PayPlug API Full Reference

[PHP]

Version: 1.0.7
Date: 23-05-2019

Overview

Based on simple REST concepts, the PayPlug API returns JSON objects for your payments, refunds, installment plans and notifications.

All API URLs listed in this documentation are relative to **https://api.payplug.com**. For example, the create payment API call is reachable at **https://api.payplug.com/v1/payments**.

> All the features with the **Pro** or **Premium** label are only available if you subscribed to the corresponding offer.

Authentication :

Example:

```
<?php
// Not required using composer
require_once('PATH_TO_PAYPLUG/payplug_php/lib/init.php');
\Payplug\Payplug::setSecretKey('sk_live_43b7e007298f57f732800a52');
```

You must authenticate to the PayPlug API by providing in all your requests one of your secret Keys available in the **My account** section of the PayPlug Portal.

Authentication occurs using the HTTP Authorization header. You do not need to provide your password in your requests.

All API requests must use HTTPS: every call made over simple HTTP will fail.

Access LIVE and TEST modes using the same endpoint. To switch between each mode you have 2 differents secret keys:

Mode	Secret key example
Live	sk_live_43b7e007298f57f7aedee32800a52301
Test	sk_test_7c5cb3b54abcf5062f056639e809368c

You simply have to provide the secret key from the mode you wish to use.

Because the prefix is different, you will easily recognize them. ;)

Notifications :

Example:

```php
<?php
require_once('PATH_TO_PAYPLUG/payplug_php/lib/init.php');
\Payplug\Payplug::setSecretKey('sk_test_500e54fxx362355a7xx00ad');

$input = file_get_contents('php://input');
try {
  $resource = \Payplug\Notification::treat($input);

  if ($resource instanceof \Payplug\Resource\Payment
    && $resource->is_paid
    // Ensure that the payment was paid
    ) {
    // Process a paid payment.
  } else if ($resource instanceof \Payplug\Resource\Refund) {
    // Process the refund.
  }
}
catch (\Payplug\Exception\PayplugException $exception) {
 // Handle errors
}
```

```
{
  "id": "re_3NxGqPfSGMHQgLSZH0Mv3B",
  "payment_id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "object": "refund",
  "is_live": true,
  "amount": 358,
  "currency": "EUR",
  "created_at": 1434012358,
  "metadata": {
    "customer_id": 42710,
    "reason": "The delivery was delayed"
  }
}
```

You can configure a callout notification to be sent to your application when creating a payment or an installment plan.

PAYMENT

You will get a notification when the payment fails, is paid, is refunded **or when the payment times out 15 minutes after the creation if not paid**.

When you create a payment, specify the `notification_url` parameter. Once the payment has been created, notifications about this payment will be sent through a **POST request** to the provided url.

For a refund request, you will get a refund object on the `notification_url` specified when you created the payment.

> Please be careful about the **is_live** value, it will allow you to identify if the notification is about a LIVE or a TEST payment.

INSTALLMENT PLAN

You will get a notification when an installment plan is aborted, or whenever an installment (a payment) succeeds or fails.

When you create an installment plan, specify the `notification_url` parameter. Notifications about this installment plan, and all related payments and refunds will be sent through a **POST request** to the provided url.

NOTIFICATION PERIOD

According to the payment situation a notification can be sent shortly or after 15 minutes. Here is a summary of the different possible cases:

Case	Period	Error code	Error message
The payment was paid	Shortly	-	
The payment attempt failed	15 min	Last attempt error code	Last attempt error message.
The payment was not paid and then it was cancelled by the customer	Never	canceled	The payment was canceled by the customer.
The payment was not paid and then it was aborted by the API	Never	aborted	You have aborted the transaction.
The payment has not been paid	15 min	timeout	The customer has not tried to pay and left the payment page.
The payment was cancelled after several attempts	15 min	Last attempt error code	Last attempt error message.
The payment was not paid after several attempts	15 min	Last attempt error code	Last attempt error message.
The payment was paid after several failed attempts	Shortly	-	

The **notification_url** must be publicly accessible over the Internet! Notifications will not works with localhost, proxies, firewall...

Pagination :

Example:

```
<?php
$perPage = 5;
$page = 1;
$payments = \Payplug\Payment::listPayments($perPage, $page);
```

Example response:

```
{
  "object": "list",
  "page": 1,
  "per_page": 1,
  "has_more": true,
  "data": [{
    "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
    "object": "payment",
    "is_live": true,
    "amount": 3300
  }]
}
```

The PayPlug API uses the same pagination structure for requests to list payments and refunds.

URL ARGUMENTS:

Key	Description
per_page	Number of objects to be listed in the request.

Key	Description
page	Page of results to be used in the request.

The **page** parameter is based on the maximum number listed in the **per_page** parameter. For exemple, if you have 12 objects and you have specified into your request: per_page=8 , if you want to see the part of the list with the ninth object you should use: page=1 .

If you don't specify the **page** parameter in your request, you will receive the last 10 objects of the list.

RESPONSE DETAIL:

Key	Description
page *integer*	List page do display (Default is: 0)
per_page *integer*	Maximum number of objects listed in data
has_more *boolean*	Whether there are more elements or not after this page
data *JSON object*	List of requested objects

The **per_page** parameter is limited to 100. You can't retrieve more than 100 objects in a single API request. If you do so, the API will answer a list of 100 objects, without issuing an error.

Metadata :

All PayPlug's API objects support additional custom metadata.

Metadata enables you to store information as user-defined key-value pairs. You can define up to 10 keys, keys names have a maximum length of 20 characters, and each data field have a maximum of 500 characters.

Example refund object:

```
{
  "id": "re_281362",
  "object": "refund",
  "metadata": {
    "transaction_id": "tsct_201506023456",
    "customer_id": 58790,
    "product_id": "ts_blk_00234",
    "shipping": "The delivery was delayed"
  }
}
```

For example, when you create a payment, you could add information about your customer such as a customer ID, a product ID, a transaction ID, shipping information…

METADATA EXAMPLE:

Key	Description
transaction_id *string*	Id of the customer transaction related to the payment.
customer_id *integer*	Id of the customer.
product_id *string*	Id of the product bought by the customer.
shipping *string*	Information about the shipping (shipping rates, delivery options, details…).

The Metadata added must be a JSON object, with a maximum of one depth level, 10 keys, and a total length up to 500 characters long.

Errors :

Example response:

```
{
  "object": "error",
  "message": "You did not provide any Authentication HTTP header. You must authenticate with your LIVE or TEST AP
I key.",
  "id": "2fa981d70eaa11e58abba8206650bd9f"
}
```

The PayPlug API use standard HTTP response status code. Errors are returned with additional data in the body of the return call (JSON-formatted).

Key	Description
message *string*	A short description of the cause of the error.
id *string*	Error ID.
details *string*	For a 400 error, this additional field is added, so that you can easily know what are the wrong or missing parameters you sent.

HTTP STATUS CODES

Example response:

```
{
  "object": "error",
  "message": "Wrong or missing parameters (see details).",
  "id": "349c30b00eac11e5a301a8206650bd9f",
  "details": {
    "amount": "The minimum amount is 1 EUR (100 cents). The amount you provided is 5 cents.",
    "customer": {
      "first_name": "The first_name is too long. The maximum length is 100 characters."
    },
    "force_3ds": "This parameter must be a boolean (true or false)."
  }
}
```

Codes in the 4xx range indicate an error. An object is returned to your requests including information about the error with a reference ID.

400 error code (Bad Request) is including additional information about which parameters are wrong or missing in your request.

Codes in the 2xx range indicate your request worked whitout any issues and 5xx range appear when something went wrong on PayPlug's end.

Code	Description
200	OK – The request has succeeded as expected. The result of the request is available in the response.
201	Created – The request has been executed and a new resource has been created.

Code	Description
400	Bad Request – The request could not be executed due to wrong or missing parameter. The response will contain more information.
401	Unauthorized – The request requires authentication and has been refused for the provided API key.
403	Forbidden error – The request has succeed, but the access to this ressource is forbidden.
404	Resource not found – The requested resource could be found.
405	Method not allowed – The request used a method not supported by this resource.
50x	Server errors – The API is currently unable to handle the request due to an unexpected temporary condition. You can try to resend the request after a couple of minutes.

PAYMENT FAILURE CODES

Unsuccessful payment return information about the error in the response with a failure code and a failure message.

Types	Description
processing_error	Error while processing the card.
card_declined	The card has been rejected.
insufficient_funds	Insufficient funds to cover the payment.
3ds_declined	The 3-D secure code has been rejected.
incorrect_number	The card number is incorrect.
fraud_suspected	Payment rejected because a fraud has been detected.
method_unsupported	The payment method is not supported (e.g. e-carte bleue).
card_scheme_mismatch	The card number does not match with the selected brand.
card_expiration_date_prior_to_last_installment_date	The card expiration date is prior to the last installment date.
aborted	Payment has been aborted with the Abort a payment feature.
timeout	The customer has not tried to pay and left the payment page.

INSTALLMENT PLAN FAILURE CODES

Unsuccessful installment plan return information about the error in the response with a failure code and a failure message.

Types	Description
failed_payment	A payment has failed.
aborted	Installment plan has been aborted with the Abort an installment plan feature.

API Testing :

BEST PRACTICES WHEN TESTING THE PAYPLUG API

When integrating the PayPlug API you should use the TEST mode to ensure the following:

1. Card number, expiration, amount and CVC are set correctly
2. Error messages from the API are handled and displayed correctly
3. Sensitive information about the cards (number and CVC) are not stored on your server
4. All information and values related to payments sent to the API are valid and correctly formatted
5. Never hardcode your secret key into your code, use configuration values

To use the API TEST mode, be sure to use the test secret key.

CARD NUMBERS AVAILABLE FOR TESTING

Using TEST mode, the following test cards are available for testing your integration:

Test cards numbers	Expected
4242 4242 4242 4242 *Visa*	Success
5500 0055 5555 5559 *MasterCard*	Success
4000 0000 0000 0051 *Visa*	Failure code: card_declined
4000 0000 0000 0085 *Visa*	Failure code: processing_error
4000 0000 0000 0077 *Visa*	Failure code: insufficient_funds
5184 6800 0000 0170 *MasterCard*	Failure code: ds_declined
5184 6800 0000 0097 *MasterCard*	Failure code: incorrect_number
5184 6800 0000 0121 *MasterCard*	Failure code: fraud_suspected

Any CVC numbers and any expiration date in the future will be considered valid.

All others card numbers even if they are valid will fail (Failure code: card_declined).

More information about error codes is available here.

Test cards numbers will not work in LIVE mode.

Payments

Use the /payments endpoint to create a payment, list the payments or retrieve a payment.

PAYMENTS ENDPOINT REFERENCE:

Method	Endpoint	Usage
POST	/v1/payments	Create a payment
GET	/v1/payments/{payment_id}	Retrieve a payment
PATCH	/v1/payments/{payment_id}	Abort a payment
PATCH	/v1/payments/{payment_id}	Capture a payment
PATCH	/v1/payments/{payment_id}	Update a payment
GET	/v1/payments	List all payments

The payment object :

Example:

```
{
  "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "object": "payment",
  "is_live": true,
  "amount": 3300,
  "amount_refunded": 0,
  "authorization": null,
  "currency": "EUR",
  "created_at": 1434010787,
  "installment_plan_id": null,
  "is_paid": true,
  "paid_at": 1555073519,
  "is_refunded": false,
  "is_3ds": false,
  "save_card": false,
  "card": {
    "last4": "1800",
    "country": "FR",
    "exp_month": 9,
    "exp_year": 2017,
    "brand": "Mastercard",
    "id": null
  },
  "customer": {
    "first_name": "John",
    "last_name": "Watson",
    "email": "john.watson@example.net",
    "phone_number": null,
    "address1": null,
    "address2": null,
    "postcode": null,
    "city": null,
    "country": null,
    "language": "en"
  },
  "hosted_payment": {
    "payment_url": "https://www.payplug.com/pay/5iHMDxy4ABR4YBVW4UscIn",
    "return_url": "https://example.net/success?id=42",
    "cancel_url": "https://example.net/cancel?id=42",
    "paid_at": 1434010827,
    "sent_by": null
  },
  "notification": {
    "url": "https://example.net/notifications?id=42",
    "response_code": 200
  },
  "failure": null,
  "description": null,
  "metadata": {
    "customer_id": 42
  }
}
```

```php
<?php
// Get an attribute of the payment object:
// Print the payment amount
echo htmlentities($payment->amount);
// Print the payment id
echo htmlentities($payment->id);
// Print the payment creation time in an human readable way
echo htmlentities(date('d/m/Y H:i:s', $payment->created_at));

// Chain relations between objects:
// Print the payment URL
echo htmlentities($payment->hosted_payment->payment_url);
// Print the customer email
echo htmlentities($payment->customer->email);

// Retrieve metadata
echo htmlentities($payment->metadata['customer_id']);
echo htmlentities($payment->metadata['type']);
```

OBJECT ATTRIBUTES:

Key	Description
id *string*	Payment ID.
object *string*	Value is: payment .
is_live *boolean*	true for a payment in LIVE mode, false in TEST mode.
amount *positive integer*	Amount of the payment in cents.
amount_refunded *positive integer or zero*	Amount that has been refunded in cents.
authorization *JSON object*	Information about the authorization in case of a deferred payment, null otherwise.
	authorized_amount: The amount that was authorized. *positive integer*
	authorized_at: Date at which the payment was authorized by the customer, null if the payment has not yet been authorized. *timestamp*
	expires_at: Date at which the authorization expires, null if the payment has not yet been authorized. *timestamp*
currency *currency*	Currency code (three-letter ISO 4217) in which the payment was made.
installment_plan_id *string*	ID of the installment plan related to this payment, if any.
is_paid *boolean*	true if the payment has been paid, false if not.
paid_at *timestamp*	Date at which the payment has been paid, null if the payment has not yet been paid.
is_refunded *boolean*	true if this payment has been fully refunded, false if not.
is_3ds *boolean or null*	true if the payment was processed using 3-D Secure.
save_card *boolean*	true if the payment was used to save a card. On the payment page, saving a card was mendatory.
allow_save_card *boolean*	Alternative to **save_card**. true if the payment gave the possibility to a customer to save a card. On the payment page, saving a card was optional.
created_at *timestamp*	Creation date.
card *JSON object*	Information about the card used for the payment.
	last4: Last 4 digits of the card number. *string*
	country: Country code (two-letter ISO 3166). *string*

Key	Description
	exp_year: Credit card expiration year. *integer*
	exp_month: Credit card expiration month. *integer*
	brand: Credit card brand, can be Mastercard , Maestro , Visa or CB . *string*
	id: Credit card ID, available when a payment has been created with save_card=true , or has been created with this id . *string*
customer *JSON object*	Information about the customer.
	email: Customer email address. *string*
	phone_number: Customer phone number (digits only international format, no + sign). *string*
	first_name: Customer first name. *string*
	last_name: Customer last name. *string*
	address1: Customer address line 1 (Street address/PO Box/Company name). *string*
	address2: Customer address line 2 (Apartment/Suite/Unit/Building). *string*
	postcode: Customer Zip/Postal code. *string*
	city: Customer city. *string*
	country: Customer country code (two-letter ISO 3166). *string*
	language: Customer language code (two-letter ISO 639-1). *string*
hosted_payment *JSON object*	Information about the payment.
	payment_url: Payment URL where you should redirect your customer to. *string*
	return_url: URL where the customer will be redirected after the payment page whether it succeeds or not. *string*
	cancel_url: URL where the customer will redirected after a click on "**Cancel Payment**". *string*
	paid_at: Date at which the payment goes through. null if the payment has not been not paid yet or has failed. *timestamp* ***deprecated***
	sent_by: By what means the payment URL was sent to customer, if any. *string*
notification *JSON object*	Data related to notifications.
	url: URL that will be used by PayPlug to send notifications. *string*
	response_code: Integer http response code received when calling the url of your notification page. *integer*
failure *JSON object or null*	Information for unsuccessful payments.
	code: Payment failure code if the payment has failed. *string*
	message: Descriptive message explaining the reason of the payment failure. *string*
description *string* ***optional***	Description shown to the customer.
metadata *JSON object*	Custom metadata object added when creating the payment.

Tips: The **response_code** can be very useful for debugging because it gives you important information when you do not receive the notifications.

Create a payment :

Example:

```
 <?php
$payment = \Payplug\Payment::create(array(
  'amount'            => 3300,
  'currency'          => 'EUR',
  'save_card'         => false,
  'customer'          => array(
    'email'           => 'john.watson@example.net',
    'first_name'      => 'John',
    'last_name'       => 'Watson'
  ),
  'hosted_payment'   => array(
    'return_url'      => 'https://example.net/success?id=42',
    'cancel_url'      => 'https://example.net/cancel?id=42'
  ),
  'notification_url' => 'https://example.net/notifications?id=42',
  'metadata'          => array(
    'customer_id'     => 42
  )
));

$payment_url = $payment->hosted_payment->payment_url;
$payment_id = $payment->id;
```

Example response:

```
{
  "amount": 3300,
  "amount_refunded": 0,
  "authorization": null,
  "card": {
    "brand": null,
    "country": null,
    "exp_month": null,
    "exp_year": null,
    "id": null,
    "last4": null
  },
  "created_at": 1449157171,
  "currency": "EUR",
  "customer": {
    "address1": null,
    "address2": null,
    "city": null,
    "country": null,
    "email": "john.watson@example.net",
    "phone_number": null,
    "first_name": "John",
    "last_name": "Watson",
    "postcode": null,
    "language": "en"
  },
  "failure": null,
  "hosted_payment": {
    "cancel_url": "https://example.net/cancel?id=42",
    "paid_at": null,
    "payment_url": "https://www.payplug.com/pay/test/2DNkjF024bcLFhTn7OBfcc",
    "return_url": "https://example.net/success?id=42",
    "sent_by": null
  },
  "id": "pay_2DNkjF024bcLFhTn7OBfcc",
  "is_3ds": null,
  "is_live": false,
  "is_paid": false,
  "paid_at": null,
  "is_refunded": false,
  "installment_plan_id": null,
  "description": null,
  "metadata": {
    "customer_id": "42"
  },
  "notification": {
    "response_code": null,
    "url": "https://example.net/notifications?id=42"
  },
  "object": "payment",
  "save_card": false
}
```


To accept a payment from your customer, first create a payment and then redirect to the **payment_url** provided in the response object. Once your customer has attempted to pay, he will be redirected to your return_url .

Once a payment URL is obtained and the payment page is displayed to your client, he will be able to try multiple times to pay using the same payment URL, in case the previous payment attempts fail.

A notification will also be sent (if **notification_url** was provided) in case the payment is successful (is paid).

The payment URL will be available for 15 minutes. If the payment was not successfully paid during this time period, it will automatically expire after 15 minutes and a notification will be sent (if **notification_url** was provided) containing the last attempt's failure code and failure reason.

You can create a hosted payment that will never expire (the **hosted_payment.payment_url** will always be available), by providing **hosted_payment.sent_by** value:

- EMAIL or SMS : an email or a SMS will be sent to the customer by PayPlug.
- OTHER : you are responsible for sharing the payment URL to your customer.

If you saved a card and use it to create a payment, the payment will be paid directly, so that there is no **payment_url**.

If you create a deferred payment (see parameter authorized_amount), then when the customer pays using the payment page, the payment is only *authorized*, meaning the funds are locked in the customer's account, but no transaction is issued yet. You should then *capture* the payment up to 7 days following the authorization (see Capture a payment).

For a better integration to your site, you can create a more customised look and feel to your payment page with your own custom logo, background and colors.

PARAMETERS:

Key	Description
amount *positive integer*	Amount in cents, for instance use **4207** for a **42.07€** payment. Currently, the minimum and maximum amounts are 100 and 2,000,000 cents.
authorized_amount *positive integer*	In order to create a deferred payment, use this field for the payment's amount instead of amount .
currency *currency* ***required***	Currency code (three-letter ISO 4217). For now, Euro is the only currency supported (value: EUR).
customer *JSON object* ***optional***	All customer attributes are optional. First name, last name and email will be asked on the payment page if you do not provide them.
	first_name: Customer first name (maximum length: 100 characters). *string* ***optional*** (***required*** if payment_method is not null).
	last_name: Customer last name (maximum length: 100 characters). *string* ***optional*** (***required*** if payment_method is not null).
	email: Customer email address (maximum length: 255 characters). *email* ***optional*** (***required*** if payment_method is not null, or if hosted_payment.sent_by is EMAIL).
	phone_number: Customer phone number (digits only international format, no + sign). *phone number* ***optional*** (***required*** if hosted_payment.sent_by is SMS).
	address1: Customer address line 1 (maximum length: 255 characters). *string* ***optional***
	address2: Customer address line 2 (maximum length: 255 characters). *string* ***optional***
	postcode: Customer Zip/Postal code (maximum length: 16 characters). *string* ***optional***
	city: Customer city (maximum length: 100 characters). *string* ***optional***
	country: Customer country two-letter ISO 3166 code *code* ***optional***
	language: Customer language two-letter ISO 639-1 code *code* ***optional***. Supported languages: fr , en and it .
hosted_payment *JSON Object* ***optional***	All urls must start with "http://" or "https://" and must be accessible from anywhere on the Internet (which is not the case with local environments).
	return_url: URL where the customer will be redirected after the payment page whether it succeeds or not. *string* ***optional***
	cancel_url: URL where the customer will redirected after a click on "**Cancel Payment**". *string* ***optional***
	sent_by: To send a message to customer with payment URL, which will not expire. Supported values: EMAIL , SMS , and OTHER . *string* ***optional*** **Pro**
notification_url *string* ***optional***	URL that will be used by PayPlug to send notifications.

Key	Description
payment_method *string* ***optional*** **Premium**	The Card ID of a saved card, or a Token generated by PayPlug.js. If provided, hosted_payment must be NULL.
force_3ds *boolean* ***optional***	If true 3-D Secure will be activated for the payment, else, PayPlug will decide whether it should be activated or not based on default configuration.
save_card *boolean* ***optional*** **Premium**	If true , card information will be saved, and a card ID will be returned once the payment has been done. You must provide a customer when this is set to true .
allow_save_card *boolean* ***optional*** **Premium**	Alternative to **save_card**. If true , the customer will be able to save its card by checking a box on the payment page. A card ID will be returned once the payment has been done. You must provide a customer when this is set to true .
description *string* ***optional***	Description shown to the customer (maximum length: 80 characters).
metadata *JSON object* ***optional***	Custom metadata object.

Retrieve a payment :

Example:

```
<?php
$payment = \Payplug\Payment::retrieve('pay_5iHMDxy4ABR4YBVW4UscIn');
```

Example response:

```
{
  "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "object": "payment",
  "is_live": true,
  "amount": 3300,
  "amount_refunded": 0,
  "authorization": null,
  "currency": "EUR",
  "created_at": 1434010787,
  "installment_plan_id": null,
  "is_paid": true,
  "paid_at": 1555073519,
  "is_refunded": false,
  "is_3ds": false,
  "save_card": false,
  "card": {
    "last4": "1800",
    "country": "FR",
    "exp_month": 9,
    "exp_year": 2017,
    "brand": "Mastercard",
    "id": null
  },
  "customer": {
    "first_name": "John",
    "last_name": "Watson",
    "email": "john.watson@example.net",
    "phone_number": null,
    "address1": null,
    "address2": null,
    "postcode": null,
    "city": null,
    "country": null,
    "language": "en"
  },
  "hosted_payment": {
    "payment_url": "https://www.payplug.com/pay/5iHMDxy4ABR4YBVW4UscIn",
    "return_url": "https://example.net/success?id=42",
    "cancel_url": "https://example.net/cancel?id=42",
    "paid_at": 1434010827,
    "sent_by": null
  },
  "notification": {
    "url": "https://example.net/notifications?id=42",
    "response_code": 200
  },
  "failure": null,
  "description": null,
  "metadata": {
    "customer_id": 42
  }
}
```


Retrieves the information of a specific payment that has already been created.

The request will return a payment object if the ID provided is valid, and it will return an error object otherwise.

URL ARGUMENTS:

Key	Description
payment_id	ID of the payment to be retrieved.

Update a payment :

Example :

```
 <?php
$paymentId = 'pay_5iHMDxy4ABR4YBVW4UscIn';
$payment = \Payplug\Payment::retrieve($paymentId);

// To update metadatas keys
$data = array(
  "metadata"          => array(
    "customer_id" => 42,
    "comment"     => "asked questions about size"
  )
);
$update = payment->update($data);

// To remove all metadatas
$update = payment->update();
```

Example response:

```
{
  "amount": 3300,
  "amount_refunded": 0,
  "authorization": null,
  "card": {
    "brand": null,
    "country": null,
    "exp_month": null,
    "exp_year": null,
    "id": null,
    "last4": null
  },
  "created_at": 1449157475,
  "currency": "EUR",
  "customer": {
    "address1": null,
    "address2": null,
    "city": null,
    "country": null,
    "email": "john.watson@example.net",
    "phone_number": null,
    "first_name": "John",
    "last_name": "Watson",
    "postcode": null,
    "language": "en"
  },
  "failure": {
    "code": "aborted",
    "message": "You have aborted the transaction."
  },
  "hosted_payment": {
    "cancel_url": "https://example.net/cancel?id=42",
    "paid_at": null,
    "payment_url": "https://www.payplug.com/pay/test/5iHMDxy4ABR4YBVW4UscIn",
    "return_url": "https://example.net/success?id=42",
    "sent_by": null
  },
  "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "is_3ds": null,
  "is_live": false,
  "is_paid": false,
  "paid_at": null,
  "is_refunded": false,
  "installment_plan_id": null,
  "description": null,
  "metadata": {
      "customer_id": "42",
      "comment": "asked questions about size"
  },
  "notification": {
    "response_code": null,
    "url": "https://example.net/notifications?id=42"
  },
  "object": "payment",
  "save_card": false
}
```


Update Metadata for a payment that has already been created to update custom informations you have defined.

The request will respond an updated payment object if the ID provided is valid, and it will respond an error object otherwise.

PARAMETERS:

Key	Description
metadata *JSON object* ***optional***	Custom metadata object added to the request the object.

All metadata keys will be deleted during the update. To update only one key and keep the others you will need to send again all the keys and information you want to keep.

Capture a payment **Premium** :

Create a deferred payment:

```
<?php
$payment = \Payplug\Payment::create(array(
  'authorized_amount' => 3300,
  'currency'          => 'EUR',
  'save_card'         => false,
  'customer'          => array(
    'email'           => 'john.watson@example.net',
    'first_name'      => 'John',
    'last_name'       => 'Watson'
  ),
  'hosted_payment'    => array(
    'return_url'      => 'https://example.net/success?id=42',
    'cancel_url'      => 'https://example.net/cancel?id=42'
  ),
  'notification_url'  => 'https://example.net/notifications?id=42',
  'metadata'          => array(
    'customer_id'     => 42
  )
));

$payment_url = $payment->hosted_payment->payment_url;
$payment_id = $payment->id;
```

Capture the payment:

```
<?php
// Capture from a payment ID
$payment_capture = \Payplug\Payment::capture($payment_id);

// Capture from a payment object
$payment_capture = \Payplug\Payment::retrieve($payment_id);
$payment_capture = $payment_capture->capture();
```

Example response:

```
{
  "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "object": "payment",
  "is_live": true,
  "amount": 3300,
  "amount_refunded": 0,
  "authorization": {
    "authorized_at": 1555073502,
    "expires_at": 1555632000,
    "authorized_amount": 3300
  },
  "currency": "EUR",
  "created_at": 1434010787,
  "installment_plan_id": null,
  "is_paid": true,
  "paid_at": 1555073519,
  "is_refunded": false,
  "is_3ds": false,
  "save_card": false,
  "card": {
    "last4": "1800",
    "country": "FR",
    "exp_month": 9,
    "exp_year": 2017,
    "brand": "Mastercard",
    "id": null
  },
  "customer": {
    "first_name": "John",
    "last_name": "Watson",
    "email": "john.watson@example.net",
    "phone_number": null,
    "address1": null,
    "address2": null,
    "postcode": null,
    "city": null,
    "country": null,
    "language": "en"
  },
  "hosted_payment": {
    "payment_url": "https://www.payplug.com/pay/5iHMDxy4ABR4YBVW4UscIn",
    "return_url": "https://example.net/success?id=42",
    "cancel_url": "https://example.net/cancel?id=42",
    "paid_at": 1434010827,
    "sent_by": null
  },
  "notification": {
    "url": "https://example.net/notifications?id=42",
    "response_code": 200
  },
  "failure": null,
  "description": null,
  "metadata": {
    "customer_id": 42
  }
}
```


Capture a deferred payment that has been authorized.

The request will respond a payment object if the ID provided is valid, and it will respond an error object otherwise.

> The capture will fail if the authorization has expired. Monitor the field `authorization.expires_at` to know when the payment expires. A payment expires 7 calendar days after its authorization.

URL ARGUMENTS:

Key	Description
payment_id	ID of the payment to be captured.

PARAMETERS:

Key	Description
captured *boolean* ***required***	This parameter must be true .

Abort a payment :

Example:

```
 <?php
// Abort from a payment ID
$paymentId = 'pay_5iHMDxy4ABR4YBVW4UscIn';
$payment = \Payplug\Payment::abort($paymentId);

// Abort from a payment object
$paymentId = 'pay_5iHMDxy4ABR4YBVW4UscIn';
$payment = \Payplug\Payment::retrieve($paymentId);
$payment = $payment->abort();
```

Example response:

```
{
 "amount": 3300,
 "amount_refunded": 0,
 "authorization": null,
 "card": {
  "brand": null,
  "country": null,
  "exp_month": null,
  "exp_year": null,
  "id": null,
  "last4": null
 },
 "created_at": 1449157475,
 "currency": "EUR",
 "customer": {
  "address1": null,
  "address2": null,
  "city": null,
  "country": null,
  "email": "john.watson@example.net",
  "phone_number": null,
  "first_name": "John",
  "last_name": "Watson",
  "postcode": null,
  "language": "en"
 },
 "failure": {
  "code": "aborted",
  "message": "You have aborted the transaction."
 },
 "hosted_payment": {
  "cancel_url": "https://example.net/cancel?id=42",
  "paid_at": null,
  "payment_url": "https://www.payplug.com/pay/test/5iHMDxy4ABR4YBVW4UscIn",
  "return_url": "https://example.net/success?id=42",
  "sent_by": null
 },
 "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
 "is_3ds": null,
 "is_live": false,
 "is_paid": false,
 "paid_at": null,
 "is_refunded": false,
 "installment_plan_id": null,
 "description": null,
 "metadata": {
    "customer_id": "42"
 },
 "notification": {
  "response_code": null,
  "url": "https://example.net/notifications?id=42"
 },
 "object": "payment",
 "save_card": false
}
```


Abort a payment that has already been created to prevent its utilization.

The payment page will display a message explaining this payment is no longer available.

When a payment is aborted, the `failure` attribute will return: `aborted`.

The request will respond a payment object if the ID provided is valid, and it will respond an error object otherwise.

> Aborting a payment which is already paid or failed is not possible.

URL ARGUMENTS:

Key	Description
payment_id	ID of the payment to be aborted.

PARAMETERS:

Key	Description
aborted *boolean* ***required***	This parameter must be true .

List all payments :

Example:

```
<?php
$payments = \Payplug\Payment::listPayments();
$payment = $payments[0];
```

Example response:

```
{
  "object": "list",
  "page": 0,
  "per_page": 10,
  "has_more": true,
  "data": [{
    "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
    "object": "payment",
    "is_live": true,
    "amount": 3300
  }]
}
```

Retrieves a list of all the payment objects created for this authentication ID. All the payments are returned as a sorted list object (learn more in pagination section).

Requesting a payment list will only return payments created with the API. All payments created through the PayPlug portal will not be listed.

Save a card **Premium** :

Create a payment and save the card:

```
 <?php
$payment = \Payplug\Payment::create(array(
  'amount'           => 3300,
  'currency'         => 'EUR',
  'save_card'         => true,
  'customer'         => array(
    'email'          => 'john.watson@example.net',
    'first_name'     => 'John',
    'last_name'      => 'Watson'
  ),
  'hosted_payment'   => array(
    'return_url'     => 'https://www.example.net/success?id=42',
    'cancel_url'     => 'https://www.example.net/cancel?id=42'
  ),
  'notification_url' => 'https://example.net/notifications?id=42',
  'metadata'         => array(
    'customer_id'    => 42
  )
));
```

Example object once the payment is completed:

```
{
  "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "object": "payment",
  "is_live": true,
  "amount": 3300,
  "amount_refunded": 0,
  "authorization": null,
  "currency": "EUR",
  "created_at": 1434010787,
  "is_paid": true,
  "paid_at": 1555073519,
  "is_refunded": false,
  "is_3ds": true,
  "installment_plan_id": null,
  "save_card": true,
  "card": {
    "last4": "1800",
    "country": "FR",
    "exp_month": 9,
    "exp_year": 2017,
    "brand": "Mastercard",
    "id": "card_e7133426b8de947b37161dfba1897dd1"
  },
  "customer": {
    "first_name": "John",
    "last_name": "Watson",
    "email": "john.watson@example.net",
    "phone_number": null,
    "address1": null,
    "address2": null,
    "postcode": null,
    "city": null,
    "country": null,
    "language": "en"
  },
  "hosted_payment": {
    "payment_url": "https://www.payplug.com/pay/5iHMDxy4ABR4YBVW4UscIn",
    "return_url": "https://example.net/success?id=42",
    "cancel_url": "https://example.net/cancel?id=42",
    "paid_at": 1434010827,
    "sent_by": null
  },
  "notification": {
    "url": "https://example.net/notifications?id=42",
    "response_code": 200
  },
  "failure": null,
  "description": null,
  "metadata": {
    "customer_id": 42
  }
}
```


Create a payment with the card ID:

```
<?php
$payment = \Payplug\Payment::create(array(
  'amount'           => 1000,
  'currency'         => 'EUR',
  'payment_method'   => 'card_e7133426b8de947b37161dfba1897dd1',
  'customer'         => array(
    'email'          => 'john.watson@example.net',
    'first_name'     => 'John',
    'last_name'      => 'Watson'
  ),
  'notification_url' => 'https://example.net/notifications?id=42',
  'metadata'         => array(
    'customer_id'    => 42
));
```

Example response:


```
{
 "amount": 1000,
 "amount_refunded": 0,
 "authorization": null,
 "card": {
  "brand": null,
  "country": null,
  "exp_month": null,
  "exp_year": null,
  "id": "card_e7133426b8de947b37161dfba1897dd1",
  "last4": null
 },
 "created_at": 1449159160,
 "currency": "EUR",
 "customer": {
  "address1": null,
  "address2": null,
  "city": null,
  "country": null,
  "email": "john.watson@example.net",
  "phone_number": null,
  "first_name": "John",
  "last_name": "Watson",
  "postcode": null,
  "language": "en"
 },
 "failure": null,
 "hosted_payment": null,
 "id": "pay_5iHMDxy4ABR4YBVW4UscIn",
 "is_3ds": false,
 "is_live": false,
 "is_paid": true,
 "paid_at": 1555073519,
 "is_refunded": false,
 "installment_plan_id": null,
 "metadata": null,
 "description": null,
 "notification": {
  "response_code": null,
  "url": null
 },
 "object": "payment",
 "save_card": false
}
```


Save a card in order to create a payment later, using the same card, without using a payment page.

Saving a card is very simple, just use the property `save_card = true` or `allow_save_card = true` when you create a new payment. The response object will include a card ID (`id`) once the payment has been done (via notification or by retrieving the payment). The card ID will be functional for 15 months maximum.

This card id can then be used to create a new payment with the same card.

> In order to save a card, the first name, last name and the email address from the customer object need to be provided.

With `save_card` key, the customer is forced to save the card to process the payment. On the payment page the box is checked by default and the customer will not be able to validate the payment if he tries to uncheck the box.

Alternatively with `allow_save_card` key, saving the card is optional to process the payment. On the payment page the customer will be able to choose to save or not the card by checking the box.

The first payment using `save_card = true` and `allow_save_card = true` will automatically activate the 3-D Secure.

`save_card` and `allow_save_card` are not supposed to be used at the same time.

Refunds

The refunds object enables you to fully or partially refund a payment that has been paid. Refunds will be applied to the credit card originally used for the payment.

REFUNDS ENDPOINT REFERENCE:

Method	Endpoint	Usage
POST	/v1/payments/{payment_id}/refunds	Create a refund
GET	/v1/payments/{payment_id}/refunds/{refund_id}	Retrieve a refund
POST	/v1/payments/{payment_id}/refunds	List all refunds

The refund object :

Example:

```
<?php
// Print the amount of the refund
echo htmlentities($refund->amount);
// Print the refund creation time in an human readable way
echo htmlentities(date('d/m/Y H:i:s', $refund->created_at));
```

```
{ "amount": 358,
  "created_at": 1434012358,
  "currency": "EUR",
  "id": "re_3NxGqPfSGMHQgLSZH0Mv3B",
  "is_live": true,
  "metadata": {
    "customer_id": 42,
    "reason": "The delivery was delayed"
  },
  "object": "refund",
  "payment_id": "pay_5iHMDxy4ABR4YBVW4UscIn"
}
```

OBJECT ATTRIBUTES:

Key	Description
id *string*	Refund ID
payment_id *string*	ID of the payment refunded.
object *string*	Value is: refund .
is_live *boolean*	true for a payment in LIVE mode, false in TEST mode.
amount *positive integer*	Amount of the refund in cents and must be highier than 0.10€ (10 cents).
currency *currency*	Currency code three-letter ISO 4217 in which the refund was made.
created_at *timestamp*	Date of creation of the refund.
metadata *JSON object*	Custom metadata object added to the request the object.

Create a refund :

Example:

```
 <?php
// Create a refund from a payment id
$paymentId = 'pay_5iHMDxy4ABR4YBVW4UscIn';
$data = array(
  'amount'   => 358,
  'metadata' => array(
      'customer_id' => 42,
      'reason' => 'The delivery was delayed'
    )
  );
$refund = \Payplug\Refund::create($paymentId, $data);

// Also refund a payment object directly
$payment = \Payplug\Payment::retrieve('pay_5iHMDxy4ABR4YBVW4UscIn');
$data = array('amount' => 358);
$refund = $payment->refund($data);

// Or more simply for a total refund
$payment = \Payplug\Payment::retrieve('pay_5iHMDxy4ABR4YBVW4UscIn');
$refund = $payment->refund();
```

Example response:

```
{
  "id": "re_3NxGqPfSGMHQgLSZH0Mv3B",
  "payment_id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "object": "refund",
  "is_live": true,
  "amount": 358,
  "currency": "EUR",
  "created_at": 1434012358,
  "metadata": {
    "customer_id": 42,
    "reason": "The delivery was delayed"
  }
}
```

To refund a payment, you can either refund the total amount without setting up any parameters, or partially refund it by providing the amount wanted. The refund will be applied to the credit card originally used for the payment. You can refund a payment several times, as long as the payment has not been fully refunded.

When a payment is entirely refunded, it will impossible to refund it again. If you try to refund a payment already entirely refunded you will get an error message.

PARAMETERS:

Key	Description
amount *positive integer* ***optional***	Amount in cents, for instance use **308** for a **3.08€** refund. If not provided the payment will be fully refunded.
metadata *JSON object* ***optional***	Custom metadata object added to the request the object.

The `amount_refunded` attribute in the payment object indicates how much has been refunded.

Retrieve a refund :

Example:

```
<?php
// Retrieve a refund object from its id and its payment id
$paymentId  = 'pay_5iHMDxy4ABR4YBVW4UscIn';
$refundId   = 're_3NxGqPfSGMHQgLSZH0Mv3B';
$refund = \Payplug\Refund::retrieve($paymentId, $refundId);
```

Example response:

```
{
  "id": "re_3NxGqPfSGMHQgLSZH0Mv3B",
  "payment_id": "pay_5iHMDxy4ABR4YBVW4UscIn",
  "object": "refund",
  "is_live": true,
  "amount": 358,
  "currency": "EUR",
  "created_at": 1434012358,
  "metadata": {
    "customer_id": 42,
    "reason": "The delivery was delayed"
  }
}
```

Retrieve details about a refund for a given payment.

URL ARGUMENTS:

Key	Description
payment_id	ID of the payment to be retrieved.
refund_id	ID of the refund to be retrieved.

List all refunds :

Example:

```
<?php
// Retrieve a list of refunds from the payment id.
$paymentId = 'pay_5iHMDxy4ABR4YBVW4UscIn';
$refunds = \Payplug\Refund::listRefunds($paymentId);
$refund = $refunds[0];

// Retrieve a list of refunds from the payment object.
$payment = \Payplug\Payment::retrieve('pay_5iHMDxy4ABR4YBVW4UscIn');
$refunds = $payment->listRefunds();
$refund = $refunds[0];
```

Example response

```
{
  "object": "list",
  "page": 0,
  "per_page": 10,
  "has_more": false,
  "data": [{
    "id": "re_3NxGqPfSGMHQgLSZH0Mv3B",
    "payment_id": "pay_5iHMDxy4ABR4YBVW4UscIn",
    "object": "refund",
    "is_live": true,
    "amount": 358,
    "currency": "EUR",
    "created_at": 1434012358,
    "metadata": {
      "customer_id": 42,
      "reason": "The delivery was delayed"
    }
  }]
}
```


Retrieve a list of all the refund objects for a given payment. All the refunds are returned is a sorted list object (learn more in pagination section).

URL ARGUMENTS:

Key	Description
payment_id	ID of the payment to be retrieved.

Installment plans **Premium**

Use the /installment_plans endpoint to create an installment plan, abort an installment plan or retrieve an installment plan.

INSTALLMENT PLANS ENDPOINT REFERENCE:

Method	Endpoint	Usage
POST	/v1/installment_plans	Create an installment plan
GET	/v1/installment_plans/{installment_plan_id}	Retrieve an installment plan
PATCH	/v1/installment_plans/{installment_plan_id}	Abort an installment plan

The installment plan object :

Example:

```
{
  "id": "inst_1FTGSRWYHla7eDkfTo2Usd",
  "object": "installment_plan",
  "is_live": true,
  "is_active": true,
  "currency": "EUR",
  "created_at": 1548326773,
  "is_fully_paid": false,
  "customer": {
    "first_name": "John",
    "last_name": "Watson",
    "email": "john.watson@example.net",
    "address1": null,
    "address2": null,
    "postcode": null,
    "city": null,
    "country": null
  },
  "hosted_payment": {
    "payment_url": "https://secure.payplug.com/pay/1FTGSRWYHla7eDkfTo2Usd",
    "return_url": "https://example.net/success?id=42",
    "cancel_url": "https://example.net/cancel?id=42"
  },
  "notification": {
    "url": "https://example.net/notifications?id=42"
  },
  "schedule": [
    {
      "date":"2019-01-24",
      "amount": 15000,
      "payment_ids": ["pay_62VazeAbq5ttYietdC2QxR"]
    },
      "date":"2019-02-24",
      "amount": 15000,
      "payment_ids": ["pay_12uazeAbq5ttYietdC2QxP"]
    },
      "date":"2019-03-24",
      "amount": 10000,
      "payment_ids": []
    },
  ],
  "failure": null,
  "metadata": {
    "customer_id": 42
  }
}
```


```php
 <?php
// Get an attribute of the installment plan object:
// Print the installment plan is_fully_paid value
echo htmlentities($installment_plan->is_fully_paid);
// Print the $installment_plan id
echo htmlentities($installment_plan->id);
// Print the $installment_plan creation time in an human readable way
echo htmlentities(date('d/m/Y H:i:s', $installment_plan->created_at));

// Chain relations between objects:
// Print the first payment date
echo htmlentities($installment_plan->schedule[0]->date);
// Print the customer email
echo htmlentities($installment_plan->customer->email);

// Retrieve metadata
echo htmlentities($installment_plan->metadata['customer_id']);
echo htmlentities($installment_plan->metadata['type']);
```

OBJECT ATTRIBUTES:

Key	Description
id *string*	Installment plan ID.
object *string*	Value is: installment_plan .
is_live *boolean*	true for an installment_plan in LIVE mode, false in TEST mode.
is_active *boolean*	true if the installment_plan is active, false otherwise.
currency *currency*	Currency code (three-letter ISO 4217) in which the installment_plan was made.
created_at *timestamp*	Creation date.
is_fully_paid *boolean*	true if the installment_plan is fully paid (all installments are paid), false otherwise.
customer *JSON object*	Information about the customer.
	first_name: Customer first name. *string*
	last_name: Customer last name. *string*
	email: Customer email address. *string*
	address1: Customer address line 1 (Street address/PO Box/Company name). *string*
	address2: Customer address line 2 (Apartment/Suite/Unit/Building). *string*
	postcode: Customer Zip/Postal code. *string*
	city: Customer city. *string*
	country: Customer country code (two-letter ISO 3166). *string*
hosted_payment *JSON object*	Data related to the first installment.
	payment_url: Payment URL where you should redirect your customer to. *string*
	return_url: URL where the customer will be redirected after the payment page whether it succeeds or not. *string*
	cancel_url: URL where the customer will be redirected after a click on "**Cancel Payment**". *string*
notification *JSON object*	Data related to notifications.
	url: URL that will be used by PayPlug to send notifications. *string*
schedule *Array of JSON object*	List of scheduled installments.
	date: Installment date. *string*
	amount: Amount of the payment in cents. *integer*
	payment_ids: List of payments ids. *array*
failure *JSON object or null*	Information for unsuccessful installment plans.
	code: Installment plan failure code if the installment plan has failed. *string*
	message: Descriptive message explaining the reason of the installment plan failure. *string*
metadata *JSON object*	Custom metadata object added when creating the installment plan.

The **response_code** can be very useful for debugging because it gives you important information when you do not receive the notifications.

Create an installment plan :

Example:

```
 <?php
$payment = \Payplug\InstallmentPlan::create(array(
  'currency'         => 'EUR',
  'schedule'         => array(
    array('date' => 'TODAY', 'amount' => 1000),
    array('date' => '2019-03-02', 'amount' => 2050),
    array('date' => '2019-04-02', 'amount' => 1026)
  ),
  'customer'         => array(
    'email'          => 'john.watson@example.net',
    'first_name'     => 'John',
    'last_name'      => 'Watson'
  ),
  'hosted_payment'   => array(
    'return_url'     => 'https://example.net/success?id=42',
    'cancel_url'     => 'https://example.net/cancel?id=42'
  ),
  'notification_url' => 'https://example.net/notifications?id=42',
  'metadata'         => array(
    'customer_id'    => 42
  )
));
```

Example response:

```
{
  "id": "inst_1FTGSRWYHla7eDkfTo2Usd",
  "object": "installment_plan",
  "is_live": true,
  "is_active": true,
  "currency": "EUR",
  "created_at": 1548326773,
  "is_fully_paid": false,
  "customer": {
    "first_name": "John",
    "last_name": "Watson",
    "email": "john.watson@example.net",
    "address1": null,
    "address2": null,
    "postcode": null,
    "city": null,
    "country": null
  },
  "hosted_payment": {
    "payment_url": "https://secure.payplug.com/pay/1FTGSRWYHla7eDkfTo2Usd",
    "return_url": "https://example.net/success?id=42",
    "cancel_url": "https://example.net/cancel?id=42"
  },
  "notification": {
    "url": "https://example.net/notifications?id=42"
  },
  "schedule": [
    {
      "date":"2019-02-02",
      "amount": 1000,
      "payment_ids": ["pay_12VazeAbq5ttYietdC2QxR"]
    },
      "date":"2019-03-02",
      "amount": 2050,
      "payment_ids": ["pay_22uazeAbq5ttYietdC2QxP"]
    },
      "date":"2019-04-02",
      "amount": 1026,
      "payment_ids": []
    },
  ],
  "failure": null,
  "metadata": {
    "customer_id": 42
  }
}
```


Allow customers to spread out payments over 2, 3 or 4 installments.

PARAMETERS:

currency *currency*	Currency code (three-letter ISO 4217) in which the installment_plan was made.
schedule *Array of JSON object*	List of scheduled installments.
	date: Installment date, set TODAY for the first payment. *string*
	amount: Amount of the payment in cents. *integer*
customer *JSON object*	Information about the customer.
	first_name: Customer first name. *string*
	last_name: Customer last name. *string*
	email: Customer email address. *string*
	address1: Customer address line 1 (Street address/PO Box/Company name). *string*

	address2: Customer address line 2 (Apartment/Suite/Unit/Building). *string*
	postcode: Customer Zip/Postal code. *string*
	city: Customer city. *string*
	country: Customer country code (two-letter ISO 3166). *string*
hosted_payment *JSON object*	Data related to the first installment.
	return_url: URL where the customer will be redirected after the payment page whether it succeeds or not (for the first installment only). *string*
	cancel_url: URL where the customer will redirected after a click on “**Cancel Payment**”. *string*
notification_url *string* ***optional***	URL that will be used by PayPlug to send notifications.
metadata *JSON object* ***optional***	Custom metadata object.

Retrieve an installment plan :

Example:

```
 <?php
$installmentPlan = \Payplug\InstallmentPlan::retrieve('inst_2JhnPxy4ABR4YBVW4UscWx');
```

Example response:

```
{
 "id": "inst_2JhnPxy4ABR4YBVW4UscWx",
 "object": "installment_plan",
 "is_live": true,
 "is_active": true,
 "currency": "EUR",
 "created_at": 1548326773,
 "is_fully_paid": false,
 "customer": {
  "first_name": "John",
  "last_name": "Watson",
  "email": "john.watson@example.net",
  "address1": null,
  "address2": null,
  "postcode": null,
  "city": null,
  "country": null
 },
 "hosted_payment": {
  "payment_url": "https://secure.payplug.com/pay/2JhnPxy4ABR4YBVW4UscWx",
  "return_url": "https://example.net/success?id=42",
  "cancel_url": "https://example.net/cancel?id=42"
 },
 "notification": {
  "url": "https://example.net/notifications?id=42"
 },
 "schedule": [
  {
   "date":"2019-02-02",
   "amount": 1000,
   "payment_ids": ["pay_12VazeAbq5ttYietdC2QxR"]
  },
   "date":"2019-03-02",
   "amount": 2050,
   "payment_ids": ["pay_22uazeAbq5ttYietdC2QxP"]
  },
   "date":"2019-04-02",
   "amount": 1026,
   "payment_ids": []
  },
 ],
 "failure": null,
 "metadata": {
  "customer_id": 42
 }
}
```


Retrieves the information of a specific installment plan that has already been created.

The request will return an installment_plan object if the ID provided is valid, and it will return an error object otherwise.

URL ARGUMENTS:

Key	Description
installment_plan_id	ID of the installment_plan to be retrieved.

Abort an installment plan :

Example:

```php
<?php
// Abort from an installment plan ID
$installmentPlanId = 'inst_2JhnPxy4ABR4YBVW4UscWx';
$installmentPlan = \Payplug\InstallmentPlan::abort($installmentPlanId);

// Abort from an installment_plan object
$installmentPlanId = 'inst_2JhnPxy4ABR4YBVW4UscWx';
$installmentPlan = \Payplug\InstallmentPlan::retrieve($installmentPlanId);
$installmentPlan = $installment_plan->abort();
```

Example response:


```
{
  "id": "inst_2JhnPxy4ABR4YBVW4UscWx",
  "object": "installment_plan",
  "is_live": true,
  "is_active": false,
  "currency": "EUR",
  "created_at": 1548326773,
  "is_fully_paid": false,
  "customer": {
    "first_name": "John",
    "last_name": "Watson",
    "email": "john.watson@example.net",
    "address1": null,
    "address2": null,
    "postcode": null,
    "city": null,
    "country": null
  },
  "hosted_payment": {
    "payment_url": "https://secure.payplug.com/pay/2JhnPxy4ABR4YBVW4UscWx",
    "return_url": "https://example.net/success?id=42",
    "cancel_url": "https://example.net/cancel?id=42"
  },
  "notification": {
    "url": "https://example.net/notifications?id=42"
  },
  "schedule": [
    {
      "date":"2019-02-02",
      "amount": 1000,
      "payment_ids": ["pay_12VazeAbq5ttYietdC2QxR"]
    },
      "date":"2019-03-02",
      "amount": 2050,
      "payment_ids": ["pay_22uazeAbq5ttYietdC2QxP"]
    },
      "date":"2019-04-02",
      "amount": 1026,
      "payment_ids": []
    },
  ],
  "failure": {
    "code": "aborted",
    "message": "You have aborted the transaction."
  },
  "metadata": {
    "customer_id": 42
  }
}
```


Abort an installment plan to prevent future installments to be created.

When an installment plan is aborted, its `failure code` will be set to `aborted`, and `is_active` will be set to `false`.

The request will respond an installment_plan object if the ID provided is valid, and it will respond an error object otherwise.

Aborting an installment plan which is already inactive or fully paid is not possible.

URL ARGUMENTS:

Key	Description
installment_plan_id	ID of the installment_plan to be aborted.

PARAMETERS:

Key	Description
aborted *boolean* ***required***	This parameter must be true .