

### Integrating PayPal PLUS

A Quick Integration Guide for PayPal PLUS

01. Introduction 4

01.1. Change History 4

02. Using the PayPal API 5

02.1. Architecture 5

02.2. PayPal Sandbox 5

02.3. API Endpoints 5

02.4. Creating client\_ID and secret 6

02.5. Authentication & Headers 7

03. Checkout Experience 7

04. Payments 8

04.1. Create a payment 8

04.2. Update a payment resource (Patch payment) - Optional 9

04.3. Look up a payment resource – Optional 9

04.4. Execute an approved PayPal payment 9

05. Refunds 11

05.1. Look up a sale 11

05.2. Refund a sale 11

06. Payment Notifications 11

07. Switching from Sandbox to Live environment 12

08. Integrating PayPal PLUS 13

08.1. Integrating the payment wall: 16

08.2. pppluslibrary 17

08.3. API 17

08.4. Configuration object 17

08.5. Integrating the external “continue button” 22

08.6. Update (Patch) payment 23

08.7. Redirect to authorize payment 24

08.8. Look Up Payment 25

08.9. Update (Patch) payment 25

08.10. Execute Payment 26

09. Integrating manual Bank Transfer (mEFT) 26

10. Additional Steps to Integrate Pay upon Invoice 27

10.1. Before you start: 27

10.2. Enabling Pay upon Invoice in Sandbox 27

10.3. Step 1: Retrieving Payment Instructions: 27

10.4. Presentation on the Success Page 29

10.5. Step 2: Managing invoice transactions 29

11. Integration using requireJS 31

12. PayPal PLUS Walk Through 33

12.1. Checkout with ELV 33

12.2. Checkout with Credit Card 35

12.3. Checkout with Pay upon Invoice 36

13. Appendix 38

13.1. Technical Documentation 38

13.2. API Documentation 38

13.3. Reporting 38

# Introduction

PayPal PLUS is a solution where PayPal offers PayPal, Credit Card, Direct Debit (ELV) and pay upon invoice as individual payment options on the payment selection page. The available payment methods are provided in a PayPal hosted iframe.

Based on the payment method selected by the buyer, he will at a later stage (on redirect to PayPal / in context checkout) be presented with either the PayPal Login page or an input mask for bank / credit card details or a pay upon invoice details page.

The redirect can be initiated from within the iframe or at any later stage during checkout.

The final payment execution happens on the merchant side.

**Please note:**

PayPal PLUS requires a vetting process. To apply for PayPal PLUS go to this link:

[www.paypal.de/paypal-plus](http://www.paypal.de/paypal-plus)

PayPal PLUS requires a fully verified PayPal merchant account

Pay upon invoice may not be available for all merchants.

PayPal PLUS is by default enabled for all German Sandbox accounts.

**Please note:** Nothing in this Guide shall be considered to constitute legal advice. Please contact your legal counsel in case of questions.

## Change History

This version replaces all previous PayPal PLUS integration guides (Vers. 1.x) rendering them obsolete!

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Author** | **Description** |
| 10/12/2015 | 2 | Kristian Büsch | Initial draft based on October 2015 integration standards |
| 10/30/2015 | 2.1 | Kristian Büsch | Added Customization of the payment selection page PSP Invoice legal note |
| 01/14/2016 | 2.2 | Kristian Büsch | Loading Indicator, requireJS |
| 01/14/2016 | 2.3 | Kristian Büsch | Minor updates |
| 01/14/2016 | 2.4 | Kristian Büsch | Update adding 3rd Party Methods |
| 05/23/2016 | 2.5 | Kristian Büsch | Best Practise Update |
| 07/01/2016 | 2.6 | Kristian Büsch | Billing Address object removed |
| 07/07/2016 | 2.7 | Kristian Büsch | Added IPN support |
| 07/12/2016 | 2.8. | Kristian Büsch | Update Sandbox Links |
| 07/19/2016 | 2.9. | Kristian Büsch | Update Response Object |

# Using the PayPal API

## Architecture

PayPal PLUS utilizes PayPal’s RESTful API.

<https://developer.paypal.com/webapps/developer/docs/api/>

## PayPal Sandbox

The PayPal test environment “Sandbox” is a mirror of the PayPal live system. Any part of the PayPal integration can be tested in this environment without “real” funds being moved.

PayPal recommends using the Sandbox for integration and switching to the live system once the integration has been finished. Therefore you can create preconfigured buyer and seller account within the developer platform of PayPal at

<https://developer.paypal.com/docs/classic/lifecycle/sb_create-accounts/>

## API Endpoints

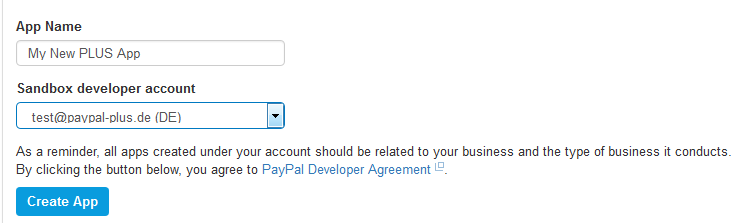
The PayPal REST APIs are supported in the Sandbox environment for testing purposes. Use your Test credentials to generate an access token for making calls with the Sandbox endpoints. When you’re set to go live, get your credentials for the Live environment and generate a new access token that you can use with the Live endpoints.

The following URIs form the base for the endpoints in the supported environments:

* Sandbox, base URI: https://api.sandbox.paypal.com
* Live, base URI : https://api.paypal.com

## Creating client\_ID and secret

PayPal automatically provides Sandbox OAuth keys (a client\_id and secret) for each application you create. You can then go to “[Manage your applications](https://developer.paypal.com/webapps/developer/docs/integration/admin/manage-apps/)” for details on creating and managing your PayPal applications. When selecting the facilitator please make sure to pick a DE account.



If you create Sandbox Test accounts prior to creating the APP they will be listed in the dropdown menu there

Direct Link: <https://developer.paypal.com/developer/applications/>

To create a new PayPal app log in to the [Developer portal](https://developer.paypal.com/) with your PayPal account and navigate to the [My REST apps](https://developer.paypal.com/webapps/developer/applications/myapps) page. Click Create App to begin the application-creation process.

Direct link: <https://developer.paypal.com/developer/applications/create>

## Authentication & Headers

With each API call a request headers needs to be set including an OAuth 2.0 access token. An access token can be requested by using the OAuth 2.0 client\_credentials token grant type with your clientId:secret as your Basic Auth credentials.

Access tokens should not be created on a transactional basis. When an access token is generated PayPal returns the lifespan as "expires\_in": 32187 in seconds. There is no refresh token so the remaining time should be tracked and/or a new access token generated once PayPal returns error response 401 unauthorized when a token is expired.

For more information, see [Make your first call](https://developer.paypal.com/webapps/developer/docs/integration/direct/make-your-first-call/). You can use your Sandbox access token to try any of the code in this reference.

# Checkout Experience

Endpoint POST /v1/payment-experience/webprofiles

The Checkout experience allows a merchant to create a profile where default parameters can be set and it allows designing the overall experience for a consumer during checkout.

These global parameters are to be set only once and the referenced in create payment via the experience\_profile\_id that was returned upon submitting a successful request. Localized or sub shops should be referenced with individual profiles.

By default address override is set when a shipping address is submitted. With PayPal PLUS you shall not pass name and shipping address in the createPayment but in the update (PATCH) payment. You therefore have to specify address\_override in the experience profile.

Experience profiles must not be generated with each transaction. They should be set once and then referenced.

**Recommended parameters:**

* brand\_name => My Shop Name
* logo\_image => https://www.myshop.de/logo.gif
* locale\_code => DE
* allow\_note => false
* no\_shipping => 0
* address\_override => 1

**Please note:** Please do not specify a dedicated landing\_page\_type for PayPal PLUS!

[Developer Documentation](https://developer.paypal.com/webapps/developer/docs/integration/direct/rest-experience-overview/)

# Payments

URI: <https://api.paypal.com/v1/payments/payment>

PayPal provides various payment related operations using the /payment resource and related sub-resources. Use payment for direct credit card payments and PayPal account payments. You can also use sub-resources to get payment related details.

## Create a payment

Endpoint POST /v1/payments/payment

Depending on the payment\_method and the funding\_instrument, you can use a number of payment resources. For PayPal PLUS the only valid payment resource is [PayPal account payments](https://developer.paypal.com/webapps/developer/docs/integration/web/accept-paypal-payment/).

In the response PayPal returns a payment object along with the state of the payment and sale. The response includes a payment ID that uniquely identifies the transaction. Also included is an approval\_url that is required to render the PayPal PLUS payment wall.

Since Create Payment is a call prior to the actual payment selection you shall not transfer any user specific data to PayPal. These data shall be transferred only after the selection of a PayPal PLUS related payment method by the customer. To update the payment resource please use PATCH.

**Recommended parameters:**

* For PayPal PLUS line items must be submitted
* Name, shipping / billing and any other customer related data shall not be submitted here but later by using PATCH Payment.

**Please note:**

Including / excluding VAT

To avoid rounding errors we recommend not submitting tax amounts on line item basis. Calculated tax amounts for the entire shopping basket may be submitted in the amount objects. In this case the item amounts will be treated as amounts excluding tax. In a B2C scenario, where taxes are included, no taxes should be submitted to PayPal.

[Developer documentation](https://developer.paypal.com/webapps/developer/docs/api/#create-a-payment)

## Update a payment resource (Patch payment)

Endpoint PATCH /v1/payments/payment/<Payment-Id>

Use this call to partially update the Payment resource for the given identifier. Only the transaction amount object, invoice\_number and shipping address can be updated.

**Please note:**

* Lines items cannot be update at any stage.
* Shipping costs, discounts and other fees can be updated as part of the amount object

**Please note:** Only add and replace are valid operations. All other operations will fail.   
 **Please note:** Patch can only be utilized prior to Execute Payment. Once Execute Payment has been successfully completed, PATCH will no longer work.

**Please note:** Costumer specific data shall not be submitted to PayPal prior to payment method selection!

**Recommended parameters:**

* Amount (/transactions/0/amount) including all sub parameters including subtotal. Please note: The entire amount object is replaced when updated
* Invoice / Order number (/transactions/0/invoice\_number). **Must be unique**
* Shipping Address (/transactions/0/item\_list/shipping\_address)

[Developer documentation](https://developer.paypal.com/webapps/developer/docs/api/#update-a-payment-resource)

## Look up a payment resource – Optional

Endpoint GET /v1/payments/payment/{paymentId}

Use this call to get details about payments that have not completed, such as payments that are created and approved, or if a payment has failed.

[Developer documentation](https://developer.paypal.com/webapps/developer/docs/api/#look-up-a-payment-resource)

## Execute an approved PayPal payment

Endpoint POST /v1/payments/payment/{paymentId}/execute

Use this call to execute (complete) a PayPal payment that has been approved by the payer. Execute should only submit the Payer ID. All transaction related updates (shipping address, charges that are part of the amount object, invoice\_number should be updated by using PATCH payment.

**Important:** This call only works after a buyer has approved the payment using the provided PayPal approval URL.

[Developer documentation](https://developer.paypal.com/webapps/developer/docs/api/#execute-an-approved-paypal-payment)

# Refunds

Before initiating a refund it is recommended getting the details about completed payments (sale transaction) created by a payment request or to refund a direct sale transaction, PayPal provides the /sale resource and related sub-resources. You can find the sale transactions in the payment resource within related\_resources.

URI <https://api.paypal.com/v1/payments/sale>

## Look up a sale

Endpoint GET /v1/payments/sale/{transactionId}

Use this call to get details about a sale transaction.   
Note: This call returns only the sales that were created via the REST API.

[Developer documentation](https://developer.paypal.com/webapps/developer/docs/api/#look-up-a-sale)

## Refund a sale

Endpoint POST /v1/payments/sale/{id}/refund

Use this call to refund a completed payment. Provide the sale\_id in the URI and an empty JSON payload for a full refund. For partial refunds, you can include an amount.

[Developer documentation](https://developer.paypal.com/webapps/developer/docs/api/#refund-a-sale)

# Payment Notifications

Payment notifications in REST are managed via webhooks.

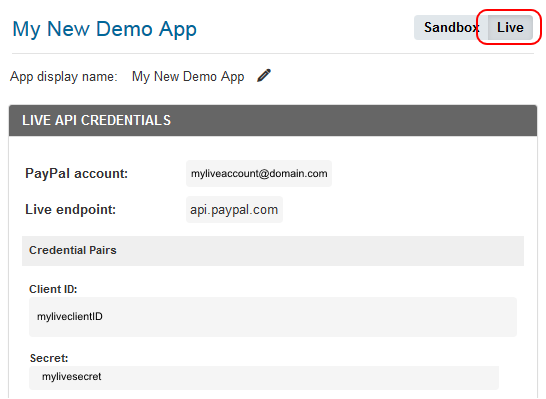
**Please note:** Webhooks require https!

[Developer documentation](https://developer.paypal.com/webapps/developer/docs/api/#notifications)

You may also use Instant Payment Notification ([IPN](https://developer.paypal.com/docs/classic/ipn/integration-guide/IPNIntro/?mark=IPN)). To address the listener script, add a notify\_url parameter to the transaction object in Create Payment.

# Switching from Sandbox to Live environment

When testing is complete and you are ready for switch to live there are three touch points.

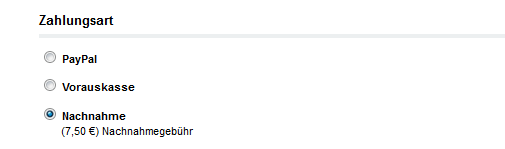
1. Change the API endpoint from <https://api.sandbox.paypal.com> to <https://api.paypal.com>
2. Change the Environment in the configuration element of the PLUS library by commenting out "mode": "sandbox", or replace sandbox with live.
3. Update the Sandbox REST credentials (Client / secret) with the live credentials from your REST APP. You can either generate a new REST APP or simply switch to Live within the credentials view of your APP.  
     
   

1. Make sure your live account is enabled for PayPal PLUS.

# Integrating PayPal PLUS

With PayPal PLUS your payment selection page is replaced by the PayPal hosted payment wall. Up to 5 non PayPal payment methods can be hosted within this payment wall.

**Without PayPal PLUS:**



**With PayPal PLUS:**



PayPal PLUS requires a PayPal Integration based on [PayPal’s REST payment API](https://developer.paypal.com/webapps/developer/docs/api/).

Prior to rendering the payment wall a payment resource must be created. The corresponding API call is “create payment”. When creating the payment resource amount, currency and items details must be submitted.

The create payment call shall not contain any customer specific data.

Once a buyer has selected one of the PayPal methods the payment resource is updated via Patch and the shipping address can be added. (Patch Payment)

In the next step the actual PayPal payment wall is rendered. Each instance is rendered on demand. Unique identifier is the approvalURL as returned in the Create payment call.

Depending on the configuration of this payment wall different scenarios may occur.

As default a payment button will be displayed within the payment wall and displayed when a user selects a payment method. It is also possible to use an existing button outside of the payment wall. To implement this use case, please refer to the chapter.

Integrating the external “continue button”

When a user is clicking the “checkout now” a “DoCheckout” function is invoked and the user is redirected to a PayPal hosted pages. Based on the selected method PayPal will direct the user to either the PayPal login - , credit card - or bank details page. All fields where information previously submitted will appear prepopulated.

It is possible to implement an extended flow where the “DoCheckout function” is executed on a subsequent page. For details on how to integrate please refer to the chapter

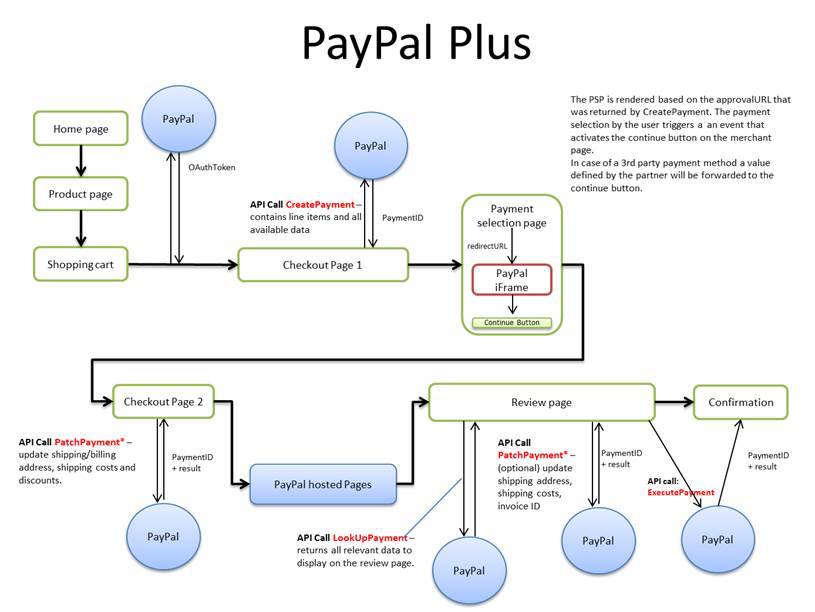
Integrating the external “continue button”

When hosting 3rd party methods in the paymentwall the “DoCheckout function” will redirect the user to the redirectURL as defined in the config element of the payment wall.

Once the user completed the required steps he will be redirect to the returnURL. From there the transaction should be handled like any standard PayPal transaction.

**Please note:**

PayPal PLUS is designed to support transaction type “Sale” (instant settlement).



PayPal PLUS Flow

## Integrating the payment wall:

The payment wall is the actual payment selection frame. It is rendered on demand and based on the approvalURL as received in the links object of the *Create Payment* response.

[links] => Array

(

...

[1] => Array

(

[href] => https://www.sandbox.paypal.com/cgi-bin/webscr?cmd=\_express-checkout&token=EC-31737613MN931841V

[rel] => approval\_url

[method] => REDIRECT

)

...

Depending on the Billing address country a buyer will see all available funding sources for his country of origin.

1) Include JS library:

<script src="<https://www.paypalobjects.com/webstatic/ppplus/ppplus.min.js>" type="text/javascript"></script>

2) Define a placeholder DIV:

<div id="ppplus"> </div>

3) Render Payment wall:

<script type="application/javascript">

            var ppp = PAYPAL.apps.PPP({

            "approvalUrl": "'.$approval\_url.'",

            "placeholder": "ppplus",

            "mode": "sandbox",

"country": "DE"

            });

</script>

4) Configure the “Continue Button” ([see Integrating the external “continue button](#_Integrating_the_external)).

## pppluslibrary

Provides a JavaScript library for including the PayPal PLUS Payment Selection Page (PSP) in a merchants website. When loaded, the library will attach itself as an object to PAYPAL.apps as PAYPAL.apps.PPP. The PPP object defines the following API:

## API

var ppp = PAYPAL.apps.PPP(config)

Creates an instance of the PSP. It requires as only parameter a configuration object (described below). It returns the instance object.

ppp.doContinue()

This function should be invoked by the outside continue button when it is being clicked.

PAYPAL.apps.PPP.doCheckout()

This function should be invoked by the elsewhere continue button or when the checkout event is handled by the library.

## Configuration object

The configuration object takes the following parameters:

**Required parameters**

approvalUrl : string

*(required)* The approvalUrl that is returned by the `create payment` call.

placeholder : string

*(required)* The id of a DOM object where the PSP will be mounted in the page. The PSP will adapt the width of that object.

country : ISO 3166-1 alpha-2, (e.g. FR, GB, DE, default US)

Country in which the PayPal PLUS service will be deployed.

**Please note:** The country code must be taken from the billing address a buyer entered during checkout.

**Optional parameters**

mode : "live" or "sandbox" (defaults to "live")

Depending on the mode, the library will load the PSP from different locations. live will load it from paypal.com whereas sandbox will load it from sandbox.paypal.com. The library will also emit warning to the console if the mode is sandbox (in live mode it will do so only for required parameters).

buttonLocation : "inside" or "outside" (defaults to "inside")

Determines the location of the Continue button. Don't forget to set the *onContinue* parameter.

preselection : "none", "paypal", "Vorkasse" synonyms (defaults to "paypal")

* Determines if one of the following should be preselected:
* nothing (="none"),
* PayPal Wallet (="paypal") or
* thirdparty method with methodName "Vorkasse" (or one of its synonyms).
* "Vorkasse" synonyms: "Vorkasse", "Überweisung", "Vorauszahlung", "Vorausbezahlung", "Vorauskasse", "Vorabzahlung", "prepayment", "advance payment", "payment in advance" and "pay in advance“

language : Lanuage: [ll]\_[CC], e.g "en\_US", "de\_DE" (defaults to "en\_US")

[ll] = language (ISO\_639-1)  
[CC] = country (ISO\_3166-1\_alpha-2)   
[all supported locale codes](https://developer.paypal.com/docs/classic/api/locale_codes/)

Language in which to render PayPal PLUS PSP. This should be the language of the merchant shop to keep the content for the buyer consistent.   
  
**Please note:** The language code should be dynamically filled according to the language displayed in the shop frontend. Currently unsupported languages will default to English but may be added at a later stage.

onContinue : callback or string

If a string is given that string will be interpreted as ID of a form in the document which will be submitted when the inside continue button is clicked. If a callback is given that callback will be invoked when the inside continue button is clicked.

onLoad : callback

A function that should be called when the iframe loaded successfully.

enableContinue : callback or string

If a string is given that string will be interpreted as ID of the outside continue button (may be any element) which will be enabled when a payment selection is made. In case of a callback that callback will be invoked then.

disableContinue : callback or string

If a string is given that string will be interpreted as ID of the outside continue button (may be any element) which will be disabled if no payment method is selected. In case of a callback that callback will be invoked then.

deselectPaymentMethod

Emitted when the action for deselecting all payment methods is being called.

useraction: "commit"

Checkout flow to be implemented by the Merchant. If not set, the default will be set to the "Continue" flow. The checkout flow selected determines whether the merchant explicitly requires that the buyer reviews and confirms the payment on a review page ("Continue" Flow) or if he/she can confirm payment on PayPal ("Commit" Flow).

surcharging: "true" (defaults to false)

If set to "true" it will activate a message that indicates that surcharges will be applied.

showLoadingIndicator: true or false (defaults to false)

If set to "true" it will show a loading spinner until the PSP is completely rendered.

showPuiOnSandbox: true or false (defaults to false)

If set to "true" PUI is shown in sandbox mode (NOTE: this parameter is ignored in production mode!)

onThirdPartyPaymentMethodSelected : callback

The callback will be invoked when a thirdPartyMethod is selected. Along with it you receive the method name.

onThirdPartyPaymentMethodDeselected : callback

The callback will be invoked when a thirdPartyMethod loses selection state through the selection of another paymentMethod

Example:

onThirdPartyPaymentMethodSelected: function (data) {

/\* for example:

switch(data.thirdPartyPaymentMethod) {

case x: ...

case y: ...

} \*/

var output = "onThirdPartyPaymentMethodSelected callback, returned data: " + JSON.stringify(data);

console.log(output);

},

onThirdPartyPaymentMethodDeselected: function (data) {

var output = "onThirdPartyPaymentMethodDeselected callback, returned data: " + JSON.stringify(data);

console.log(output);

},

**Integrating third party payment methods in to the payment selection page**

thirdPartyPaymentMethods : JSON Array

The user may configure up to 10 additional payment methods within the PSP. If more payment methods are defined, only the first 5 will be taken into account. If third party payment methods are specified, the user will be not be redirected to the PayPal checkout but to the given redirect URL directly (doCheckout function). The following is the structure of the JSON Object:

thirdPartyPaymentMethods:

[{"redirectUrl":"<http://myshop.com/vorkasse.php>",   
"methodName": "Bank transfer",   
"imageUrl": "<https://myshop.com/logo.png>",   
"description": "my description"}]

The following validations will be performed on the JSON Array:

* If any of the Payment methods defined has errors, all the methods will be discarded.
* Parameters redirectURL and methodName are mandatory.
* The imageUrl must be a secure (https) URL.
* The image size must be maximum 100 x 25 px.
* methodName can contain maximum 25 Characters, additional characters will be truncated.
* methodName of a thirdParty method should be unique inside the third party JSON array to ensure proper functionality of thirdparty payment methods.

There is a magic 3rdParty which you can use for displaying PayPal Installments in the list. Just set methodName: "PP\_Installments" and specify the redirect url. The logo and the description will be ignored and set by PayPal.

**Customization**

The following are the customization whitelists currently supported by the PSP:

Only whitelisted attributes and selectors may be defined. Additional attributes and selectors will be discarded. Take into account that not all attributes are available to all selectors. URL's and URI's are not allowed.

#### CSS attributes

font-family (only for psp) (Default value: Arial)

Allowed values: "Arial", "Helvetica", "Times New Roman", "Times", "Courier New", "Courier", "Sans Serif". Webfonts are currently not supported.

font-size (only for psp) Default value: 12px

Sizes should be defined in Pixels (px) and/or EMs (em) with no space between value and unit.  
range for em: 0.75 - 1.0

range for px: 12 - 16

font-style (only for psp) : Default value: "normal"

Allowed values: "normal", "italic"

color

Allowed formats: "#f0f0f0", "rgb(100,100,100)", "red"

text\_decoration (only for link, visited, hover and active)

Allowed values: "none", "underline"

#### CSS selectors

Psp

link (a)

visited (a:visited)

hover (a:hover)

active (a:active)

#### styles object example

{

"psp": {

"font-family": "Times New Roman",

"color": "green",

"font-size": "14px",

"font-style": "italic"

},

"link": {

"color": "red",

"text-decoration": "underline"

},

"visited": {

"color": "green",

"text-decoration": "underline"

},

"active": {

"color": "yellow",

"text-decoration": "underline"

},

"hover": {

"color": "rgb(100,66,200)",

"text-decoration": "underline"

}}

## Integrating the external “continue button”

If configured in the JS library a continue button will appear next to the selected payment method. In most scenarios an external button will be used though. Please follow these simple steps to configure an external button.

In the JS library use the following setup:

buttonLocation: "outside",  
disableContinue: "continueButton",  
enableContinue: "continueButton",

PayPal recommends disabling the button if there is no payment method selected but it is not required. Replace “continueButton” with your button ID.

Example button:

<button

type="submit"

id="continueButton"

onclick="PAYPAL.apps.PPP.doCheckout();">

Checkout Now

</button>

In this scenario a click on the button will directly lead to the hosted PayPal pages.

If there is an additional page prior to the redirect, please add the onContinue function to the configuration object of the PayPal PLUS library.

onContinue: function () {  
window.location = "nextpage.php";  
},

Add this to the continue button

<button

type="submit"

id="continueButton"

onclick="ppp.doContinue(); return false;">

Checkout

</button>

On this nextpage.php the PayPal PLUS library needs to be loaded and the checkout function added to the button.

<button  
type="button"  
onclick="PAYPAL.apps.PPP.doCheckout();">  
PayPal PLUS Checkout Now  
</button>

## Update (Patch) payment

In a best practice integration user specific data like a name or shipping address is not submitted prior to the user selecting a PayPal payment method. Once the buyer indicates that he wants to use one of the payment methods provided by PayPal the payment resource should be updated with user specific data.

In addition the amount object can be updated. When replacing values within the object make sure to submit all properties you wish to keep. Patch replaces the entire object and not just the individual value.

Please note: The line items are fixed at this stage and must not be updated.

Method: PATCH

Resource: https://api.sandbox.paypal.com/v1/payments/payment/PAY-3GA34907N1379922SKYN25II

**Sample PATCH Request**

[0] => Array

(

[op] => replace

[path] => /transactions/0/amount

[value] => Array

(

[total] => 0.01

[currency] => EUR

[details] => Array

(

[subtotal] => 4.99

[shipping] => 1.00

[shipping\_discount] => -5.98

)

[1] => Array

(

[op] => add

[path] => /transactions/0/item\_list/shipping\_address

[value] => Array

(

[recipient\_name] => Max Mustermann

[line1] => Lieferstr. 1

[city] => Berlin

[country\_code] => DE

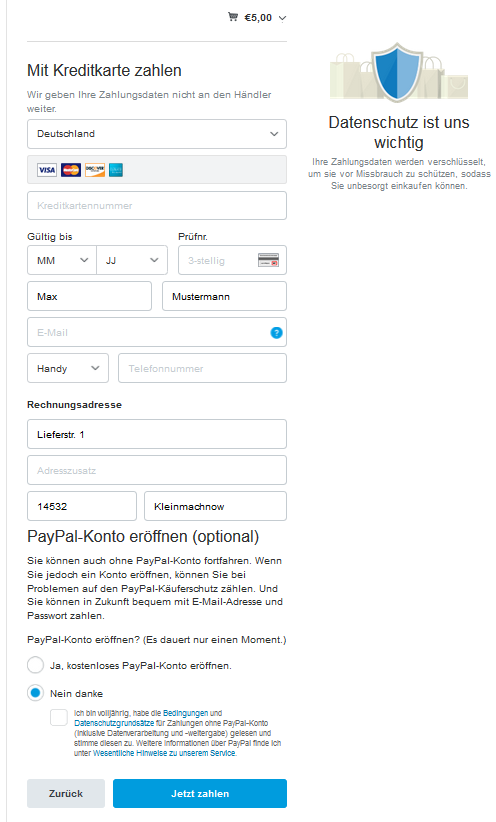
[postal\_code] => 12345

)

In the response the updated Resource is returned.

## Redirect to authorize payment

Depending on the selected payment method the user will be directed to the correct landing page. (Either PayPal Login, Credit Card Credentials, ELV details or invoice). This redirect is initiated by either the inside button or – if you integrated the “continue button” – by the PAYPAL.apps.PPP.doCheckout function.



After completion the user is returned to the returnURL as defined in Create Payment.

## Look Up Payment

Please use the GET URL as returned in the Create Payment

Method: GET

Resource: https://api.sandbox.paypal.com/v1/payments/payment/PAY-3GA34907N1379922SKYN25II

Look up payment is by definition optional but we recommend using it on return to the final review page. The data returned should be used to populate the shipping address. The user must not be allowed to change order details but can edit the shipping address, as long as the latter is submitted in a subsequent PATCH Payment call. An invoice number, shipping and handling charges as part of the amount object can be added as well.

The payment method should be referenced as “processed by PayPal”.

## Update (Patch) payment

After a buyer has approved a transaction on the PayPal hosted pages a typical use case fpr an additional PATCH call would be updating the shipping address and/or adding an invoice number

[0] => Array

(

[op] => add

[path] => /transactions/0/invoice\_number

[value] => My unique Invoice ID 12345678ABCD

)

[1] => Array

(

[op] => add

[path] => /transactions/0/item\_list/shipping\_address

[value] => Array

(

[recipient\_name] => Erika Mustermann

[line1] => Neue Lieferstr. 7

[city] => Berlin

[country\_code] => DE

[postal\_code] => 12345

)

## Execute Payment

Execute payment completes the transaction. It can either be initiated from the final review page or in the background upon return from the hosted pages when there is no final review page (commit flow). Any updates and changes to shipping address, shipping and handling charges must prior be submitted to PayPal via PATCH.

**Please note:** The actual PayPal transaction ID and the status of the PayPal transaction are returned in the related\_resources array as:

[id] => 0XK08547HM111121A

[state] => completed

**Sample Request:**

Method: POST

Resource: https://api.sandbox.paypal.com/v1/payments/payment/PAY-3GA34907N1379922SKYN25II/execute

Data: {

"payer\_id": "HE4WLMC3N3AWN"

}

# Integrating manual Bank Transfer (mEFT)

As part of PayPal’s funding mix a buyer may select manual bank transfer as funding instrument for a PayPal payment. In this case the transaction will be non-instant and the payment status returned “pending” until the buyer settles the amount with PayPal.

In this scenario PayPal will return the following in the execute payment response:

"reason\_code": "ECHECK",

"state": "pending",

"payment\_instruction": {

"reference\_number": "C12W2P058PNFV5V",

"recipient\_banking\_instruction": {

"routing\_number": "501 108 00",

"bank\_name": "J.P. Morgan AG, Frankfurt",

"bank\_identifier\_code": "CHASDEFX",

"account\_holder\_name": "PayPal Pte. Ltd",

"account\_number": "6161604670",

"international\_bank\_account\_number": "DE86 5011 0800 6161 6046 70"}

To accommodate this scenario, at least two steps are required:

* define a bankpending\_url in the experience profile and / or
* capture the reason code “echeck” and when returned display the bank details on the success page with instruction to wire the due amount to bank account shown.
* In order to receive the status updates for this pending transaction, webhooks should be integrated. The transaction will also appear in the settlement files once it is completed and can be queried via look up.

# Additional Steps to Integrate Pay upon Invoice

## Before you start:

Pay upon Invoice is enabled on account level. There are no additional steps required to trigger invoice in the PayPal PLUS payment selection page. Invoice transactions will be marked as such and they return a payment instructions object in the execute payment response. Other than this they are treated like any other PayPal transaction.

As per default Pay upon Invoice is offered for total amounts between 1,50 and 1470 EUR.

## Enabling Pay upon Invoice in Sandbox

To test pay upon invoice in Sandbox you can enable the payment method in PLUS by adding the following parameter to the configuration object:

showPuiOnSandbox: "true",

## Step 1: Retrieving Payment Instructions:

When a user selects Pay upon Invoice (PuI) the following parameters are returned in a Payment Instruction Object as part of the response to a successful Execute Payment Request.

**reference\_number** string ID of payment instruction

**instruction\_type** string Type of payment instruction

**recipient\_banking\_instruction** object Recipient bank Details.

**amount** object Base object for all financial value related fields   
 (balance, payment due, etc.)

**payment\_due\_date** string Date by which payment should be received

An addition a HATEAOS link to retrieve payment instructions is returned.

Payment instructions and HATEAOS link to retrieve payment instructions must be stored in a database table where it can later be retrieved for the invoice print.

**Sample response Execute Payment Response:**

[payment\_instruction] => Array

(

[reference\_number] => 7FC35911MD7697828

[instruction\_type] => PAY\_UPON\_INVOICE

[recipient\_banking\_instruction] => Array

(

[bank\_name] => Deutsche Bank

[account\_holder\_name] => PayPal Europe

[international\_bank\_account\_number] => DE07120700888000XXXXXX

[bank\_identifier\_code] => DEUTDEDBPAL

)

[amount] => Array

(

[value] => 4.99

[currency] => EUR

)

[payment\_due\_date] => 2015-09-24

[links] => Array

(

[0] => Array

(

[href] => https://api.paypal.com/v1/payments/payment/PAY-1VG08356YH269600PKXOBRAY/payment-instruction

[rel] => self

[method] => GET

Payment instructions may also be retrieved at any later stage by calling the Lookup Payment Instructions for a payment resource.

**Sample response GET Payment Instructions:**

lookup payment instructions] => Array

(

[reference\_number] => 7FC35911MD7697828

[instruction\_type] => PAY\_UPON\_INVOICE

[recipient\_banking\_instruction] => Array

(

[bank\_name] => Deutsche Bank

[account\_holder\_name] => PayPal Europe

[international\_bank\_account\_number] => DE07120700888000XXXXXX

[bank\_identifier\_code] => DEUTDEDBPAL

)

[amount] => Array

(

[value] => 4.99

[currency] => EUR

)

[payment\_due\_date] => 2015-09-24

[links] => Array

(

[0] => Array

(

[href] => https://api.paypal.com/v1/payments/payment/PAY-1VG08356YH269600PKXOBRAY/payment-instruction

[rel] => self

[method] => GET

)

)

)

## Presentation on the Success Page

On successful completion of a transaction the buyer is redirected to a Success Page. On this page a buyer should be advised that the amount is to be settled with PayPal and the Payment Instructions as returned by PayPal should be displayed.

## Step 2: Managing invoice transactions

Transactions flagged as pay upon invoice are regular PayPal transactions so they are handled in the same way as any other PLUS transaction. There is one significant difference though: The payment instruction returned by PayPal must be displayed on the printed invoice that is generated in the administration panel of the shop.

This invoice should only contain the PayPal bank details as returned in the GET call. The merchant bank details must not be shown.

The invoice details must contain (taken from the example above):

Bank name: "Deutsche Bank",

Account holder name "PayPal Europe",

IBAN "DE07120700888000XXXXXX",

BIC "DEUTDEDBPAL"

Amount due / currency "4.99 EUR"

Payment due date "2015-09-24"  
Reference "7FC35911MD7697828"

The payment instructions must always be taken from the values as in returned in the instruction. None of the values from the payment instruction object may be hard coded including bank, recipient and IBAN. These details may be retrieved from the database table or looked up by executing another GET call for the payment resource ID in question.

A legal note must be added to the invoice:

*„[Händlername] hat die Forderung gegen Sie im Rahmen eines laufenden Factoringvertrages an die PayPal (Europe) S.àr.l. et Cie, S.C.A. abgetreten. Zahlungen mit schuldbefreiender Wirkung können nur an die PayPal (Europe) S.àr.l. et Cie, S.C.A. geleistet werden.“*

**Please Note:** IBAN is not a static value and must not be saved for future reference.

If prior to the invoice print a partial refund was initiated these changes must be reflected in the payment instructions.

# Integration using requireJS

Example for mostly integrated use case: immediate checkout with outside button

**checkout.html (excerpt):**

<button

id="outsideButton"

type="submit"

onClick="require(['OutsideButton']); return false;">

<your button title>

</button>

...

<script src="static/require.js"></script>

...

<script type="text/javascript">

<!-- kick off PP+ -->

require(["main"], function onConfigLoaded() {

require(['CheckoutStart']);

});

</script>

**main.js:**

require.config({

paths: {

"PPP": "https://www.paypalobjects.com/webstatic/ppplus/ppplus.min"

}

});

**CheckoutStart.js:**

define("CheckoutStart", ["PPP"], function (PPP) {

"use strict";

return PPP(<your-configuration-object>);

});

**OutsideButton.js:**

define("OutsideButton", ["CheckoutStart"], function (configuredPPP) {

"use strict";

// if onContinue callback is given this will be executed, take the user to

another page

// if no onContinue callback given this will execute doCheckout internally and

redirect

// the user to the selected payment method checkout page

configuredPPP.doContinue();

});

If you choose the elsewhere button experience you also need to call doCheckout():

**elsewhere.html (excerpt):**

<button

id="elsewhereButton"

type="submit"

onClick="require(['ElsewhereButton']); return false;">

<your button title>

</button>

...

<script data-main="main" src="static/require.js"></script>

...

<script type="text/javascript">

<!-- NO kick off needed here for PP+ since the payment selection page was rendered

in checkout.html -->

</script>

**ElsewhereButton.js:**

define("ElsewhereButton", ["PPP"], function (PPP) {

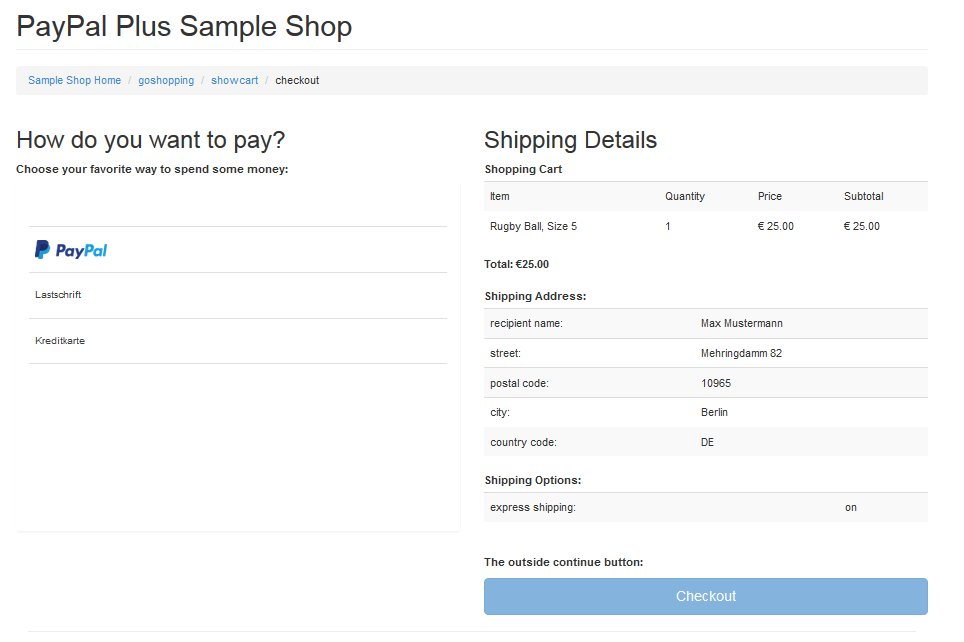
"use strict";

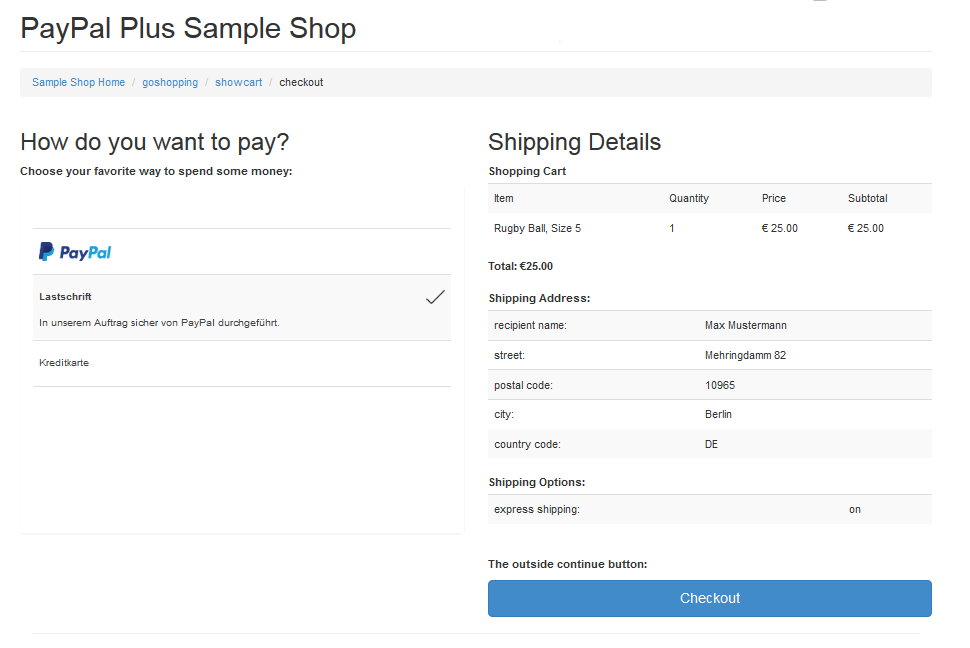
PPP.doCheckout();

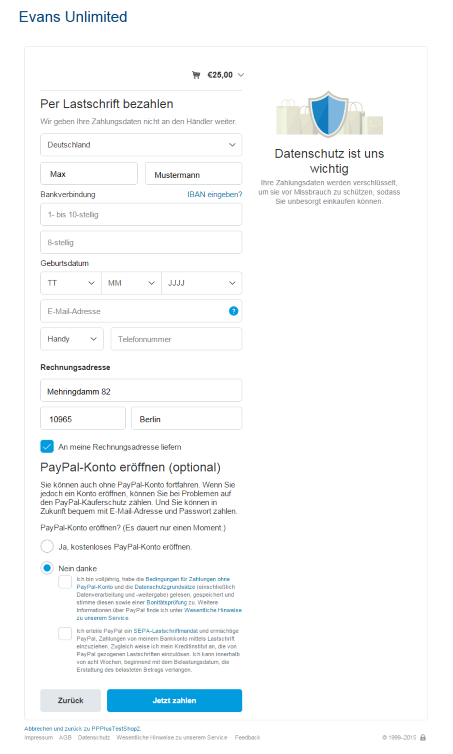
});

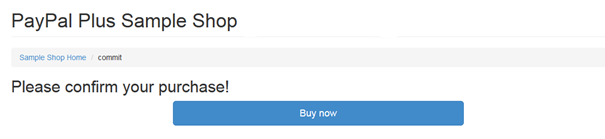
# PayPal PLUS Walk Through

## Checkout with ELV

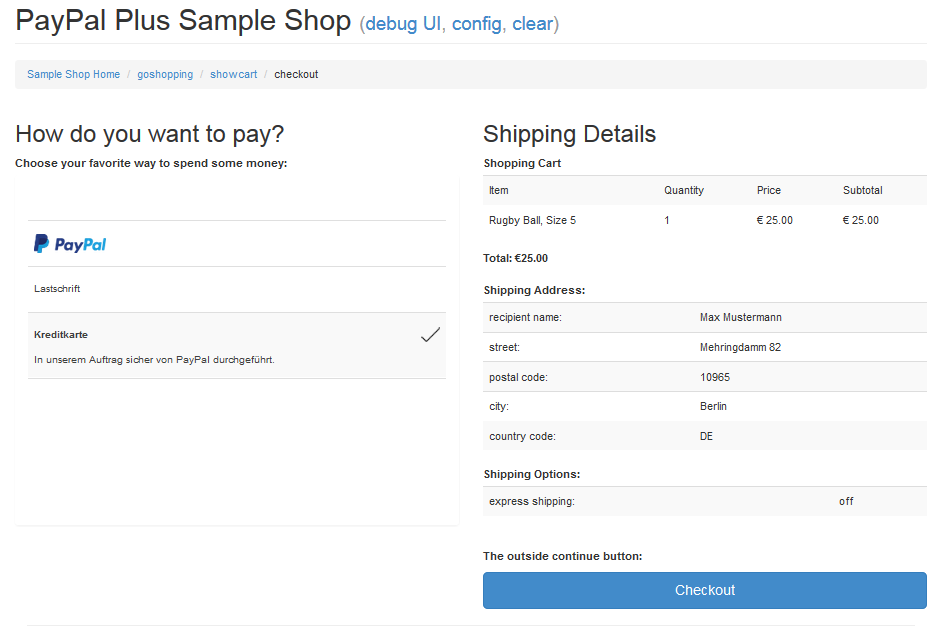


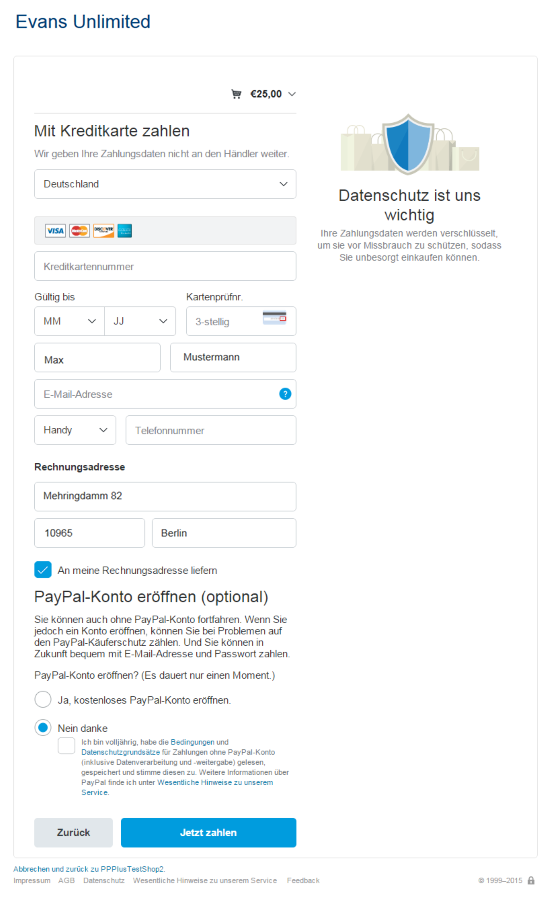


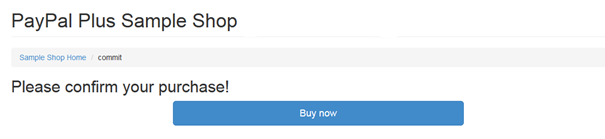




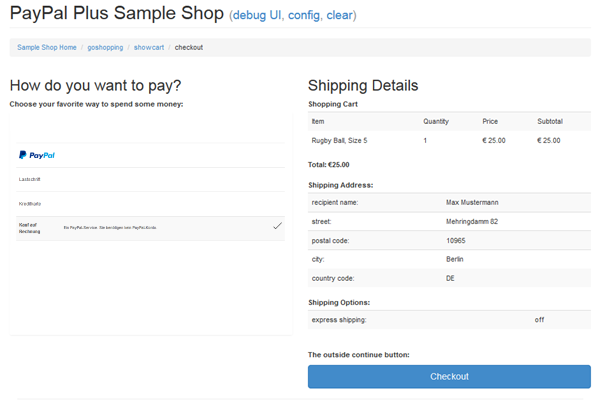
## Checkout with Credit Card

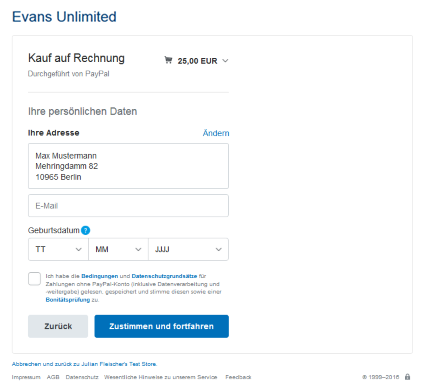






## Checkout with Pay upon Invoice





# Appendix

## Technical Documentation

Account set up and configuration

**Sandbox and User Guide**

The PayPal Sandbox self-service testing environment

<https://developer.paypal.com/>

<https://developer.paypal.com/webapps/developer/docs/classic/lifecycle/ug_sandbox/>

## API Documentation

**PayPal API Reference**

<https://developer.paypal.com/webapps/developer/docs/api/>

**PayPal REST SDK:**<https://github.com/paypal/PayPal-PHP-SDK/releases/tag/v1.2.1>

## Reporting

**Settlement Report Specification (STL v7)**  
Detailed information about the structure of the daily Settlement Report

<https://cms.paypal.com/cms_content/US/en_US/files/developer/PP_LRD_Gen_SettlementReport.pdf>

**Transaction Details Report (TRR v10)**  
Detailed information about the structure of the daily Transaction Details Report

<https://cms.paypal.com/cms_content/US/en_US/files/developer/PP_LRD_Gen_TransactionDetailReport.pdf>

**Downloadable Dispute Report (DDR v7)**  
Detailed information about the structure of the daily Dispute Report

<https://cms.paypal.com/cms_content/US/en_US/files/developer/PP_LRD_GenDisputeReport.pdf>