB052505

Name: Ivan C Vinogradov

Email: I.VinogradovVN@gmail.com

Amount to Reauthorize: 244.63 USD

Submit Cancel

Figure 19 Reauthorize form

6.1.2 PayPal Styles Configuration

PayPal Styles Configuration allow to choose styling of the Smart Button from the Business Manager. After installation new menu item will appear in under Site Preferences in the Business Manager (Figure 20).

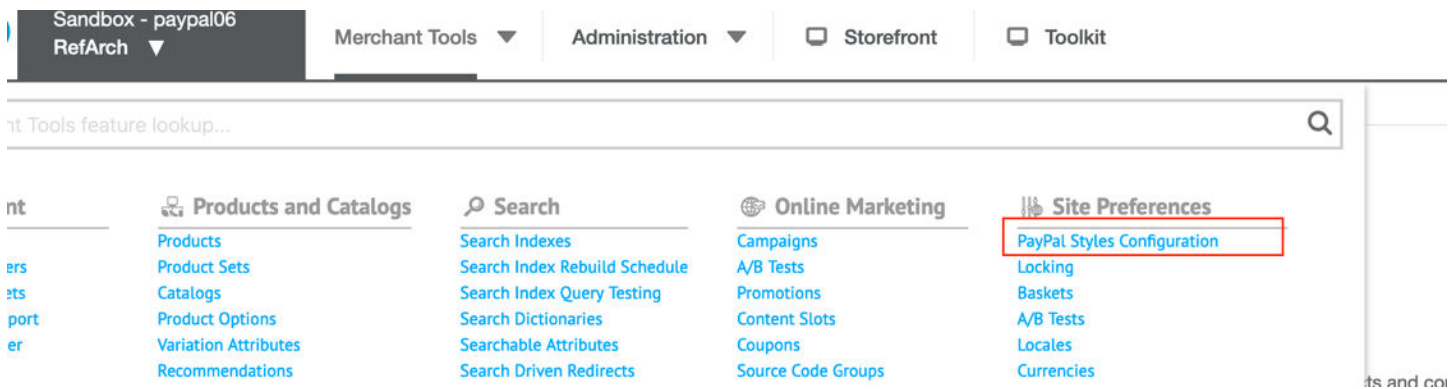


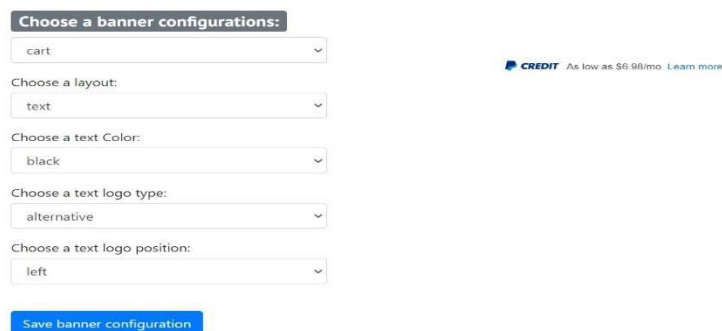
Figure 20. PayPal Styles Configuration in the Business Manager

If paypal_credit_messaging_ocapi cartridge was installed properly you should see styling options for the PayPal Smart Button. Depends on buttons locations will be for Billing page or for Cart page. To read more about styling options visit [Customize the PayPal Buttons](#). After configuration done click **Save Smart Button Configuration** to update styling (Figure 21.1.).



Figure 21.1. Smart Button styling options

Also, you can customize the properties of Credit Banners using PayPal Styles



Configuration (Figure 21.2.).

Figure 21.2. Credit Banners styling options

6.2 Storefront Functionality

Integration has 3 types of checkout with PayPal. All of them are available from payment page or cart page.


NOTE: Checkout from cart requires customer data from PayPal. Review [PayPal-provided Billing Address and Phone Number](#) before enable PayPal button on a cart page.

1. One-time checkout. Available for both guest and registered user types if billing agreement is disabled in Custom Preferences.
2. Checkout with billing agreement creation. Available for both guest and registered user types if billing agreement is enabled in Custom Preferences.
3. Checkout using saved billing agreement. Available for registered user who already path checkout once with billing agreement and saved it.

NOTE: Reference transaction feature must be enabled at the merchant account setting to activate flow 2 and 3. Review [Billing Agreement and Reference Transactions](#) for more details.

Your Cart

Continue Shopping 1 Items Need Help? Call 1-800-555-0199

Turquoise and Gold Necklace			
	Color: Gold In Stock	Each \$45.00	Quantity 1
			Total \$45.00

Enter Promo Code

Promo Code Submit

Shipping

Ground (7-10 Business Days)

Shipping cost \$15.99

Sales Tax \$3.05

Estimated Total \$64.04

CREDIT Pay over 24 months. Min purchase required.

PayPal Checkout

Debit or Credit Card

Powered by **PayPal**

Checkout

Figure 22 Checkout from the cart page using Smart Button (Example)

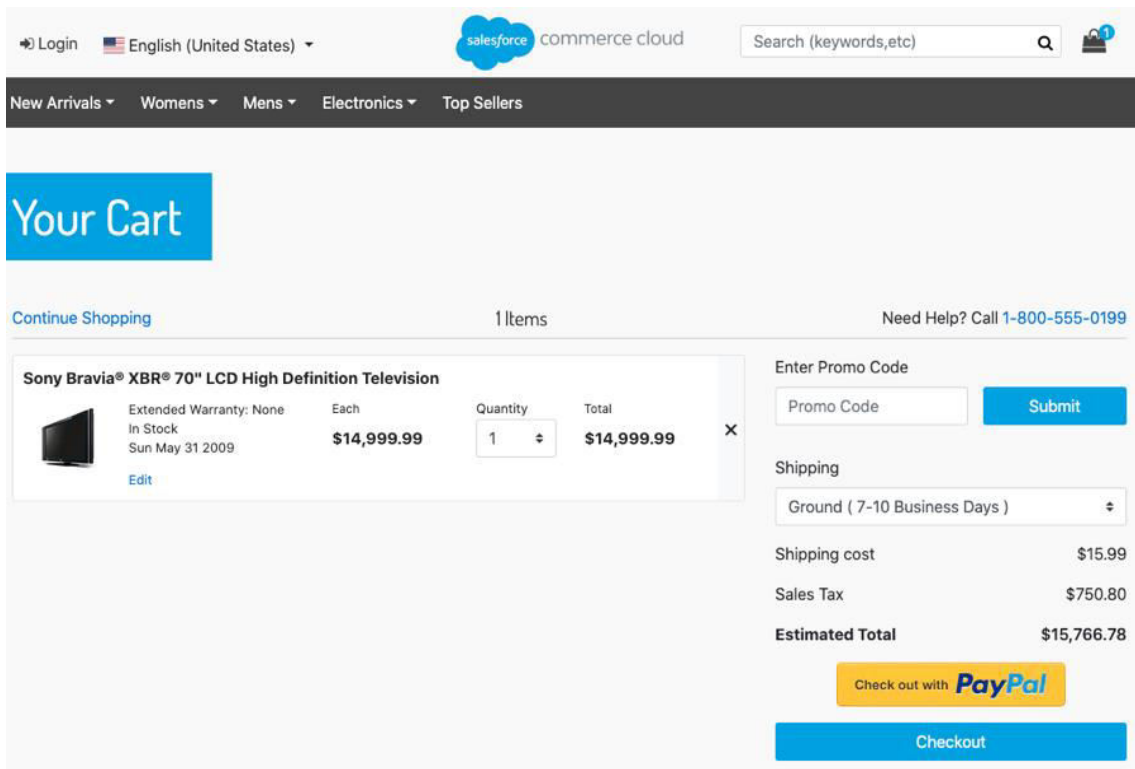


Figure 23. Checkout from the cart page using static image (Example)

7. Multi-site and multi-credential support

7.1 General info

PayPal cartridges doesn't come with multi-site or multi-credential support out of the box. On service level build in logic will use only `int_paypal.http.rest` service and credential attached to the service to communicate PayPal REST API.

8. Open commerce API Hooks

8.1 Ocapi Functionality (Ocapi Hooks)

8.1.1 Basic Variable's description.

This section describes the basic variables used when working with OCAPI requests.

The base URL looks like this example:

```
{{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/customers/auth
```

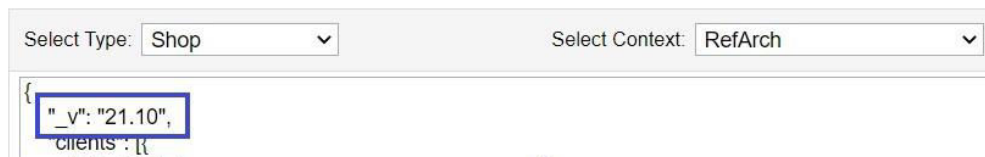
Where:

- `{{host}}` - the part of the URL address that is responsible for the host address.
Example: <https://zpz-013.sandbox.us01.dx.commercecloud.salesforce.com>
- `{{site}}` - the part of the URL address that is responsible for the site name.
Example: RefArch
- `{{ocapi_version}}` - the part of the URL address that is responsible for version of OCAPI you are using. Example: v21_10 (to get this value you should go to Administration > Site Development > Open Commerce API Settings choose your site and find "_v" in your settings)

[Administration](#) > [Site Development](#) > Open Commerce API Settings

Open Commerce API Settings

This page allows you to make client application-specific configurations of Open Commerce API resources, i.e. manage resource access, configure and specify whether your settings are site-specific or global (organization-wide). Please note that due to caching, changes may not be visible immediately. You can browse the Open Commerce API here [API Explorer](#).



```
{
  "_v": "21.10",
  "clients": {
    ...
  }
}
```

Figure 24. Ocap version location

8.1.2 A Registered and an unregistered user authentication.

To get the necessary information to work with registered and unregistered user (Bearer Token, customer_id), first you need to provide a next request:

- **Method:** POST
- **URL:** `{{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/customers/auth`
(The variables used in URL are described in paragraph 8.1.1. **Basic variable's description**)
- **Authorization:** you should use authorization type Basic Auth (using Username and Password of your customer) for registered user auth.
- **Headers:**

1. `x-dw-client-Id`

To get this value you should go to Administration > Site Development > Open Commerce API Settings choose your site and find "client_id" in your settings

(for example in this case we have test value -
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa)

RefArch Merchant Tools Administration Storefront Toolkit

This page allows you to make client application-specific configurations of Open Commerce API resources, i.e. manage resource access privilege configure and specify whether your settings are site-specific or global (organization-wide). Please note that due to caching, changes may take You can browse the Open Commerce API here [API Explorer](#).

Select Type: Shop Select Context: RefArch

```
{
  "_v": "21.10",
  "clients": [{
    "client_id": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
    "allowed_origins": [],

```

Figure 23. Ocapi client_id location

2. Content-Type (value - application/json)

- **Body:**

for registered user request body should contain next parameter:

```
{
  "type": "credentials"
}
```

For unregistered user request body should contain next parameter:

```
{
  "type": "guest"
}
```

In a case of success, you will get response with information about your customer, list with current billing agreements, SDK links for PayPal button rendering, and pdpPayPalConfigs object (depends

```
{
  "_type": "customer",
  "auth_type": "registered",
  "creation_date": "2022-07-20T15:28:05.000Z",
  "customer_id": "acq2CIzgKBHwS1aHzHv5qpiTt5",
  "customer_no": "00018503",
  "email": "mishapapash@gmail.com",
  "enabled": true,
  "first_name": "Mykhailo",
  "hashed_login": "c3c6d62ff2e1b8760b501c0d3633fa42dfa046dc40283a1463e1c3dd18422010",
  "last_login_time": "2022-07-21T13:52:41.459Z",
  "last_modified": "2022-07-21T13:52:41.459Z",
  "last_name": "Papash",
  "last_visit_time": "2022-07-21T13:52:41.459Z",
  "login": "mishapapash@gmail.com",
  "phone_home": "9234567890",
  "previous_login_time": "2022-07-21T13:50:58.509Z",
  "previous_visit_time": "2022-07-21T13:50:58.509Z",
  "visit_id": "ce20a7c91a77b7b355d30563cc",
  "c_PP_API_billingAgreement": "[{"baID": "B-4DK448552V8546235", "email": "p.lorenstest@hotline.com", "default": true}],
  "c_paypal": {
    "billingAgreementEnabled": true,
    "isBAlimitReached": false,
    "sdkUrl": "https://www.paypal.com/sdk/js?client-id=AdYw0mYpZkz6qk3RNTmDTDAAnNhWwUpL_zawBcv7wjinBmcm9b-10rKLRDwmRUzjc0wScbT9xDsi0dvAu&commit=false&vault=true&disable-funding=card,credit",
    "paypalUrls": [{"billingSdkUrl": "https://www.paypal.com/sdk/js?client-id=AdYw0mYpZkz6qk3RNTmDTDAAnNhWwUpL_zawBcv7wjinBmcm9b-10rKLRDwmRUzjc0wScbT9xDsi0dvAu&commit=false&vault=true&currency=USD&disable-funding=sepa,bancontact,eps,giropay,ideal,mybank,p24,sofort", "cartSdkUrl": "https://www.paypal.com/sdk/js?"}
  ]
}
```


on Custom Preferences configurations) for registered user:

Figure 25. Ocapi auth response for registered user

pdpPayPalConfigs - an object contains all necessary data for rendering PayPal smart button on the Product (Pdp) page.

```
29 | "pdpPayPalConfigs": {
30 |   "partnerAttributionId": "SFCC_EC_B2C_2022_1_0",
31 |   "paypalButtonSdkUrl": "https://www.paypal.com/sdk/js?client-id=AdYw0mYpZkz6qk3RNTmDTDAAnNhWwUpL_zawBcv7wjinBmcm9b-1(
32 |   "paypalEmail": null,
33 |   "showStaticImage": true,
34 |   "defaultBAemail": "ptest4@paypal.com",
35 |   "paypalStaticImageLink": "https://www.paypalobjects.com/webstatic/en_US/i/buttons/checkout-logo-large.png",
36 |   "billingAgreementEnabled": true,
37 |   "buttonConfig": {
38 |     "style": {
39 |       "color": "black",
40 |       "shape": "rect",
41 |       "label": "checkout",
42 |       "size": "medium",
43 |       "layout": "horizontal",
44 |       "tagline": true,
45 |       "height": 40
46 |     }
47 |   }
48 | }
```

Figure 26. pdpPayPalConfigs object example

In a case of success, you will get response with information about your customer for unregistered user:

```
"_v": "22.4",
"_type": "customer",
"auth_type": "guest",
"customer_id": "bc4xaBPFZSUGJwwOC1n13jDeve",
"preferred_locale": "en_GB",
"visit_id": "c4d7fa89e2aa86cf3f03cd47d8"
```

Figure 27. Ocapi auth response for unregistered user

8.1.3 Get banner configuration for category page.

To get banner configuration for category page, you need to provide a next request:

- **Method:** GET
- **URL:** {{host}}/s/Sites-{{site}}-

Site/dw/shop/{{ocapi_version}}/categories/{{product_category_id}}

(The basic variables used in URL are described in paragraph 8.1.1. Basic variable's description, how to get {{customer_id}} look in paragraph 8.1.2. A Registered

and an unregistered user authentication, {{product_category_id}} - ID of the necessary category of products).

- **Headers:**

1. *x-dw-client-Id* (More details look at paragraph 8.1.1)
2. *Content-Type* (value - *application/json*)
3. Authorization (value - {{bearer_token}}, how to get {{bearer_token}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)

In a case of success, you will get response with information about product category with

```
"c_paypalCategoryBannersConfig": {  
  "paypal": {  
    "bannerSdkUrl": "https://www.paypal.com/sdk/js?  
client-id=AWLvojdrG8Ga60b5lmk1SyukGGmE0i0h413Mo5RQ4xvcwFgW8zKeiwITB2BomdVwkM7wYhSKNwV_wiwi&commit=false&components=buttons,  
messages&currency=USD&disable-funding=sepa,bancontact,eps,giropay,ideal,mybank,p24,sofort",  
    "bannerConfig": {  
      "styleColor": "black",  
      "styleRatio": "20x1",  
      "styleLayout": "flex",  
      "styleTextColor": null,  
      "styleLogoPosition": null,  
      "styleLogoType": null  
    },  
    "paypalAmount": null  
  },  
  "creditMessageAvailable": true  
}
```

c_paypalCategoryBannersConfig:

8.1.4 Get banner configuration for product page.

To get banner configuration for product page, you need to provide a next request:

- **Method:** GET
- **URL:** {{host}}/s/Sites-{{site}}-
Site/dw/shop/{{ocapi_version}}/products/{{product_id}}
(The basic variables used in URL are described in paragraph 8.1.1. Basic variable's description, how to get {{customer_id}} look in paragraph 8.1.2. A Registered and an unregistered user authentication, {{product_id}} - ID of the necessary product).
- **Headers:**
 1. *x-dw-client-Id* (More details look at paragraph 8.1.1)
 2. *Content-Type* (value - *application/json*)

3. Authorization (value - {{bearer_token}}, how to get {{bearer_token}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)

In a case of success, you will get response with information about product category with

```
"c_paypalProductBannersConfig": {  
  "paypal": {  
    "bannerSdkUrl": "https://www.paypal.com/sdk/js?  
client-id=AWLvojdrg8Ga60b5lmkiSyukGGmE0i0h413Mo5RQ4xvcwFgw8zKeiwITB2BomdVwkM7wYhSKNwV_wiwi&commit=false&components=buttons,  
messages&currency=USD&disable-funding=sepa,bancontact,eps,giropay,ideal,mybank,p24,sofort",  
    "bannerConfig": {  
      "styleColor": "white",  
      "styleRatio": "20x1",  
      "styleLayout": "flex",  
      "styleTextColor": "black",  
      "styleLogoPosition": "left",  
      "styleLogoType": "primary"  
    },  
    "paypalAmount": null  
  },  
  "creditMessageAvailable": true  
}
```

c_paypalProductBannersConfig:

8.1.5 Create billing agreement token.

Please note: More information on how to render the PayPal button can be read in the documentation at this [link](#). The best way to create a billing agreement token is using PayPal button 'onClick' event.

To create billing agreement token, you need to provide a next request:

- **Method:** GET
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/customers/{{customer_id}}?stage=createBillingAgreementToken
(The basic variables used in URL are described in paragraph 8.1.1. Basic variable's description, how to get {{customer_id}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)
- **Headers:**
 1. x-dw-client-Id (More details look at paragraph 8.1.2)
 2. Content-Type (value - application/json)
 3. Authorization (value - {{bearer_token}}, how to get {{bearer_token}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)

In a case of success, you will get response with information about your which includes billing agreement token (c_billingAgreementToken):

```
"_v": "22.4",
"_type": "customer",
"_resource_state": "9e5445e82943e4a0d0b5006508b0154215d1bed3d1e9af1d3e288c5a9cdfa120",
"auth_type": "registered",
"creation_date": "2022-07-20T15:28:05.000Z",
"customer_id": "acq2CIzgKBHwS1aHzHv5qpiTt5",
"customer_no": "00018503",
"email": "mishapapash@gmail.com",
"enabled": true,
"first_name": "Mykhailo",
"last_login_time": "2022-07-21T17:57:30.247Z",
"last_modified": "2022-07-21T17:57:30.247Z",
"last_name": "Papash",
"last_visit_time": "2022-07-21T17:57:30.247Z",
"login": "mishapapash@gmail.com",
"phone_home": "9234567890",
"previous_login_time": "2022-07-21T17:57:30.247Z",
"previous_visit_time": "2022-07-21T17:57:30.247Z",
"c_PP_API_billingAgreement": "[{\"baID\":\"B-4DK448552V8546235\", \"email\":\"p.lorenstest@hotline.com\", \"default\":true}]",
"c_billingAgreementToken": "BA-1HG23334RA496921T"
```

Figure 28. Get billing agreement token response example

8.1.6 Add billing agreement.

Please note: More information on how to render the PayPal button can be read in the documentation at this [link](#). The best way to process add a billing agreement is using PayPal button 'onApprove' event.

To add billing agreement, you need to provide a next request:

- **Method:** POST
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/customers/{{customer_id}}/payment_instruments
(The basic variables used in URL are described in paragraph 8.1.1. Basic variable's description, how to get {{customer_id}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)
- **Headers:**
 1. x-dw-client-Id (More details look at paragraph 8.1.2)
 2. Content-Type (value - application/json)
 3. Authorization (value - {{bearer_token}}, how to get {{bearer_token}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)
- **Body:** {
 "payment_method_id": "PayPal",

```
    "c_billingAgreementToken": {{BAToken}}
  }
```

(How to get {{BAToken}} look in paragraph 8.1.3. Create billing agreement token)

In a case of success, you will get response with information about your which includes billing agreement token (c_billingAgreementToken):

```
_v : 22.4
_type : customer_payment_instrument
creation_date : 2022-07-21T18:26:49.521Z
last_modified : 2022-07-21T18:26:51.180Z
payment_bank_account : {"_type":"payment_bank_account"}
payment_card : {"_type":"payment_card","credit_card_expired":false}
payment_instrument_id : 76c2f8029b8eac0665713ea3d3
payment_method_id : PayPal
c_paypalBillingAgreementEmail : ptest4@paypal.com
c_paypalBillingAgreementID : B-3BV69582WY240494R
c_paypalBillingAgreementToken : BA-1XY962054E9159844
```

Figure 29. Add billing agreement response example

8.1.7 Display a registered user's payment instruments (billing agreements).

To display the payment instruments of the registered user, you need to provide a next request:

- **Method:** GET
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/customers/{{customer_id}}/payment_instruments
(The basic variables used in URL are described in paragraph 8.1.1. Basic variable's description, how to get {{customer_id}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)
- **Headers:**
 1. *x-dw-client-Id* (More details look at paragraph 8.1.2)
 2. *Content-Type* (value - *application/json*)

3. Authorization (value - {{bearer_token}}, how to get {{bearer_token}} look in paragraph 8.1.2. A Registered and an unregistered user authentication)

In a case of success, you will get response with information about your customer' payment

```
"_v": "22.4",
"_type": "customer_payment_instrument_result",
"count": 1,
"data": [
  {
    "_type": "customer_payment_instrument",
    "creation_date": "2022-07-21T08:03:40.716Z",
    "last_modified": "2022-07-21T08:03:41.849Z",
    "payment_bank_account": {
      "_type": "payment_bank_account"
    },
    "payment_card": {
      "_type": "payment_card",
      "credit_card_expired": false
    },
    "payment_instrument_id": "9e3596bfa77b443567f3ab5be2",
    "payment_method_id": "PayPal",
    "c_paypalBillingAgreementEmail": "p.lorenstest@hotline.com",
    "c_paypalBillingAgreementID": "B-4DK448552V8546235",
    "c_paypalBillingAgreementToken": "BA-38N04150UW0168832"
  }
],
"total": 1
```

instruments which includes billing agreement information:

Figure 30. Get a registered user's payment instrument response example

8.1.8 Delete a registered user's payment instrument (billing agreement)

To delete the payment instrument you need to provide a next request:

- **Method:** DELETE
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/customers/{{customer_id}}/payment_instruments/{{payment_instrument_id}}
(The basic variables used in URL are described in paragraph 8.1.1. **Basic variable's description**, how to get {{customer_id}} look in paragraph 8.1.2. **A Registered and an unregistered user authentication**, {{payment_instrument_id}} you can find in "Figure 28. Get a registered user's payment instrument response example"

- **Headers:**
 1. *x-dw-client-Id* (More details look at paragraph 8.1.2)
 2. *Content-Type* (value - *application/json*)
 3. Authorization (value - `{{bearer_token}}`, how to get `{{bearer_token}}` look in paragraph 8.1.2. **A Registered and an unregistered user authentication**)

In a case of success, you will get 204 "NO CONTENT" response.

To get more information about 'Customers resource Shop API' that uses on the Account page, follow the link:

https://documentation.b2c.commercecloud.salesforce.com/DOC1/index.jsp?topic=%2Fcom.demandware.dochelp%2FOCAPI%2Fcurrent%2Fshop%2FResources%2FCustomers.html&cp=0_16_3_4

9. An order placing due a PayPal payment method by using OCAPI

9.1.1 Create a basket

To create a basket you need to provide a next request:

- **Method:** POST
- **URL:** `{{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/baskets`
(The basic variables used in URL are described in paragraph 8.1.1 **Basic variable's description.**)
- **Headers:**
 1. *Content-Type* (value - *application/json*)
 2. Authorization (value - `{{bearer_token}}`, how to get `{{bearer_token}}` look in 8.1.2 **Registered and an unregistered user authentication**)
 3. Content-length: 0
- **Body:** A basket must contain a valid currency code that is related to the payment method you will use in this basket. Some payment methods support all currencies

of site, but in case a local payment method you need to provide an appropriate currency.

For example: 'LPM' (Local payment method) 'mybank', supports only a 'EUR' currency, in this case to set a valid basket currency you need to provide the following body:

```
{
  "currency": "EUR"
}
```

Figure 31. A create basket body example

Important: If you have not provided a currency code for a basket, the default currency of site will set to the basket, and this can lead an error during the placing order due LPM.

Request example:

```
REQUEST:
POST /dw/shop/v22_4/baskets HTTP/1.1
Host: example.com
Authorization: Bearer eyJfdiI6IjXXXXXX.eyJfdiI6IjEiLCJleHA5XXXXXX.-d5wQW4c404wt-2k17_fiEiALW1XXXX
Content-Type: application/json
Content-Length: 0
```

Figure 32. A create basket request example

In a case of success, you will get response 200 OK, with information about a new created basket with its ID and resource state:


```

RESPONSE:
HTTP/1.1 200 OK
Content-Length:124
Content-Type:application/json;charset=UTF-8
{
  "v" : "22.4",
  "_resource_state" : "860cde3040519cce439cd99e209f8a87c3ad0b7e2813edbf6f5501f763b73bd5",
  "_type" : "basket",
  "_flash" :
  [
    ...
  ]
  ...
  "basket_id" : "bczFTaOjgEqUkaaadvHwbgrP5",
  "currency" : "USD",
  "customer_info" :
  {
    "_type" : "customer_info",
    "customer_id" : "adNJrbxJovaT5DPxUSfOywk6Et",
    "email" : ""
  },
  ...
  "order_total" : 0.00,
  "product_sub_total" : 0.00,
  "product_total" : 0.00,
  "shipments" :
  [
    {
      "_type" : "shipment",
      "id" : "me",
      "shipment_id" : "bc50TaOjgEqUoaaadvHwbgrP5"
    }
  ],
  "shipping_items" :
  [
    {
      ...
      "shipment_id" : "me",
      "item_id" : "bcwsbaOjgEqUsaaadvHwbgrP5"
    }
  ],
  "shipping_total" : 0.00,
  "shipping_total_tax" : 0.00,
  "taxation" : "net",
  "tax_total" : 0.00
}

```

Figure 33. A create basket response example

9.1.2 The basket's flashes

To find the detail information about basket flashes follow the link: [OCAPI flash](#)

```

    "_v": "22.4",
    "_type": "basket",
    "_resource_state": "797908ba3779070eab18aa40f737d39bae1958489627b77a2aeeec9e16c67b593",
    "_flash": [
      {
        "_type": "flash",
        "type": "ProductItemsRequired",
        "message": "Product items are required.",
        "path": "$.product_items"
      },
      {
        "_type": "flash",
        "type": "PaymentMethodRequired",
        "message": "No payment method ID was specified. Please provide a valid payment method ID.",
        "path": "$.payment_instruments[0].payment_method_id"
      },
      {
        "_type": "flash",
        "type": "BillingAddressRequired",
        "message": "No billing address was specified. Please provide a valid billing address.",
        "path": "$.billing_address"
      },
      {
        "_type": "flash",
        "type": "ShippingAddressRequired",
        "message": "No shipping address was specified. Please provide a valid shipping address.",
        "path": "$.shipments[0].shipping_address",
        "details": {
          "shipmentId": "me"
        }
      },
      {
        "_type": "flash",
        "type": "CustomerEmailRequired",
        "message": "Customer Email is required",
        "path": "$.customer_info.email"
      }
    ]
  },
  ]

```

!Important. To place an order, all the basket flashes must be resolved.

Figure 34. The Basket's flashes example

9.2 Add an item (product) to the basket

To add an item (product) to the basket you need to provide a next request:

- **Method:** POST
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/basket/{basket_id}/items (The basic variables used in URL are described in paragraph 8.1.1 Basic variable's description)
- **Headers:**
 1. *Content-Type* (value - *application/json*)
 2. *Authorization* (value - {{bearer_token}}, how to get {{bearer_token}} look in 8.1.2 Registered and an unregistered user authentication)
 3. *Content-length:* 0
- **Body (example):**

```

... [{
...   "product_id" : "008884303989M",
...   "quantity" : 1.00
... }]

```

Figure 35. 'Add an item (product) to a basket body example

Request example:

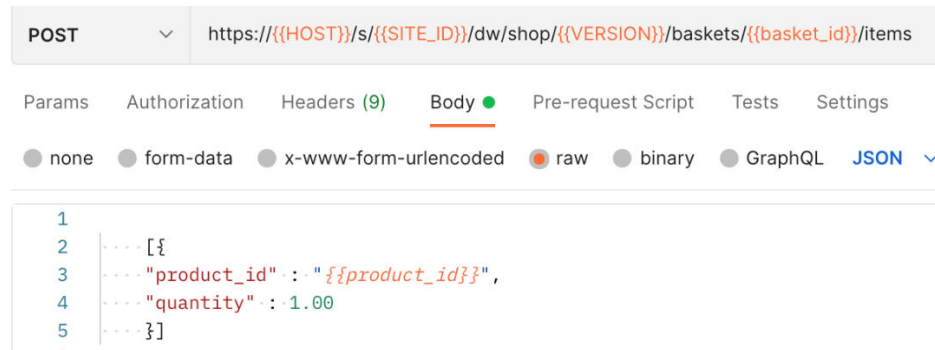


Figure 36. 'Add an item (product) to a basket request example

There are no custom properties for 'Add an item(product) to a basket' request.

In case of success:

```

"product_items": [
  {
    "_type": "product_item",
    "adjusted_tax": 5.00,
    "base_price": 99.99,
    "bonus_product_line_item": false,
    "gift": false,
    "item_id": "9912b66558da7300d228ade082",
    "item_text": "Platinum Blue Stripes Easy Care Fitted Shirt ",
    "price": 99.99,
    "price_after_item_discount": 99.99,
    "price_after_order_discount": 99.99,
    "product_id": "008884304009M",
    "product_name": "Platinum Blue Stripes Easy Care Fitted Shirt ",
    "quantity": 1,
    "shipment_id": "me",
    "tax": 5.00,
    "tax_basis": 99.99,
    "tax_class_id": "standard",
    "tax_rate": 0.05
  }
],

```

Figure 37. 'A snippet of 'Add an item (product) to a basket' response

9.3 Sets the billing address of a basket

To set the billing address to a basket you need to provide a next request:

- **Method:** PUT
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/basket/{basket_id}/billing_address (The basic variables used in URL are described in paragraph **8.1.1 Basic variable's description**)
- **Headers:**
 1. *Content-Type* (value - *application/json*)
 2. *Authorization* (value - {{bearer_token}}, how to get {{bearer_token}} look in **8.1.1 Registered and an unregistered user authentication**)
 3. *Content-length:* 0
- **Body (example):**

```
{
  "first_name": "Jon",
  "last_name": "Smith",
  "city": "Boston",
  "country_code": "US",
  "address1": "1 Main St",
  "phone": "4088161358",
  "postal_code": "95131",
  "state_code": "CA"
}
```

Figure 38 'Sets the billing address to a basket' body example

In case when you use a LPM (Local payment method), make sure that you have provided an appropriate 'country_code'.

For example: LPM (Local payment method) 'mybank', is connected only with Italia, in this case a 'country_code' must be set to 'IT'. In case you have not provided an appropriate 'country_code' this can lead an error during the placing order due LPM.

If one of the required fields will not provide into 'Ocapi' request, the Hook will return an error.

Request example:

```
REQUEST:
PUT /dw/shop/v22_4/baskets/cdTWMiWbOhGJgaaadkIKbj5op9/billing_address HTTP/1.1
Host: example.com
Authorization: Bearer af7f5c90-ffc1-4ea4-9613-f5b375b7dc19
Content-Type: application/json
{
  "first_name": "John",
  "last_name": "Smith",
  "city": "Boston",
  "country_code": "US",
  "c_strValue": "cTest"
}
```

Figure 39 ‘Sets the billing address to a basket’ request example

There are no custom properties for ‘Sets the billing address to a basket’ request.

In case of success:

```
RESPONSE:
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8
{
  "v": "22.4",
  "resource_state": "860cde3040519cce439cd99e209f8a87c3ad0b7e2813edbf6f5501f763b73bd5",
  ...
  "billing_address": {
    "type": "order_address",
    "city": "Boston",
    "country_code": "US",
    "first_name": "John",
    "full_name": "John Smith",
    "last_name": "Smith",
    "c_strValue": "cTest"
  },
  ...
}
```

Figure 40. ‘Sets the billing address to a basket’ response example

9.4 Update a shipment of basket

A shipment will be already created by ‘Create a basket’ request with the default values. To fill it with all necessary data you need to provide a next request:

- **Method:** PATCH
- **URL:** {{host}}/s/Sites-{{site}}-
Site/dw/shop/{{ocapi_version}}/basket/{basket_id}/shipments/{shipping_id} (The basic variables used in URL are described in paragraph 8.1.1 Basic variable’s description.
A shipping_id value will be included inside ‘Create a basket’ response: response.shipments[0].shipping_items.shipment_id; A default value is ‘me’.
- **Headers:**
 1. *Content-Type* (value - *application/json*)

2. Authorization (value - {{bearer_token}}, how to get {{bearer_token}} look in

8.1.2 A Registered and an unregistered user authentication)

3. Content-length: 0

- Body (example):

```
    {
      "shipping_method":
      {
        "id": "002"
      },
      "shipping_address":
      {
        "address1": "1 Main St",
        "phone": "408-972-7588",
        "state_code": "CA",
        "postal_code": "95131",
        "first_name": "John",
        "last_name": "Smith",
        "city": "Boston",
        "country_code": "US"
      }
    }
```

Figure 41 'Update a shipment of basket' body example

Shipping_method.id - the shipping method id from your shop - required.

If one of the required fields will not provide into OCAPI request, the Hook will return an error.

9.5 Sets a customer email for an existing basket

To set an email address to a basket you need to provide a next request:

- **Method:** PUT
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/basket/{basket_id}/customer (The basic variables used in URL are described in paragraph 8.1.1 Basic variable's description)
- **Headers:**
 1. *Content-Type* (value - *application/json*)
 2. *Authorization* (value - {{bearer_token}}, how to get {{bearer_token}} look in 8.1.2 Registered and an unregistered user authentication)
 3. *Content-length:* 0
- **Body (example):**

```
... {  
... "email": "valid@email.com"  
... }
```

Figure 42 'Sets a customer email for an existing basket' a body example

An email address from checkout process - required.

If one of the required fields will not provide into OCAPI request, the Hook will return an error.

Request example:

```
REQUEST:  
PUT /dw/shop/v22_4/baskets/cdCxIiWbPdIKAAAadhKTtczLvK/customer HTTP/1.1  
Host: example.com  
Authorization: Bearer af7f5c90-ffcl-4ea4-9613-f5b375b7dc19  
Content-Type: application/json  
{  
  "customer_no": "customer0012",  
  "email": "valid@email.com"  
}
```

Figure 43 'Sets a customer email for an existing basket' a request example

In case of success:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
{
  "_v" : "22.4",
  "_resource_state" : "860cde3040519cce439cd99e209f8a87c3ad0b7e2813edbf6f5501f763b73bd5",
  ...
  "customer_info" :
  {
    "_type" : "customer_info",
    "customer_id" : "bc42Gjva2LWAZVEvE6Mv1bX6GD",
    "customer_no" : "customer0012",
    "email" : "valid@email.com"
  },
  ...
}
```

Figure 44. Sets a customer email for an existing basket 'a response example

9.6 Add a payment instrument to the basket

To add a payment instrument to the basket you need to provide a next request:

- **Method:** POST
- **URL:** {{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/basket/{basket_id}/payment_instruments
(The basic variables used in URL are described in paragraph 8.1.1 Basic variable's description.

Headers:

1. *Content-Type* (value - *application/json*)
2. *Authorization* (value - {{bearer_token}}, how to get {{bearer_token}} look in 8.1.2 Registered and an unregistered user authentication)
3. *Content-length:* 0

Body request:

Flow with a saved PayPal billing agreement body creation (Works only in case if 'PP_API_BA_ENABLED' site preference is enabled):

To add a Saved payment instrument to the basket, you need to transfer the following properties:

- **amount** - the order total amount. This is required parameter. Type - Number.
- **payment_method_id** - the payment method ID. Always use 'PayPal' for adding PayPal payment method. This is required parameter. Type - String.

- **c_PP_API_ActiveBillingAgreement** - JSON string value, that contains: baID, email and default (indicated whether a current billing agreement is default) values of current billing agreement. This is required parameter. Type - String.

You can get a list of the saved billing agreements of customer by using the following request: **8.1.2 Registered and an unregistered user authentication**.

- **c_paymentId** - you can configurate PayPal JavaScript SDK to see additional payment methods on your site by using the site preferences from 'Paypal_Checkout' site preferences group. The following paymentId' are supported: 'Venmo', 'PayPal Debit/Credit Card', and all local payment method provided into the site preference - PP_API_APM_methods (Available Alternative Payment Methods). To enable 'Venmo' payment method use the following site preference: PP_API_Venmo_Enabled. In case of using one of that payment methods, please provide it into the c_paymentId custom attribute.

This is required parameter. Type - String.

More information about JavaScript SDK script configuration here: [JavaScript SDK script configuration](#)

```
{
  .... "amount": {{order_total}},
  .... "payment_method_id": "PayPal",
  .... "c_PP_API_ActiveBillingAgreement": "{ \"baID\": \"B-68Y08988NL6028922\", \"email\": \"ptest4@paypal.com\", \"default\": true }",
  .... "c_paymentId": "PayPal"
}
```

Figure 45. The saved PayPal billing agreement body example (PayPal payment method used)

In case of success your basket payment instrument will created with new billing agreement:

```
"payment_instruments": [
  {
    "_type": "order_payment_instrument",
    "amount": 111.28,
    "payment_instrument_id": "c15f275708ba13477ad402f9f7",
    "payment_method_id": "PayPal",
    "c_PP_API_ActiveBillingAgreement": "{ \"baID\": \"B-68Y08988NL6028922\", \"email\": \"ptest4@paypal.com\" }",
    "c_currentPaypalEmail": "ptest4@paypal.com",
    "c_paymentId": "PayPal"
  }
],
```

Figure 46. A basket payment instrument with the saved active billing agreement

- Flow with new PayPal billing agreement body creation (billing agreement is enabled in site preferences):

To add a new PayPal billing agreement to the basket, you need to transfer the following properties:

- amount - the order total amount. This is required parameter. Type - Number.
 - **payment_method_id** - the payment method ID. Use 'PayPal' for adding PayPal payment method This is required parameter. Type - String.
 - **c_paypalToken** - billing agreement token. Use for a billing agreement creating. This is required parameter. Type - String.
- Look at 9.7 'Get basket' of how to get c_paypalToken.**
- **c_paymentId** - you can configurate PayPal JavaScript SDK to see additional payment methods on your site by using the site preferences from 'Paypal_Checkout' site preferences group. The following paymentId' are supported: 'Venmo', 'PayPal Debit/Credit Card', and all local payment method provided into the site preference - PP_API_APM_methods (Available Alternative Payment Methods). To enable 'Venmo' payment method use the following site preference: PP_API_Venmo_Enabled. In case of using one of that payment methods, please provide it into the c_paymentId custom attribute.

This is required parameter. Type - String.

```
{
  "amount": "{{order_total}}",
  "payment_method_id": "PayPal",
  "c_paypalToken": "BA-0N637818NF139164N",
  "c_paymentId": "PayPal"
}
```

Figure 47. The new billing agreement body example

In case of success your basket payment instrument will be created with new billing agreement:

```
"payment_instruments": [
  {
    "_type": "order_payment_instrument",
    "amount": 111.28,
    "payment_instrument_id": "c15f275708ba13477ad402f9f7",
    "payment_method_id": "PayPal",
    "c_PP_API_ActiveBillingAgreement": "{\"baID\":\"B-68Y08988NL6028922\",\"email\":\"ptest4@paypal.com\"}",
    "c_currentPaypalEmail": "ptest4@paypal.com",
    "c_paymentId": "PayPal"
  }
],
```

Figure 48. A basket payment instrument with a new active billing agreement

- Order id flow (billing agreement is disabled in site preferences):

To use 'Order id' flow, you need to transfer the following properties to the basket:

- amount - the order total amount. This is required parameter. Type - Number.
- **payment_method_id** - the payment method ID. Use 'PayPal' for adding PayPal payment method. This is required parameter. Type - String.
- **c_paypalOrderID** - paypal order id. Use when billing agreement flow is disabled. This is required parameter. Type - String.
Look at 9.7 'Get basket' of how to get c_paypalOrderID.
- **c_paymentId** - payment id. Generally we have one payment method - PayPal. But PayPal payment method supports a few additional payment methods as additional functionality. In order to indicate which payment method was used we save c_paymentId into payment instrument custom attribute. PayPal also supports the following paymentId: 'Venmo', 'PayPal Debit/Credit Card', and all local payment method provided into the site preference - PP_API_APM_methods (Available Alternative Payment Methods). This is required parameter. Type - String.

```

{
  "amount": "{{order_total}}",
  "payment_method_id": "PayPal",
  "c_paypalOrderID": "0N637818NF139164N",
  "c_paymentId": "PayPal"
}

```

Figure 49. The Order id flow body example

In case of success your basket payment instrument will be created with new order id:

```

"payment_instruments": [
  {
    "_type": "order_payment_instrument",
    "amount": 111.28,
    "payment_instrument_id": "30fb6bfc283d1baad633531b8d",
    "payment_method_id": "PayPal",
    "c_currentPaypalEmail": "ptest4@paypal.com",
    "c_paymentId": "PayPal",
    "c_paypalOrderID": "7K246814CJ5758911"
  }
],

```

Figure 50. A basket payment instrument with PayPal order id

9.7 Get basket (get banner config for cart / minicart / billing pages)

An Additional request to get some helpful data to place an order due PayPal. To get a basket you need to provide a next request:

- **Method:** GET
- **URL:** `{{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/basket/{basket_id}?pageId={pageid}` (The basic variables used in URL are described in paragraph 8.1.1 Basic variable's description)
- **Headers:**
 1. *Content-Type* (value - *application/json*)
 2. *Authorization* (value - `{{bearer_token}}`, how to get `{{bearer_token}}` look in 8.1.2 Registered and an unregistered user authentication)

3. Content-length: 0

- Body (example): there is no body for current request.

9.7.1 Url's query parameters

Depend on the page you need to provide the 'pagild' query parameters to the 'get Basket' url:

pagild - the current page id. Required for all pages. Possible pages id: pdp(product), cart, minicart, billing; For example: If you process the 'Mini cart' page, pagild=minicart must be provided.

9.7.2 PayPal buttons config objects creation

To have a possibility to render the PayPal payment methods on the 'Billing' or 'Cart', 'PDP' (pdpButtonConfigs can also be obtained by using **8.1.2 Registered and an unregistered user authentication request**) or 'Mini Cart' pages, a 'paypalButtonConfigSdkUrl' and the additional button configs will include into 'c_paypalButtonConfigs' object in the 'get Basket' response. If buyer has a saved billing agreement, the config object will contain the static image link, that you can use for the flow with the saved billing agreement. The response depends on 'pagild' query parameter you provided.

Example of the smart as well as the static button you can find here: **6.2 Storefront Functionality**

9.7.3 Get banner config for cart / minicart / billing pages

In a case of success, when query parameter "pagild" is 'cart' / 'minicart' / 'billing' was passed, you will get response which contain c_paypalCartBannersConfig:

```
"c_paypalCartBannersConfig": {  
  "paypal": {  
    "bannerSdkUrl": "https://www.paypal.com/sdk/js?  
client-id=AWLvojdRg8Ga60b5lmkiSyukGGmE0i0h413Mo5RQ4xvcwFgW8zKeiwITB2BomdVwkm7wYhSKNwV_Wiwi&components=messages",  
    "bannerConfig": {  
      "styleColor": "black",  
      "styleRatio": "8x1",  
      "styleLayout": "text",  
      "styleTextColor": "black",  
      "styleLogoPosition": "left",  
      "styleLogoType": "alternative"  
    },  
    "paypalAmount": 325.48  
  },  
  "creditMessageAvailable": true  
}
```

9.7.4 A billing agreement token(c_paypalToken) and a purchase unit creation

To get a billing agreement token or purchase unit parameter, this call must be occurred before a basket payment instrument will be create.

In depends on flow you use (Billing agreement - billing agreement is enabled in site preferences or order id - billing agreement is disabled in site preferences) the 'get Basket' response will contain whether a billing agreement token or purchase unit. Use a billing agreement token for creating a basket payment instrument with new billing agreement flow.

Use a purchase unit for getting order id (will use for creating a basket payment instrument with order id flow) in PayPal 'createOrder'(as example) event during Paypal button configuration.

More information about PayPal button configuration here: [JavaScript SDK reference](#)

```
        "tax_basis": 5.99,
        "tax_class_id": "standard",
        "tax_rate": 0.05
    },
    ],
    "shipping_total": 5.99,
    "shipping_total_tax": 0.30,
    "taxation": "net",
    "tax_total": 5.30,
    "c_paypalButtonConfigs": {
        "partnerAttributionId": "SFCC_EC_B2C_2022_1_0",
        "paypalButtonSdkUrl": "https://www.paypal.com/sdk",
        "paypalEmail": null,
        "showStaticImage": true,
        "defaultBAemail": "ptest4@paypal.com",
        "paypalStaticImageLink": "https://www.paypalobject",
        "billingAgreementEnabled": true,
        "buttonConfig": {
            "style": {
                "color": "black",
                "shape": "rect",
                "label": "checkout",
                "size": "medium",
                "layout": "horizontal",
                "tagline": true,
                "height": 40
            }
        }
    },
    "c_billingAgreementToken": "BA-86N662777V9111637"
```

Figure 51. A basket with response with billing agreement token example

9.7.5 A billing agreement and order details object creating

To get a billing agreement or order details object, this call must be occurred after a basket payment instrument will be create.

It depends on the flow you use (Billing agreement - billing agreement is enabled in site preferences or order id - billing agreement is disabled in site preferences) the 'get Basket' response will contain whether a billing agreement or order details object. You can use the data from this object, for example: for updating basket billing or shipping address with address from PayPal.

```
{
  "c_billingAgreementDetails": {
    "id": "B-68Y08988NL6028922",
    "billing_info": {
      "email": "ptest4@paypal.com",
      "first_name": "Monalisa",
      "last_name": "Patel",
      "payer_id": "B58N7CWR656DQ",
      "phone": "408-972-7588",
      "billing_address": {
        "line1": "1 Main St",
        "city": "San Jose",
        "state": "CA",
        "country_code": "US",
        "postal_code": "95131"
      },
      "tenant": "PAYPAL"
    },
    "shipping_address": {
      "recipient_name": "Monalisa Patel",
      "line1": "1 Main St",
      "city": "San Jose",
      "state": "CA",
      "country_code": "US",
      "postal_code": "95131"
    },
    "active": true
  }
}
```

Figure 52. A billing agreement details object example

9.8 Place order

To place an order to the basket you need to provide a next request:

- **Method:** POST
- **URL:** `{{host}}/s/Sites-{{site}}-Site/dw/shop/{{ocapi_version}}/orders`

(The basic variables used in URL are described in paragraph **8.1.1 Basic variable's description**

- **Headers:**
 1. *Content-Type* (value - *application/json*)
 2. *Authorization* (value - `{{bearer_token}}`, how to get `{{bearer_token}}` look in **8.1.2 Registered and an unregistered user authentication**)
 3. *Content-length*: 0
- **Body (example):**

```
... "basket_id" : " {{basket_id}} "
```

Figure 52. 'Place order' body example

`basket_id` - current basket id.

In case of success:

```
{
  "_v": "22.4",
  "_type": "order",
  "_resource_state": "60ccac8822215f4607b50bf6cb92e7b095e5570d26ceb249148af789b9817cbd",
  "adjusted_merchandize_total_tax": 5.00,
  "adjusted_shipping_total_tax": 0.30,
  "billing_address": {
    "_type": "order_address",
    "address1": "1 Main St",
    "city": "Boston",
    "country_code": "US",
    "first_name": "Jon",
    "full_name": "Jon Don",
    "id": "81d5be51518d5a61e5d229bb27",
    "last_name": "Don",
    "phone": "4088161358",
    "postal_code": "95131",
    "state_code": "CA"
  },
  "channel_type": "storefront",
  "confirmation_status": "confirmed",
  "created_by": "Customer",
  "creation_date": "2022-08-20T09:50:26.686Z",
  "currency": "USD",
  "customer_info": {
    "_type": "customer_info",
    "customer_id": "adR0XJDMjU7aLSIzdx90Yb41w",
    "customer_name": "Ocapi Postman",
    "customer_no": "00036502",
    "email": "valid@email.com"
  }
}
```



```

},
"customer_name": "Ocapì Postman",
"export_status": "ready",
"guest": false,
"last_modified": "2022-08-20T09:50:29.393Z",
"merchandise_total_tax": 5.00,
"notes": {
  "_type": "simple_link",
  "link": "https://zzod-008.sandbox.us03.dx.commercecloud.salesforce.com/s/RefArch/dw/shop/v22_4/orders/00028002/notes"
},
"order_no": "00028002",
"order_token": "hh7y3o3JPACuoxYVL0intCl2gNm5_eAw8ci6ktQ6FD4",
"order_total": 111.28,
"payment_instruments": [
  {
    "_type": "order_payment_instrument",
    "amount": 111.28,
    "payment_instrument_id": "ebbf1415bebaab9bf0d6048708",
    "payment_method_id": "PayPal",
    "c_PP_API_ActiveBillingAgreement": "{\"baID\":\"B-68Y08988NL6028922\", \"email\":\"ptest4@paypal.com\", \"default\":\"tr",
    "c_paymentId": "PayPal",
    "c_paypalOrderID": "80X61337G0329781X",
    "c_paypalPaymentStatus": "COMPLETED"
  }
],
"payment_status": "not_paid",
"product_items": [
  {
    "_type": "product_item",
    "adjusted_tax": 5.00,
    "base_price": 99.99,

```

```

    "base_price": 99.99,
    "bonus_product_line_item": false,
    "gift": false,
    "item_id": "b64433cae3e37fb2448d1da9f3",
    "item_text": "Platinum Blue Stripes Easy Care Fitted Shirt ",
    "price": 99.99,
    "price_after_item_discount": 99.99,
    "price_after_order_discount": 99.99,
    "product_id": "008884304009M",
    "product_name": "Platinum Blue Stripes Easy Care Fitted Shirt ",
    "quantity": 1,
    "shipment_id": "me",
    "tax": 5.00,
    "tax_basis": 99.99,
    "tax_class_id": "standard",
    "tax_rate": 0.05
  }
],
"product_sub_total": 99.99,
"product_total": 99.99,
"shipments": [
  {
    "_type": "shipment",
    "adjusted_merchandise_total_tax": 5.00,
    "adjusted_shipping_total_tax": 0.30,
    "gift": false,
    "merchandise_total_tax": 5.00,
    "product_sub_total": 99.99,
    "product_total": 99.99,
    "shipment_id": "me",
    "shipment_no": "00137501",

```

```

    "shipment_total": 111.28,
    "shipping_address": {
      "_type": "order_address",
      "address1": "1 Main St",
      "city": "Boston",
      "country_code": "US",
      "first_name": "John",
      "full_name": "John Smith",
      "id": "6f62bd2d26601a47712d2d3761",
      "last_name": "Smith",
      "phone": "408-972-7588",
      "postal_code": "95131",
      "state_code": "CA"
    },
    "shipping_method": {
      "_type": "shipping_method",
      "description": "Order received within 7-10 business days",
      "id": "001",
      "name": "Ground",
      "price": 5.99,
      "c_estimatedArrivalTime": "7-10 Business Days"
    },
    "shipping_status": "not_shipped",
    "shipping_total": 5.99,
    "shipping_total_tax": 0.30,
    "tax_total": 5.30
  }
],
"shipping_items": [
  {
    "_type": "shipping_item",

```

```

"shipping_items": [
  {
    "_type": "shipping_item",
    "adjusted_tax": 0.30,
    "base_price": 5.99,
    "item_id": "4960b0625da5b7301679eb3e77",
    "item_text": "Shipping",
    "price": 5.99,
    "price_after_item_discount": 5.99,
    "shipment_id": "me",
    "tax": 0.30,
    "tax_basis": 5.99,
    "tax_class_id": "standard",
    "tax_rate": 0.05
  }
],
"shipping_status": "not_shipped",
"shipping_total": 5.99,
"shipping_total_tax": 0.30,
"site_id": "RefArch",
"status": "new",
"taxation": "net",
"tax_total": 5.30,
"c_PP_API_TransactionID": "5P338097XT592843G",
"c_paypalPaymentMethod": "express"

```

Figure 53. Successfully placed order due PayPal payment method

9.9 Additional Ocapi resources

Follow Okapi documentation by the link: [OCAPI documentation](#)

By follow the documentation above you can use any other Ocapi Hook from the list of Shop API resources index. It gives opportunity to do more actions with Ocapi resources.

How to use:

1. Click link above.
2. In the opened page click Shop API resource index.
3. Choose the needed index (in this example we choose 'Basket' index).
4. Choose the needed Basket resource from list (in this example '/basket/{basketid}' resource with 'DELETE' http method was chosen). And click on the resource.
5. In the opened page, will be all necessary information on how to create request on this resource.
6. Go to: Business manager > Administration > Open Commerce API Settings. Set 'Select Type' - Shop, Select Context - your site id.
7. Paste your resource into resource list.

For more information of how to make Ocapi settings follow the link:

[OCAPI setting](#)

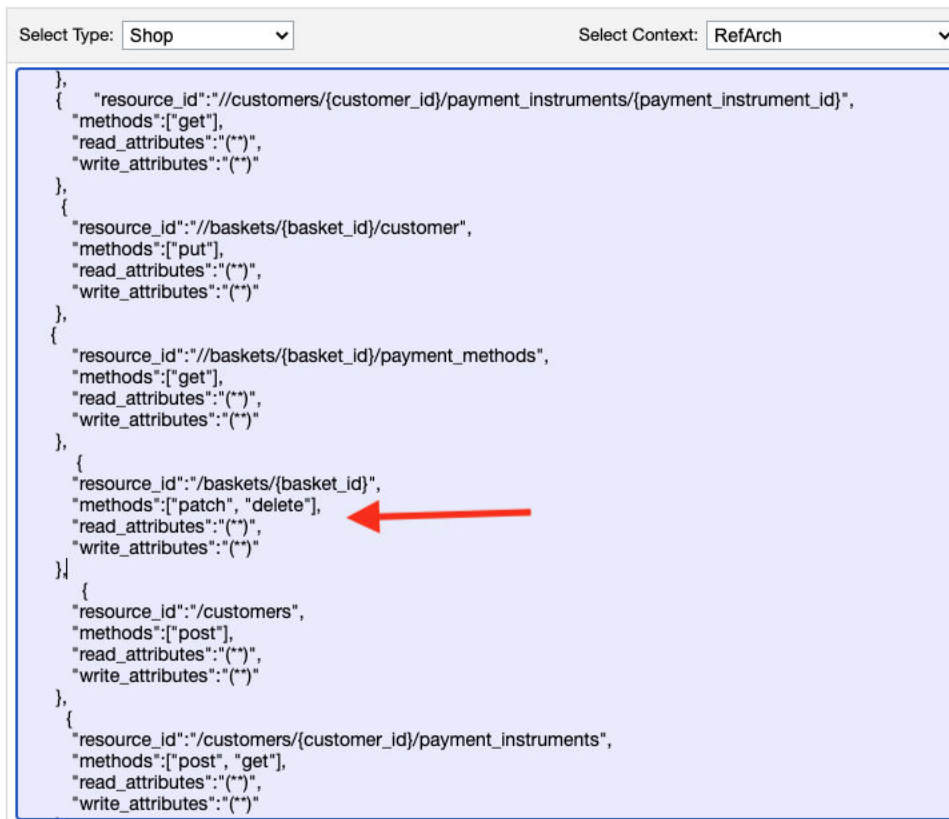


Figure 54. Adding a new resource to the Ocapi settings