EveryPay Integration Documentation

2018-4-23

Table of Contents

Quick References **4**

Test environment endpoints 4

Production environment endpoints 4

Helper libraries 4

Test Cards 5

1 Payment Integration **6**

1.1 Payment Process Overview 6

1.2 Initiating Payment 7

1.2.1. Charge 7

1.2.2. Tokenisation 8

0-amount tokenisation 8

Get token with payment 8

1.2.3. One-Off Payment 9

1.2.4. One-Click (token) Payment 10

1.3 Payment Form Types 11

1.3 API URL and Authentication 12

1.4 Important factors 12

1.6 Redirect Payment Page 14

1.6.1 Redirect Customer Back to Merchant 14

1.6.2 Payment Cancellation 15

1.7 Embedded Payment Form (iFrame) 16

1.8 Payment With Saved Card 19

1.8.1 0-amount tokenisation 20

1.8.2 Get token with first payment 20

1.8.3 Payment with token 21

1.9 Payment Parameters description: 21

1.10 Callback to Merchant 24

2 Backend Integration **26**

2.1 Principles and Capabilities 26

2.1.1 REST 26

2.1.2 Supported Formats 27

2.1.3 Security 27
2.1.4 Access to the API 27
2.1.5 HTTP Response Codes 27
2.1.6 Overview of the Resources 29
2.2 Charge (Token Payment Initiation) 30
2.3 Payment Managment 32
2.3.1 Void 33
2.3.2 Capture 34
2.3.3 Refund 36
3 Mobile App Payments 37
3.1 Mobile App Payments Workflow 37
3.2 Authentication Data Request 39
3.3 Request Encrypted Token 40
3.4 Request 3DS form 42
3.4.1 3DS Result URL (Redirect URL) 43
3.4.2 Encrypted Payment Instrument 3DS Confirmed 43
3.5 Decrypt Encrypted Token 44
Appendix 1 Integration Testing 46
AP 1.1 Testing checklist 46
Changelog 47

Quick References

Test environment endpoints

Application	endpoint URL	usage
Gateway API	https://igw-demo.every-pay.com/transactions/	for HTTP POST form action
Backend API	https://gw-demo.every-pay.com/	
Merchant Portal	https://mwt-demo.every-pay.com/merchant_settings/general	Access API username and secret and track payment data. Note: API username and secret are different in Production and Test environment.

Production environment endpoints

Application	endpoint URL	usage
Gateway API	https://pay.every-pay.eu/transactions/	for HTTP POST form action
Backend API	https://gw.every-pay.eu/	
Merchant Portal	https://portal.every-pay.eu/merchant_settings/general	Access API username and secret and track payment data. Note: API username and secret are different in Production and Test environment.

Helper libraries

EveryPay provides the following helper libraries to simplify the integration:

- PHP library – to simplify the entire integration process (initiating the payment API requests and processing the callback/redirect responses), including HMAC calculation
- JavaScript library – to simplify iFrame payment page integration

The helper libraries are available in our Github repository:
https://github.com/UnifiedPaymentSolutions/everypay-integration

Test Cards

Please note that only test cards must be used for testing. The following test cards can be used to perform successful test payments:

Card type	Card number	Expiration date	CVC code	Cardholder name
Mastercard	5168830763287080	10/21	949	(any name)
Visa	4761739001010010	09/25	179	(any name)
Mastercard	2223000010021381	12/19	656	(any name)

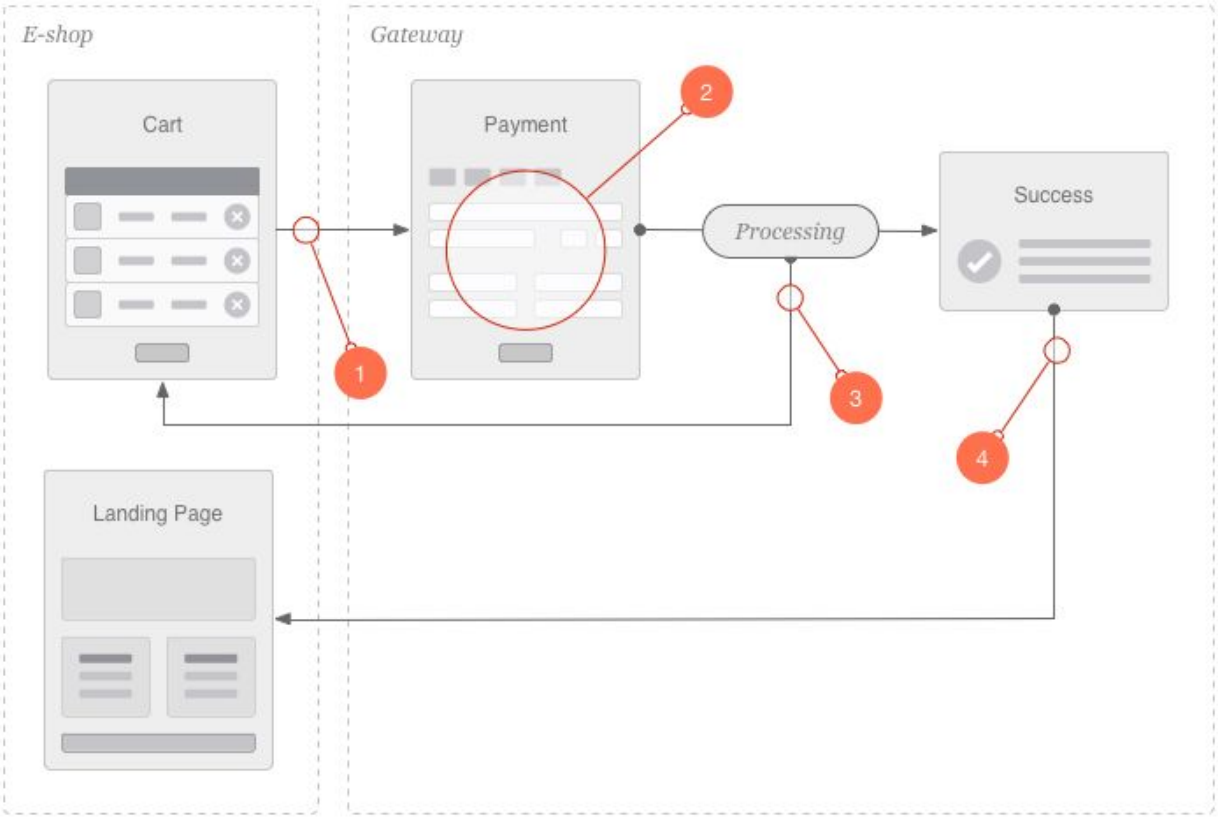
The 3DS authentication simulator (Poseidon bank) password is `secret`.

To test failed payments, the easiest ways are to enter incorrect expiration date or incorrect 3DS password.

1 Payment Integration

1.1 Payment Process Overview

The process of collecting a payment is displayed on the diagram below:



On high level the steps in the payment collection are as follows:

1. Payment initiation API request is submitted from the merchant's server to EveryPay server.
2. EveryPay Payment Page is displayed to the buyer in one of the following ways:
    a. Embedded on merchant's website (using HTML iFrame)
    b. Buyer is redirected to EveryPay payment page
3. Once the buyer has provided the card details (and performed 3D-Secure bank authentication when required), EveryPay server will process the payment (which includes communication with all required external systems). On Token payments (One-Click payments), instead of entering card details, customer clicks "Pay" button and Payment info will be sent to EveryPay for processing.
4. Once the payment is processed, depending on the payment page solution:
    a. Redirect payment page -- the buyer will be redirected back to the merchant's website

b. iFrame payment form -- the buyer will see the successful/failed message inside the iFrame

After completing this process, asynchronous automatic callback notification with the transaction status and details is sent from EveryPay server back to the merchant's server. Based on this the merchant server will update the order status depending on the payment result.

1.2 Initiating Payment

To initiate a payment with EveryPay the following **transaction types** can be used:

- Charge
- Tokenisation
    - Tokenisation with charge

By using aforenamed transaction types different payments can be performed as follows.

Payment type	Description	Comment
One-off	Standard payment initiated by cardholder	Transaction_type = charge
Checkup & save card	0-amount authorisation request to verify card validity in order to save the card data for future payments	Transaction_type = tokenisation
One-off & save card	Standard payment initiated by cardholder with a token request to be returned in a callback in order to save the card data for future payments	Transaction_type = charge Request_cc_token = 1
One-click	Payment with saved card data	Transaction_type = charge

1.2.1. Charge

A **charge** consists of authorisation and automatic capture. If the authorisation is completed successfully, the transaction will be automatically captured and included in the next capture file

1. Authorisation is done when the cardholder initiates the payment. It's a process for getting approval to the transaction and booking the required amount on the card (without actually transferring the funds yet).
2. Capture is performed automatically at 0:30 for transactions that were performed on the previous date between 0:00 and 23:59 UTC+2 (or UTC+3 during daylight saving time).
3. The funds are transferred to the merchant's bank account about 12 hours after the capture.

Capture delay functionality can be used if needed to delay the automatic capture of the authorisation for up to 8 days. This provides extra time buffer before fulfilling the customer's order and charging the money from their bank account, which might be required for certain business models, for example:

- to perform additional background checks on the customer (to prevent possible fraud)
- to verify the availability of goods/services

1.2.2. Tokenisation

To facilitate the payment process for buyer, it's possible to save the card token on merchant server and allow future payments to be made using the card token, without the need for the buyer to enter the full set of card details in the payment form. The token refers to the unique combination of card number and expiration date, stored securely in EveryPay server.

For added security it's possible to ask the buyer to enter also CVC code and/or perform 3DS authentication.

The token is only usable under a single merchant and can't be converted back to card number and expiration date outside EveryPay servers, eliminating the risk of card fraud.

IMPORTANT! The merchant is responsible for complying with any regulations that might be involved with storing the card token. According to the information available to EveryPay, there's a European Union regulation that requires the merchant to ask for their customer's permission before storing the card token for future payments. The permission must be asked as opt-in (not opt-out) -- the customer must deliberately express his/her consent with this, e.g. by checking a checkbox (the checkbox must not be checked by default). The merchant is responsible for retaining a copy of that permission.

There are two ways to obtain the card token - either via 0-amount tokenisation or in response to the first payment.

0-amount tokenisation

0-amount tokenisation (a.k.a checkup payment) means performing a 0-amount authorisation with the card to verify its validity. The token will be returned in the callback message for successful authorisations. The request is almost identical with one-off payment request. The only difference is 'transaction_type' parameter value and it does not expect shipping or billing address data.

Field name	Description
transaction_type	Value 'tokenisation'

Get token with payment

To request the token to be returned in the callback message, request_cc_token parameter

must be included in the one-off payment request. The token will be returned in the callback message for successful authorisations.

Field name	Description
request_cc_token	Accepted values are: ● '1' - cc_token will be returned in callback ● '0' (default) - cc_token will not be returned in callback

Callback parameters

If the payment was processed successfully, the callback message includes the card token in cc_token parameter. Together with the last four digits of the card number and other details, the token can be easily stored and referred to on merchant side.

Field name	Description
cc_token	cc_token will be returned only if it was requested in the Payment Initiation.

1.2.3. One-Off Payment

EveryPay Payment API expects the API request in HTTP POST format.

Description of the optional field markings:
O - field is optional
OF - field is optional, but including it improves fraud detection (therefore it's highly recommended to provide this information whenever possible)

Parameters

Field name	Optional
api_username	
account_id	
nonce	
timestamp	
callback_url	
customer_url	
email	OF
amount	
order_reference	
user_ip	OF

billing_address	OF
billing_country	OF
billing_city	OF
billing_postcode	OF
delivery_address	O
delivery_country	O
delivery_city	O
delivery_postcode	O
hmac	
hmac_fields	
transaction_type	
locale	O
request_cc_token	O
cc_token	O
token_security	O
skin_name	O

HTML form example

```
<form action="https://igw-demo.every-pay.com/transactions/" method="post">
 <input name="hmac" value="75ed21e06d7e3ed26d1eb8b3fab24bdf3d73df20">
 <input name="hmac_fields"
value="account_id,amount,api_username,billing_address,billing_city,billing_country,billin
g_postcode,callback_url,customer_url,delivery_address,delivery_city,delivery_country,deli
very_postcode,email,hmac_fields,nonce,order_reference,timestamp,transaction_type,user_ip"
>
 <input name="transaction_type" value="charge">
 <input name="locale" value="en">
 <input name="amount" value="10.0">
 <input name="api_username" value="1adcffbf5e2df582">
 <input name="account_id" value="EUR1">
 <input name="billing_address" value="Tamme 2">
 <input name="billing_city" value="Tallinn">
 <input name="billing_country" value="EE">
 <input name="billing_postcode" value="12345">
 <input name="callback_url" value="http://www.google.ee/?q=callback">
 <input name="customer_url" value="http://www.google.ee/?q=redirect">
 <input name="delivery_address" value="Kuuse 3">
 <input name="delivery_city" value="Tallinn">
 <input name="delivery_country" value="EE">
 <input name="delivery_postcode" value="12345">
 <input name="email" value="email@example.org">
 <input name="nonce" value="d6e33441e0171249e71992946896f754">
 <input name="order_reference" value="98c9fa2e52f0679610935497ff4da714">
 <input name="timestamp" value="1397038382">
 <input name="user_ip" value="82.131.119.82">
 <input class="btn btn-danger" type="submit" value="Proceed to Payment">
</form>
```

1.2.4. One-Click (token) Payment

When the API request contains the optional cc_token parameter with a valid token value, the payment is processed as token payment. Optionally also token_security parameter can be included.
Token payments are also possible via Backend integration. With token payment via Backend, no information is displayed to customer and token_security options are unavailable.

Field name	Description
cc_token	Enables payment with stored card token. Can be used in two ways: 1) separately - regular token payment without any user input (using saved card number and expiration date) 2) together with token_security field (to use added payer/card authentication measures)
token_security	To be used only together with the cc_token parameter. Enables token payments (with saved card data) with added payer/card authentication measures. Possible values are: ● 'none' - regular token payment without any user input (using saved card number and expiration date) ● 'cvc' - payer must provide CVC code ● '3ds' - payer must perform 3DS authentication ● 'cvc_3ds' - payer must provide CVC code and perform 3DS authentication

1.3 Payment Form Types

There are two different types of payment forms to choose from:

1) Redirect payment page
After completing the checkout process on merchant's website the customer will be redirected to EveryPay redirect payment page. If the payment process is completed the customer will be then redirected back to the merchant's website.

The benefits:
- Simpler integration and less changes required on merchant's website.

The downside:
- Redirecting the customer away from the merchant's website hinders the user experience.
- The redirect payment page uses EveryPay visual design which doesn't match the merchant's website design. However, by using merchant's own logo on the payment page can improve the visual of the page so it is still recognizable for the customers.

2) Embedded payment form (HTML iFrame)
The payment form will be embedded directly into the merchant's website checkout page.

The benefits:

- Customer will not be redirected away from the merchant's website.
- The payment form visual design is customizable in EveryPay Merchant Portal and can be configured to closely match the visual design of the merchant's website.

The downside:

- Slightly more complex integration and more changes required on merchant's website
- If the merchant's website is running on HTTP, not seeing the HTTPS encryption sign in web browser can scare off some customers, even though the traffic inside the iFrame is using HTTPS.
- 3D secure authentication is by default redirect. It means that after filling in card details and clicking on PAY button, the bank's 3DS page is displayed to the customers as a redirect page, not inside iFrame. After the 3DS authentication is completed the customer is redirected back to EveryPay page and from there to the customer_url URL which is sent to with the initial API request. Reliability is the most important benefit of this solution in order to eliminate the issues with 3DS implementations of some banks.

Which one should you use?
For most merchants the embedded payment form is the recommended option as it helps to improve user experience considerably compared to the redirect payment page. However, if the merchant's website uses HTTP, it might be better to use the redirect payment page as seeing the HTTPS encryption sign (while entering their credit card details) can provide the extra confidence required to prevent them from turning away with fear of security risks.

1.3 API URL and Authentication

All new implementations must go through appropriate testing process to ensure the proper payment processing.

Gateway API endpoint URLs (for HTTP POST form action) in EveryPay environments:

- **Test**: https://igw-demo.every-pay.com/transactions/
- **Production**: https://pay.every-pay.eu/transactions/

API username and secret are different in Production and Test environment. They can be found in corresponding Merchant Portal:

- **Test**: https://mwt-demo.every-pay.com/merchant_settings/general
- **Production**: https://portal.every-pay.eu/merchant_settings/general

1.4 Important factors

nonce

All EveryPay message responses and requests contain the 'nonce' field, that can be used to verify uniqueness of the response message. This approach helps to prevent possible message replay attacks.

Validating the uniqueness of the returned ‘nonce’ value on merchant's side will add an extra layer of security. It requires storing ‘nonce’ values of the received messages and comparing the ‘nonce’ values of the received messages against the stored ‘nonce’ values.

order_reference

The main purpose of `order_reference` parameter is to match the payment in EveryPay’s system to the correct corresponding order in merchant’s e-shop. By default the `order_reference` value is required to be unique. As an additional benefit, matching the `order_reference` and validating its uniqueness in merchant’s e-shop provides an extra layer of security against tampering attacks. However, if needed the `order_reference` uniqueness validation can be turned off. When disabled, multiple payment attempts are allowed for one order reference until a successful payment is performed.

HMAC

HMAC (a keyed-hash message authentication code) is used to verify API request:

1. data integrity
2. authentication

EveryPay uses SHA-1 lower hexadecimal HMAC format.

HMAC is constructed by concatenating form parameter key and value pairs into a string ordered by alphabetic order of the key name and returning its hexdigest with shared secret.
NB! Do not use any quotes or escaping, but use values as-is in the string:

```
key1=value&key2=value with space&key3=...
```

HMAC calculation can be tricky and it’s important to pay extra attention to which API parameters and how should be included in HMAC calculations. For example:

Where listed in allowed parameter list, `hmac_fields` parameter is used to assist with providing required information about which fields are included in HMAC calculation.

- If not told otherwise (for a specific API parameter, e.g. locale) all API request parameters must be included in HMAC calculation.
  HMAC string contents must not include code escapes -- neither for payment initiation API request or the callback/redirect response (e.g. the JSON contents of `processing_errors` and `processing_warnings`).

The most common mistakes with HMAC calculation are:

- not including all required parameters and their values in calculation
- adding new parameters after HMAC calculation
- using code escapes
- parameters not ordered alphabetically

HMAC string example:

```
account_id=EUR3D1&amount=10&api_username=abcd1234abcd1234&callback_url=http://www.google.
ee/?q=callback&customer_url=http://www.google.ee/?q=redirect&hmac_fields=account_id,amoun
t,api_username,billing_address,billing_city,billing_country,billing_postcode,callback_url
```

```
,customer_url,delivery_address,delivery_city,delivery_country,delivery_postcode,email,hma
c_fields,nonce,order_reference,timestamp,transaction_type,user_ip&nonce=d6e33441e0171249e
71992946896f754&order_reference=98c9fa2e52f0679610935497ff4da714&timestamp=1397038382&tra
nsaction_type=charge&user_ip=82.131.119.82
```

Code example in Ruby to calculate HMAC:

```
# Calculate hmac_fields of all fields that will be in HMAC (adjust as needed)
hmac_fields = (data.keys + ["hmac_fields"]).sort

# Not all requests require use of hmac_fields, add only when needed
data["hmac_fields"] = hmac_fields.join(",")

hmac_string = hmac_fields.map{|k| "#{k}=#{data[v]}"}.join("&")
hmac = OpenSSL::HMAC.hexdigest("sha1", "shared_api_secret", hmac_string)
```

Timestamp

Timestamp represents the time of the request. The request will be rejected if the provided timestamp is outside of an allowed time-window.

Timestamp should be calculated from the time in the UTC timezone.

The value of the timestamp is an integer number of seconds since the Unix Epoch (January 1, 1970).

1.6 Redirect Payment Page

1.6.1 Redirect Customer Back to Merchant

Buyer initiates checkout on merchant's web. For payment he/she will be redirected to EveryPay payment page.
Once the buyer has completed the payment process, he/she will be redirected back to the merchant's website -- to the URL specified in customer_url field. The redirect includes HTTP PUT (faked over POST with _method="put" hidden param) message with payment result and other related information.

HTTP PUT(faked over POST with _method="put" hidden param) Parameters

Field name	Optional Field (marked as "O")
api_username	
account_id	
nonce	
timestamp	
amount	

order_reference	
payment_reference	
payment_state	
hmac	
hmac_fields	
transaction_result	
cc_token	O
cc_last_four_digits	
cc_month	
cc_year	
cc_holder_name	
cc_type	

It's important to notice that in case of using the redirect payment page, the merchant server will receive two messages about the payment:

- asynchronous server-to-server callback (HTTP POST) from EveryPay server to merchant server which contains full details about the payment
- synchronous client-to-client redirect message (HTTP PUT (faked over POST with _method="put" hidden param)) from EveryPay payment page to merchant's shop (only relevant for redirect, not iFrame payment page solution) which contains a subset of the callback message details (as we don't want the buyer to see some of the payment details which are only meant for merchant's use)

The contents of the two messages overlap partially. As browser redirect usually takes more time the callback message is usually delivered before the redirect message. However, it can also happen the other way around. If the additional parameters in callback message will be used for something, the order records on merchant side should be updated accordingly if the callback message is delivered later than the redirect message.

1.6.2 Payment Cancellation

When using redirect payment solution, it's possible to cancel the payment by clicking BACK button on the redirect payment page. After clicking the button, the buyer will be redirected back to the merchant's website (to the URL specified in API request customer_url parameter).

Parameters

The redirect includes HTTP PUT (faked over POST with _method="put" hidden param) request with the following parameters:

Field name	Description

nonce	Random unique value to prevent replay attacks. Validating the uniqueness of the returned 'nonce' value on merchant's side will add an extra layer of security.
timestamp	Seconds from January 1, 1970 UTC
order_reference	Order reference
payment_state	Current status of the payment which reflects the definitive payment outcome based on which the order status should be updated in merchant system. In case of payment cancellation the value is ‘cancelled’.
hmac	Calculated HMAC for request authenticity verification. Use only the fields listed in hmac_fields for HMAC calculation, all the other fields should be ignored.
hmac_fields	It is a string which contains all fields (keys) that are going to be used in hmac calculation, separated by comma. “hmac_fields” contains a key “hmac_fields” and does not contain a key “hmac”. E.g: "api_username,hmac_fields,nonce,order_reference,payment_state,timestamp,transaction_result"
transaction_result	In case of a cancellation the value is ‘cancelled’.
api_username	Merchant API username

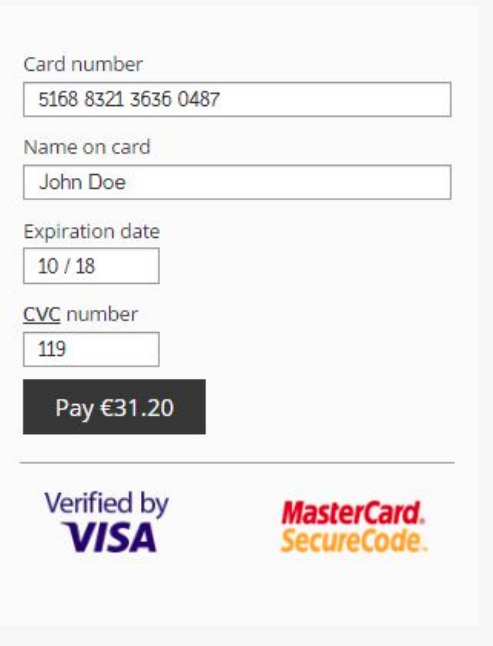
1.7 Embedded Payment Form (iFrame)

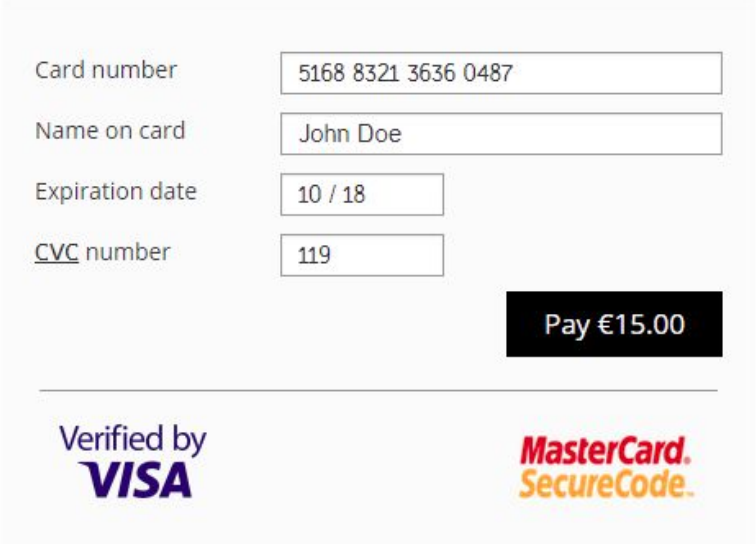
If you don’t want buyers to redirect away from your site during the checkout process, use the embedded payment form solution by integrating EveryPay iFrame payment form in payment flow. The design of the iFrame payment form contents can be fully customised to match the shop’s look-and-feel.

The customization can be done in Merchant Portal “iFrame skins” section under Settings:

- **Test**: https://mwt-demo.every-pay.com/merchant_settings/skins
- **Production:** https://portal.every-pay.eu/merchant_settings/skins

The contents of the iFrame are responsive and displayed according to the iFrame size. When iFrame width is below 400px, the field labels are displayed above (not in front of) the fields. This allows for implementing responsive design for the iFrame size in shop.





With iFrame payment page (unlike with redirect payment page) the buyer is not automatically redirected to customer_url after completing the payment process. The iFrame will contain the successful or unsuccessful payment page. Redirection inside iFrame is not done because otherwise merchant website would be displayed inside the iFrame. To redirect the buyer to a different page after completing the payment process, the merchant website can rely on the iFrame-to-parent post messages which contain information about payment result. However, when redirect 3DS is used (by default) then after the 3DS authentication is completed the customer is redirected back to EveryPay page and from there to the customer_url URL which is sent to with the initial API request.

EveryPay provides the required tools for customizing the design of the payment form inside the iFrame and ensures that the functional aspects work properly. The merchant should make sure that the iFrame itself fits nicely with the shop layout and design.

Parameters

If parameter skin_name is included in the API request, the payment form will be returned with iFrame compatible design.

Field name	Description
skin_name	Use ‘default’ if you want to use default skin design.

iFrame-to-parent communication

Using iFrames in a cross-domain environment has certain security restrictions. There are few scenarios where these security restrictions set significant limitations to web developers and make it difficult to more achieve decent end-user-experience. Because of this, EveryPay has developed a JavaScript helper library to assist merchants with the integration: https://github.com/UnifiedPaymentSolutions/everypay-integration/tree/master/javascript

The JavaScript has two main purposes:

1. receiving payment status messages from iFrame
2. resizing 3D-Secure authentication page

For 3D-Secure, default setting is redirect. By this loading merchants page inside iframe after 3D Authentication is avoided.

Payment status message

Receiving a message about the payment status (success or failure) directly from the iFrame, right after the buyer has completed the payment, allows the merchant website to display relevant message to the buyer immediately after the payment process inside the iFrame has been completed. Otherwise it would be necessary to “poll” for the arrival of the callback message from EveryPay server to merchant server.

Due to standard cross-domain iFrame-to-parent security restrictions the JavaScript includes required code to overcome these restrictions. The iFrame sends postMessage to the domain specified in the customer_url API request parameter.

The message from iFrame to parent includes two types of fields:

1. field ‘transaction_result’ (‘completed’ or ‘failed’, depending on the result of the payment attempt)
2. success/error messages (contains the same texts that the buyer would see inside the iFrame)

Some message fields are not always present.

Field name	Optional	Description
transaction_result		Possible values are ‘completed’ and ‘failed’
message_title		E.g. "Thank you! Payment successful." or "Sorry, payment was unsuccessful."
message_error	O	Information about the cause of payment failure, if the exact reason is known and can be revealed to the buyer.
message_action	O	Suggestions about what the buyer should do to perform a successful payment.

message_contact	O	Instructions about who the customer should contact if the problem persists -- the merchant or the card issuing bank.

When a payment is initiated with a card token (without the need for buyer to enter card details), thanks to these messages, it's not even always necessary to display the full iFrame itself as no user interface is required -- the iFrame can be included on the page as a hidden element and will only be necessary to receive the (error) messages to display to the buyer using the shop's visual design styles (font, color etc).

Resizing 3D-Secure page

MasterCard, Visa and other card schemes have established certain standards to banks about the size of the 3D-Secure authentication page -- roughly 400x400px. However, unfortunately not all banks follow this standard. Therefore it's better to automatically resize the iFrame to ensure that even bigger 3D-Secure pages will fit nicely without the need for scrolling.

iFrame implementation example

```
<iframe id="iframe-payment-container" name="iframe-payment-container", width="400",
height="400" sandbox="allow-top-navigation allow-forms allow-popups allow-scripts
allow-same-origin"></iframe>


<form action="https://igw-demo.every-pay.com/transactions" id="iframe_form" method="post"
style="display: none" target="iframe-payment-container">
 <input name="hmac" value="75ed21e06d7e3ed26d1eb8b3fab24bdf3d73df20">
 <input name="hmac_fields"
value="account_id,amount,api_username,callback_url,customer_url,nonce,order_reference,ski
n_name,timestamp,transaction_type,user_ip">
 <input name="transaction_type" value="charge">
 <input name="locale" value="en">
 <input name="amount" value="1.0">
 <input name="api_username" value="b3616e26a91d3cb4">
 <input name="account_id" value="EUR1">
 <input name="callback_url" value="http://www.google.ee/?q=callback">
 <input name="customer_url" value="http://www.google.ee/?q=redirect">
 <input name="nonce" value="30d7810d31dbb77d4300fd3f6a59ff11">
 <input name="order_reference" value="98c9fa2e52f0679610935497ff4da714">
 <input name="timestamp" value="1437488204">
 <input name="user_ip" value="82.131.119.82">
 <input name="skin_name" value="default">
</form>
<script>
 window.onload = function() { document.getElementById("iframe_form").submit(); }
</script>
```

1.8 Payment With Saved Card

To facilitate the payment process for buyer, it's possible to save the card token on merchant server and allow future payments to be made automatically using the card token, without the need for the buyer to enter the full set of card details in the payment form. The token refers to the unique combination of card number and expiration date, stored securely in EveryPay server.

For added security it's possible to ask the buyer to enter also CVC code and/or perform 3DS

authentication.

The token is only usable under a single merchant and can't be converted back to card number and expiration date outside EveryPay servers, eliminating the risk of card fraud.

IMPORTANT! The merchant is responsible for complying with any regulations that might be involved with storing the card token. According to the information available to EveryPay, there's a European Union regulation that requires the merchant to ask for their customer's permission before storing the card token for future payments. The permission must be asked as opt-in (not opt-out) -- the customer must deliberately express his/her consent with this, e.g. by checking a checkbox (the checkbox must not be checked by default). The merchant is responsible for retaining a copy of that permission.

There are two ways to obtain the card token - either via 0-amount tokenisation or in response to the first payment.

1.8.1 0-amount tokenisation

0-amount tokenisation (a.k.a checkup payment) means performing a 0-amount authorisation with the card to verify its validity. The token will be returned in the callback message for successful authorisations. The request is almost identical with one-off payment request. The only difference is 'transaction_type' parameter value.

Field name	Description
transaction_type	Value 'tokenisation'

1.8.2 Get token with first payment

To request the token to be returned in the callback message, request_cc_token parameter must be included in the one-off payment request. The token will be returned in the callback message for successful authorisations.

Field name	Description
request_cc_token	Accepted values are: ● '1' - cc_token will be returned in callback ● '0' (default) - cc_token will not be returned in callback

Callback parameters

If the payment was processed successfully, the callback message includes the card token in cc_token parameter. Together with the last four digits of the card number and other details, the token can be easily stored and referred to on merchant side.

Field name	Description

cc_token	cc_token will be returned only if it was requested in the Payment Initiation.

1.8.3 Payment with token

When the API request contains the optional cc_token parameter with a valid token value, the payment is processed as token payment. Optionally also token_security parameter can be included.
Token payments are also possible via Backend integration. With token payment via Backend, no information is displayed to customer and token_security options are unavailable.

Field name	Description
cc_token	Enables payment with stored card token. Can be used in two ways: 1) separately - regular token payment without any user input (using saved card number and expiration date) 2) together with token_security field (to use added payer/card authentication measures)
token_security	To be used only together with the cc_token parameter. Enables token payments (with saved card data) with added payer/card authentication measures. Possible values are: ● 'none' - regular token payment without any user input (using saved card number and expiration date) ● 'cvc' - payer must provide CVC code ● '3ds' - payer must perform 3DS authentication ● 'cvc_3ds' - payer must provide CVC code and perform 3DS authentication

1.9 Payment Parameters description:

Parameters

Field name	Description
api_username	Merchant API username. The value can be found in Merchant Portal in the Settings section.
account_id	Processing account to be used for the transaction. Processing account defines the transaction processing configuration, including the currency to be used, pricelist, 3D Secure settings, payment recurrence, clearing settings, etc.

	Example values could be: ‘EUR3D1, ‘USD3D2’. The field value can be found in Merchant Portal in the Settings section and they are Case Sensitive
nonce	Random unique value to prevent replay attacks
timestamp	Time of creating the transaction. Expressed as seconds from January 1, 1970 UTC
callback_url	Once EveryPay gateway has processed the transaction, processing result data is posted to this URL.
customer_url	When the buyer clicks on the Back button, he will be redirected to this URL.
email	E-mail address (buyer)
amount	Payment amount
order_reference	Order reference, must be unique for every payment attempt
user_ip	IP-address (buyer)
billing_address	Billing address
billing_country	Billing country, in two-character ISO 3166 format
billing_city	Billing city
billing_postcode	Billing postcode
delivery_address	Delivery address
delivery_country	Delivery country, in two-character ISO 3166 format
delivery_city	Delivery city
delivery_postcode	Delivery postcode
hmac	HMAC is constructed by concatenating form parameter keys and values into a string ordered by alphabetic order of the key name and returning its hexdigest with shared secret. Shared secret key can be viewed and changed in Merchant Portal in the Settings section.
hmac_fields	It is a string which contains all fields (keys) that are going to be used in hmac calculation, separated by comma. “hmac_fields” should contain the key “hmac_fields” and should not contain the key

	“hmac”. E.g: "account_id,amount,api_username,callback_url,customer_url,hmac_fields,nonce,order_reference,timestamp,transaction_type,user_ip"
transaction_type	“charge” or “tokenization”
locale	Sets the locale and the language of the EveryPay Payment Page user interface displayed to the customer. Defaults to ‘en’. Accepted values are: ● ‘en’ - English ● ‘et’ - Estonian ● ‘fi’ - Finnish ● ‘de’ - German ● ‘lv’ - Latvian ● ‘lt’ - Lithuanian ● ‘ru’ - Russian ● ‘es’ - Spanish ● ‘sv’ - Swedish ● ‘da’ - Danish ● ‘pl’ - Polish Please contact the Everypay support if you need other supported locales. ***Note:*** This field must not be included for HMAC calculation.
request_cc_token	Accepted values are: ● ‘1’ - cc_token will be returned in callback ● ‘0’ (default) - cc_token will not be returned in callback
cc_token	Enables payment with stored card token. If a valid cc_token is provided, the buyer will not be asked to fill in card details. Can be used in two ways: 1. separately - regular token payment without any user input (using saved card number and expiration date) 2. together with token_security field (to use added payer/card authentication measures)
token_security	To be used only together with the cc_token parameter. Enables token payments (with saved card data) with added payer/card authentication measures. Possible values are:

	• ‘none’ - regular token payment without any user input (using saved card number and expiration date) • ‘cvc’ - payer must provide CVC code • ‘3ds’ - payer must perform 3DS authentication • ‘cvc_3ds’ - payer must provide CVC code and perform 3DS authentication
skin_name	Skin name for the embedded payment form (HTML iFrame). This parameter should only be used if the required changes are made on merchant’s website to embed EveryPay iFrame in the checkout process.

1.10 Callback to Merchant

Once the payment attempt is completed (successfully or not), EveryPay system will send a callback message (HTTP form POST) to the URL defined in the payment initiation request callback_url field. If the first attempt fails (i.e. URL doesn’t return HTTP 2xx or 3xx response code), EveryPay server will attempt to resend the callback several times until it succeeds or fails permanently - maximum 6 retries will be done with the following logic

```
RETRY_INTERVALS = [
        1.second,
        5.minutes,
        1.hour,
        24.hours,
        48.hours,
        72.hours,
    ].freeze
```

When Merchant checks hmac on callback, fields used for calculating the hmac must be taken dynamically, so any added field would not break system and give wrong hmac as a result.

Parameters

Field name	Optional	Description
api_username		Merchant API username
account_id		Processing account ID that was used to process the transaction
nonce		Random unique value to prevent replay attacks. Validating the uniqueness of the returned 'nonce' value on merchant's side will add an extra layer of security.
timestamp		Seconds from January 1, 1970 UTC
amount		Transaction amount, dot as decimal separator

order_reference		Order reference
payment_reference		Payment reference ID
payment_state		Current status of the payment which reflects the definitive payment outcome based on which the order status should be updated in merchant system. Possible scenarios: ● 'settled' or 'authorised' (successful payments) ● 'cancelled' or 'waiting_for_3ds_response' (cancelled/incomplete payments) ● 'failed' (failed payments)
processing_errors		Array of error hashes in JSON format. ***Note:*** Don't use code escapes for HMAC calculation.
processing_warnings		Hash of fraud check warnings in JSON format. ***Note:*** Don't use code escapes for HMAC calculation.
hmac		Calculated HMAC for request authenticity verification over the fields in the response message. Use only the fields listed in hmac_fields for HMAC calculation, all the other fields should be ignored. The HMAC must be generated based on the exact returned fields, i.e. without code escapes for processing_errors and processing_warnings fields.
hmac_fields		It is a string which contains all fields (keys) that are going to be used in hmac calculation, separated by comma. "hmac_fields" contains a key "hmac_fields" and does not contain a key "hmac". E.g: "account_id,amount,api_username,hmac_fields,nonce,order_reference,payment_reference,payment_state,timestamp,transaction_result,user_ip"
transaction_result		Should only be considered as additional information about different phases of the payment. Possible values ● 'completed' ● 'cancelled' ● 'failed' The definitive outcome of the payment attempt is reflected by **payment_state** parameter based on which the order status should be updated in merchant system.
cc_token	O	cc_token will be returned only if it was requested in the Payment Initiation.
cc_last_four_digits		Last four digits of the card number.

cc_month		Card expiration month (as entered by the cardholder in payment form), in mm format (e.g. 1 or 12)
cc_year		Card expiration year (as entered by the cardholder in payment form), in YYYY format (e.g. 2017)
cc_holder_name		Cardholder name (as entered by the cardholder in payment form)
cc_type		Card type. Possible values are ‘visa’ and ‘master_card’.
cc_issuer_country		Card issuing bank country, in alpha-2 format (e.g. ‘EE’).
cc_issuer		Card issuing organization
stan	O	Payment STAN reference number, in numeric format. Returned only in case the payment was sent for processing to card processor and didn’t fail already on EveryPay level.
state_3ds		Payment 3D-Secure authentication status. Possible values are: ● ‘3ds’ - authentication completed successfully ● ‘attempted’ - authentication attempted but not completed ● ‘failed’ - authentication failed ● ‘no3ds’ - no 3DS coverage for payment ● ‘unknown’ - 3DS status unknown
fraud_score		Payment fraud score, calculated by EveryPay fraud prevention engine, in numeric format.

2 Backend Integration

2.1 Principles and Capabilities

2.1.1 REST

The API is implemented in the REST architectural style:

- payments and transactions are exposed as resources
- operations on resources are performed using standard HTTP methods (GET, POST, etc)
- each request must specify a media type for the resource presentation format
- error conditions on operations are expressed as HTTP response codes

2.1.2 Supported Formats

GW supports communication in JSON format.

API format must specified in the request HTTP header as follows:

- `Content-Type: application/json`
- `Accept: application/json`

All key values are handled as strings and therefore double quotes are mandatory.

JSON request example:

```
{
   "charge": {
    "api_username":"12345678",
    "account_id":"EUR1",
    "amount":"10.00",
    "order_reference":"912987",
    "cc_token":"XXXXXXXXXXXXXXXXXX",
    "device_info":"{"android_id":"1565be46e4fab518"}",
    "user_ip":"10.10.10.10",
    "email":"john.random@example.com",
    "nonce":"a9b7f7e794367c2c85d73154a01b9902",
    "timestamp":1427933807,
    "hmac":"41c60a44a64f4fd7d6622bc40148fef9bcd1ecae",
    }
}
```

2.1.3 Security

All connections to the GW must be made over SSL connection. Clients must verify the validity of the GW certificates, to avoid man-in-the-middle attacks.

HMAC calculation is identical to what is described above in HMAC section, with the exception that `hmac_fields` parameter is not used.

2.1.4 Access to the API

The API can be accessed via the following URLs:

- Production environment:
  https://gw.every-pay.eu/ (example: https://gw.every-pay.eu/charges)
- Test environment:
  https://gw-demo.every-pay.com/ (example: https://gw-demo.every-pay.com/charges)

2.1.5 HTTP Response Codes

Code	Status	Description
200	Success	Standard response for successful queries
400	Bad Request	Returned if HTTP operation was not understood or was incorrectly formatted
401	Unauthorised	Returned if processing the request is refused because of failed authentication, including incorrect HMAC, nonce or timestamp.
403	Forbidden	Returned if processing the request is refused
422	Unprocessable Entity	Returned if processing the request was not successful for any reason, including processing errors such as validation, fraud check or issuer declines.
500	Internal Server Error	Returned if request cannot be processed because of the technical errors in the server

2.1.6 Overview of the Resources

GW consists of the following resources:

URI	Method	Description
/authorisations	POST	Creates a new payment with authorisation
/voids	POST	Voids an authorised payment
/captures	POST	Captures an authorised payment
/charges	POST	Creates a new payment with automated capture

/refunds	POST	Creates a new refund of a settled payment
/payment_instruments	POST	Creates bank card tokens usable for later payments
/encrypted_payment_instruments	POST	Creates encrypted bank card tokens
/encrypted_payment_instrument_decryptions	POST	Decrypts an encrypted bank card token
/authentication3ds/new	GET	

2.2 Charge (Token Payment Initiation)

Charge transaction initiates a payment, books a requested amount on the cardholder's bank card and settles automatically within a day.

Charge can be done by providing a bank card token.

Request Parameters

URI: /charges
Request body key: charge

Parameter	Optional	Description
api_username		Merchant's API username. Please see General Settings section of the Merchant Portal for exact values.
account_id		Processing account used to process the payment. Please see Processing Account section of the Merchant Portal for exact values.
amount		Transaction amount, expressed in the same currency as defined in the Processing Account used to process the transaction.
order_reference		Merchant's order ID. Needs to be unique for each request.
cc_token		Card token.
nonce		Unique request identificator (see below for details).
timestamp		Timestamp of request's creation time (see below for details).
hmac		HMAC value of the request (see below for calculation details).

Request Body Example

```
{
 "charge": {
   "api_username":"12345678",
   "account_id":"EUR1",
   "amount":"10.00",
   "order_reference":"912987",
   "nonce":"a9b7f7e794367c2c85d73154a01b9902",
   "timestamp":1427933807,
   "hmac":"41c60a44a64f4fd7d6622bc40148fef9bcd1ecae",
   }
}
```

Response Parameters

Parameter	Description
account_id	Processing Account ID that was used to process the transaction
amount	Amount used for the transaction.
order_reference	Merchant's order ID
email	Cardholder's e-mail address
cc_token	Token referencing a bank card, that can later be used to initiate recurring payments. It is returned only if the token was requested with request_cc_token.
cc_last_four_digits	Last four digits of the card number
cc_month	Card expiration month (mm format - 1-2 digits)
cc_year	Card expiration year (YYYY format - 4 digits)
cc_holder_name	Name on card
cc_type	Card type. Possible values are 'visa' or 'master_card'.
cc_issuer_country	Card issuer country. ISO 3166 two-letter (alpha-2) format (e.g. EE)
cc_issuer	Card issuing organization
stan	Payment STAN number - a unique ID to identify payments on acquiring bank payment reports.
state_3ds	Payment 3D-Secure status. Possible values are '3ds', 'attempted', 'failed' or 'no3ds'
fraud_score	Payment fraud score
warnings	Payment processing warnings in JSON format.
user_ip	Cardholder's IP address
transaction_time	Time of the transaction
transaction_reference	Reference ID of the transaction
transaction_result	Result of the transaction. Possible values are: 'completed' or 'failed'
payment_reference	Reference ID of the payment

payment_state	Current status of the payment

Successful Response Example


```
{
 "charge": {
   "account_id": "EUR1",
   "amount": "1",
   "order_reference": "feiwhp28qy8ks7i12i63",
   "email": null,
   "cc_token": "d841bcc672b0f76523a7fa13",
   "cc_last_four_digits": "1234",
   "cc_month": "1",
   "cc_year": "2017",
   "cc_holder_name": "Tom Smith",
   "cc_type": "master_card",
   "cc_issuer_country": "EE",
   "Cc_issuer": "LHV Bank"
   "stan": "1234",
   "state_3ds": "no3ds",
   "fraud_score": "500",
   "warnings": {
     "country_match": [
       "Card issuer country (Estonia) does not match to the buyer country ()."
     ],
   },
   "user_ip": "127.0.0.1",
   "transaction_time": "2015-11-29T12:22:31Z",
   "transaction_reference": "5938005aa2de7d3f2b51b4c39e7cf03acab12d859c032b2a5a9294",
   "transaction_result": "completed",
   "payment_reference": "db98561ec7a380d2e0872a34ffccdd0c4d2f2fd237b6d0ac22f88f52a",
   "payment_state": "settled"
 }
}
```


Unsuccessful Response Example


```
{
 "errors": [
   {
     "code": 3016,
     "message": "Suspicion of manipulation; Fraud suspected; Suspected counterfeit card"
   }
 ]
}
```


2.3 Payment Managment

2.3.1 Void

Void transaction reverses an authorised payment that has not been set to be captured.

Request Parameters

URI: /voids

Request body key: void

Parameter	Description
api_username	Merchant's API username. Please see General Settings section of the Merchant Portal for exact values
payment_reference	Reference to a payment that will be voided
user_ip	Customer IP (omit if not known)
nonce	Unique request identificator (see below for details)
timestamp	Timestamp of request's creation time (see below for details)
hmac	HMAC value of the request (see below for calculation details)

Request Body Example

```
{
 "void": {
   "api_username":"12345678",
   "payment_reference":"51f40de9c243935dbd6b2d1f187668af9a03dd06b9ff0b992dc61e",
   "user_ip":"10.10.10.10",
   "nonce":"1b49e7e3d0f794a8a3332b4a9ef90cca",
   "timestamp":1427961187,
   "hmac":"29c05b7179dbd95f2f578c91b3d515519177209a"
 }
}
```

Response Parameters

Parameter	Description
transaction_time	Time of the transaction
transaction_reference	Reference ID of the transaction
transaction_result	Result of the transaction. Possible values are: 'completed', 'failed'
payment_reference	Reference ID of the payment created by the completed transaction
payment_state	Current state of the payment
user_ip	Merchant server IP

Successful Response Example

```
{
 "void": {
```

```
   "transaction_time":"2015-04-02T07:53:07Z",
   "transaction_reference":"45c5978feaab1a00dfe50045a8bee0c8c3ca1e1fbecb49b0144",
   "transaction_result":"completed",
   "payment_reference":"51f40de9c243935dbd6b2d00dc544b741f187668af90b992dc61e",
   "payment_state":"voided",
   "user_ip":"10.10.10.10"
 }
}
```

Unsuccessful Response Example

```
{
 "errors": [
   {
     "code":4016,
     "message":"Can not void non-authorized payment"
   }
 ]
}
```

2.3.2 Capture

Capturing an authorised payment will send it for processing.

Request Parameters

URI: /captures
Request body key: capture

Parameter	Description
amount	Transaction amount
api_username	Merchant's API username. Please see General Settings section of the Merchant Portal for exact values.
payment_reference	Reference to a payment that will be voided.
user_ip	Customer IP (omit if not known)
nonce	Unique request identificator (see below for details).
timestamp	Timestamp of request's creation time (see below for details).
hmac	HMAC value of the request (see below for calculation details).

Request Body Example

```
{
  "capture":{
     "amount":"1",
     "api_username":"12345678",
```

```
    "payment_reference":"61b27d2846021a4ef814349a2b7c4c27af4473be53400c00cbc03",
    "nonce":"rd8cyyu7epqc8ejgfugx",
    "timestamp":"1448805170",
    "user_ip":"127.0.0.1",
    "hmac":"890fbb365af21526bfb960284e8c661fb92ee1cf"
  }
}
```

Response Parameters

Parameter	Description
amount	Transaction amount
transaction_time	Time of the transaction
transaction_reference	Reference ID of the transaction
transaction_result	Result of the transaction. Possible values are: ‘completed’, ‘failed’
payment_reference	Reference ID of the payment created by the completed transaction
payment_state	Current state of the payment
user_ip	Cardholder’s IP address

Successful Response Example

```
{
 "capture": {
   "amount": "1",
   "user_ip": "127.0.0.1",
   "transaction_time": "2015-11-29T13:53:41Z",
   "transaction_reference": "9af901b445e73e9ca9e181ca199c81dcd78ca7c8d1e881e0f2f2fc",
   "transaction_result": "completed",
   "payment_reference": "61b27d2846021a4ef814349a28add17c4c27af4473be53400c00cbc03",
   "payment_state": "settled"
 }
}
```

Unsuccessful Response Example

```
{
 "errors": [
   {
     "code": 4010,
     "message": "Can not capture non-authorized payment"
   }
 ]
}
```

2.3.3 Refund

URI: /refunds
Request body key: refund

Refunding a settled payment will return the funds to the customer's bank card. A payment can be refunded in full or partial amount.

Request Parameters

Parameter	Description
amount	Transaction amount
api_username	Merchant's API username. Please see General Settings section of the Merchant Portal for exact values.
payment_reference	Reference to a payment that will be voided.
user_ip	Customer IP (omit if not known)
nonce	Unique request identificator (see below for details).
timestamp	Timestamp of request's creation time (see below for details).
hmac	HMAC value of the request (See 1.4 for details).

Request Body Example

```
{
  "refund":{
     "amount":"1",
     "api_username":"12345678",
     "payment_reference":"61b27d2846021a4ef814349a2b7c4c27af4473be53400c00cbc03",
     "nonce":"rd8cyyu7epqc8ejgfugx",
     "timestamp":"1448805170",
     "user_ip":"127.0.0.1",
     "hmac":"890fbb365af21526bfb960284e8c661fb92ee1cf"
  }
}
```

Response Parameters

Parameter	Description
amount	Transaction amount
transaction_time	Time of the transaction
transaction_reference	Reference ID of the transaction
transaction_result	Result of the transaction. Possible values are: 'completed', 'failed'

payment_reference	Reference ID of the payment created by the completed transaction
payment_state	Current state of the payment
user_ip	Cardholder's IP address

Successful Response Example

```
{
 "refund": {
   "amount": "1",
   "user_ip": "127.0.0.1",
   "transaction_time": "2015-11-29T13:53:41Z",
   "transaction_reference": "9af901b445e73e9ca9e181ca199c81dcd78ca7c8d1e881e0f2f2fc",
   "transaction_result": "completed",
   "payment_reference": "61b27d2846021a4ef814349a28add17c4c27af4473be53400c00cbc03",
   "payment_state": "settled"
 }
}
```

Unsuccessful Response Example

```
{
 "errors": [
   {
     "code": 4013,
     "message": "Amount exceeds current payment amount"
   },
 ]
}
```

3 Mobile App Payments

3.1 Mobile App Payments Workflow

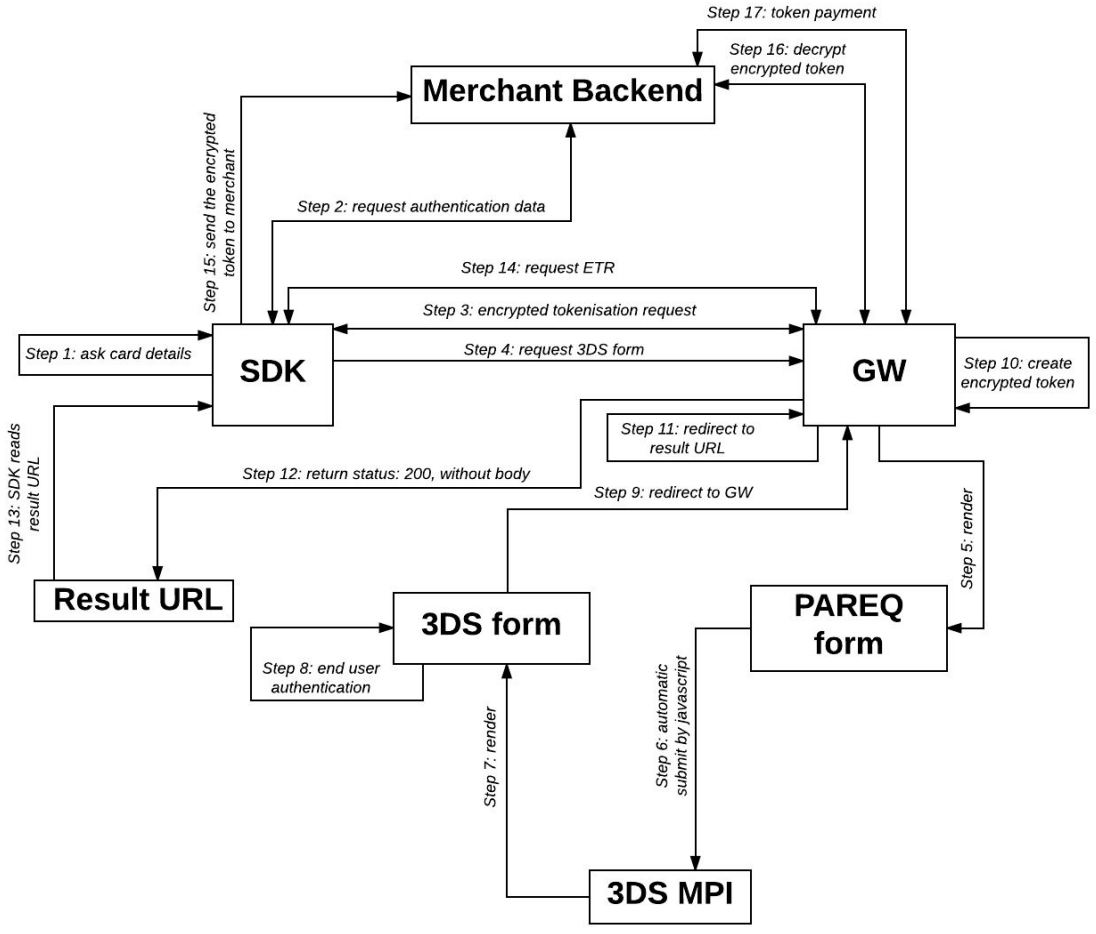
To facilitate the integration of EveryPay payment solution to mobile apps, we have developed SDKs for iOS and Android platforms:

- iOS: https://github.com/UnifiedPaymentSolutions/everypay-ios
- Android: https://github.com/UnifiedPaymentSolutions/everypay-android

The workflow for Android/iOS mobile app payments is the following:

- **Step 1.** Cardholder provides credit card details to SDK.
- **Step 2.** SDK requests authentication data from Merchant Backend
- **Step 3.** SDK combines credit card details from *Step 1*, authentication data from *Step 2*, and mobile device info (optional) in order to make an encrypted tokenisation request to GW, which returns:
    - In case of regular payments, encrypted_cc_token and payment_state

    - In case of 3DS payments, secure_code_one, payment_reference, and payment_state
- **Step 4 (only 3DS payments).** Request 3DS form from GW (mobile_3ds_hmac, secure_code_one and payment_reference should be sent as GET parameters)
- **Step 5 (only 3DS payments).** GW renders PAREQ form
- **Step 6 (only 3DS payments).** PAREQ form is automatically submitted by JavaScript to 3DS MPI
- **Step 7 (only 3DS payments).** 3DS MPI renders 3DS form
- **Step 8 (only 3DS payments).** Cardholder performs authentication on 3DS form
- **Step 9 (only 3DS payments).** User is redirected back to GW
- **Step 10 (only 3DS payments).** Encrypted token is generated
- **Step 11-12 (only 3DS payments).** Redirect to result URL
- **Step 13 (only 3DS payments).** SDK detects redirect and retrieves GET parameters from result URL
- **Step 14 (only 3DS payments).** SDK requests encrypted token from GW using GET parameters from step 13
- **Step 15.** SDK sends encrypted token to Merchant Backend
- **Step 16 (customisable on Merchant Backend).** Merchant decrypts encrypted token
- **Step 17 (customisable on Merchant Backend).** Merchant makes Token Payment



Workflow of mobile payments using SDK.

N.B.:

- the card details (number, name, expiration date and cvc) can not be signed by HMAC (Step 3) because:
    - the API secret key must not be stored in the mobile app because of security considerations.
    - the card details can not be sent to the merchant backend, unless the merchant is PCI DSS compliant
- nonce, timestamp and hmac values must be unique for every API call.
- account_id parameter is necessary to make 0-amount authorisation (to validate the card details).

3.2 Authentication Data Request

Authentication Data Request should be made to Merchant Backend, which should return as a response a list of parameters signed with HMAC.

Request Parameters

IGW-shop test server (Merchant server ending in every-pay.com) **:**
https://igwshop-demo.every-pay.com

Method: POST
URI: /merchant_mobile_payments/generate_token_api_parameters

Response (should be provided by Merchant Backend)

```
{
 "account_id": "EUR1",
 "user_ip": "10.100.1.100",
 "api_username": "95d7342hcf35ua88",
 "nonce": "ca331a24341295115ad3fa379edd99f7",
 "timestamp": 1465225649,
 "hmac": "59e6fd83e057eb530efa7d0d086fffdafb2018d5",
 "order_reference": "123f4517d0d086fffd234t5"
}
```

3.3 Request Encrypted Token

Request Parameters

URI: /encrypted_payment_instruments

Request body key: encrypted_payment_instrument

HMAC calculation excludes CC data and device_info parameters.
Fields included in HMAC calculation are marked as X in HMAC column

Parameter	Optional	HMAC	Description
account_id		x	Processing account used to process the payment. Please see Processing Account section of the Merchant Portal for exact values.
api_username		x	Merchant's API username. Please see General Settings section of the Merchant Portal for exact values.
cc_number		-	Customer's bank card number.
cc_year		-	Bank card expiration year.
cc_month		-	Bank card expiration month.
cc_holder_name		-	Cardholder name.
cc_verification		-	Bank card security code (CVC/CVV).
device_info	O	-	A string containing a JSON hash describing the relevant device specs.
nonce		x	Unique request identificator (see below for details).
timestamp		x	Timestamp of request's creation time (see below for details).
user_ip		x	Cardholder's IP address
hmac		-	HMAC value of the request (see below for calculation details).
order_reference	O in v1	x	Order reference (must be unique)

Request Body Example

```
{
```

```
 "encrypted_payment_instrument" {
   "account_id":"EUR1",
   "api_username":"12345678",
   "nonce":"a9b7f7e794367c2c85d73154a01b9902",
   "timestamp":1427933807,
   "hmac":"41c60a44a64f4fd7d6622bc40148fef9bcd1ecae",
   "cc_number":"4111111111111111",
   "cc_year":"2019",
   "cc_month":"12",
   "cc_holder_name":"John Random",
   "cc_verification":"123",
   "user_ip":"127.0.0.1",
   "Device_info":"{"android_id":"1565be46e4fab518”}",
   "order_reference":"d6622bc40148fe",
 }
}
```

Successful Non-3DS Response Example

```
{
 "encrypted_payment_instrument":{
   "cc_token_encrypted":"XpgZ/UTsNhBHJ6LJTGsGRQ==-1439355600",
   "payment_state":"authorised"
 }
}
```

Successful 3DS Response Example

```
{
 "encrypted_payment_instrument": {
   "payment_reference":"0aa6409492f358da0fb6d9b821ce6ca4a5609073489dcaf3456023cafca96efa"
,
   "payment_state":"waiting_for_3ds_response",
   "secure_code_one":"XyIfP0b7giwcJma24axOaQt2m96F/ThG62Ptd5rsX4Bj7tSAM/pfgD"
 }
}
```

Unsuccessful Response Example

```
{
 "errors": [
   {
     "code":2002,
     "message":"The hmac '1234' is invalid"
   }
 ]
}
```

3.4 Request 3DS form

Request Parameters

URI: /authentication3ds/new
Method: GET

GET Parameter	Description
payment_reference	Payment reference. Can be taken from /encrypted_payment_instruments response
secure_code_one	Secure code one. Can be taken from /encrypted_payment_instruments response
mobile_3ds_hmac	HMAC, which was originally used, as a parameter in /encrypted_payment_instruments request

3.4.1 3DS Result URL (Redirect URL)

URI: /authentication3ds/[PAYMENT REFERENCE]
Method: GET

GET Parameter	Description
payment_state	Can have values: • authorised (in case of success) • failed (in case of failure)

3.4.2 Encrypted Payment Instrument 3DS Confirmed

URI: /encrypted_payment_instruments/[PAYMENT REFERENCE]

GET Parameter	Description
payment_reference	It should not be passed as a parameter, since it is a part of URL
mobile_3ds_hmac	HMAC, which was originally used, as a parameter in /encrypted_payment_instruments request

Successful Encrypted Payment Instrument 3DS Confirmed Response Example

```
{
 "encrypted_payment_instrument":{
    "cc_token_encrypted":"QEVuQwBAEACie0slGXnEr3sdfdfdfSAFSfd4343232fsDFDFDFDsdfdFASFD7N"
 }
}
```

3.5 Decrypt Encrypted Token

Request Parameters

URI: /encrypted_payment_instrument_decryptions
Request body key: encrypted_payment_instrument_decryption

Parameter	Description
api_username	Merchant's API username. Please see General Settings section of the Merchant Portal for exact values.
cc_token_encrypted	Encrypted token sent to end-user from the encrypted_instrument_tokens API call.
nonce	Unique request identificator (see below for details).
timestamp	Timestamp of request's creation time (see below for details).
hmac	HMAC value of the request (see below for calculation details).

Request Body Example

```
{
 "encrypted_payment_instrument_decryption":{
   "api_username":"12345678",
   "nonce":"a9b7f7e794367c2c85d73154a01b9902",
   "timestamp":1427933807,
   "hmac":"41c60a44a64f4fd7d6622bc40148fef9bcd1ecae",
   "cc_token_encrypted":"XpgZ/UTsNhBHJ6LJTGsGRQ==-1439355600",
 }
}
```

Successful Response Example

```
{
 "encrypted_payment_instrument_decryption":{
   "cc_token":"123456789012345678901234"
 }
}
```

Unsuccessful Response Example

```
{
 "errors": [
   {
     "code":4021,
     "message":"Unable to decrypt encrypted payment instrument"
   }
 ]
}
```

Appendix 1 Integration Testing

AP 1.1 Testing checklist

Test cards can be found under “Quick References”

As the entire payment workflow has several steps, it’s necessary to verify that all steps and possible scenarios are working flawlessly.

The testing should include the following scenarios:

1. Sending a valid (format wise) API request for payment initiation.
2. Correct processing of the different payment result response messages from EveryPay system:
    a. Asynchronous server-to-server callback message
    b. The redirect HTTP PUT (faked over POST with _method=“put” hidden param) message (only relevant for redirect payment page solution)
    c. The iFrame-to-parent post message (only relevant for iFrame payment page solution).
2. Updating the order status correctly in merchant system based on the payment response(s) received from EveryPay system.
3. Displaying the correct page and contents to the buyer in merchant system for different scenarios.
4. Compare the payment result in EveryPay merchant Portal with the order status in merchant system to ensure that the merchant systems order status reflects the expected result.
5. All of the steps above (where relevant) should be verified with different possible payment scenarios:
    a. Successful payment (`payment_state='settled'` or `'authorised'`)
    b. Failed payment (`payment_state='failed'`)
    c. Cancelled payment (`payment_state='cancelled'` or `'waiting_for_3ds_response'`) -- only relevant for redirect payment page, when buyer clicks the BACK button on payment page
    d. Abandoned payment -- nothing is returned from EveryPay system because the buyer didn’t finish the payment process (e.g. closed the browser window).

Changelog

Date	Change
2019-02-21	● Added sandbox parameter to iFrame example: <iframe id="iframe-payment-container" name="iframe-payment-container", width="400", height="400" sandbox="allow-top-navigation allow-forms allow-popups allow-scripts allow-same-origin"></iframe>
2018-10-29	● Added explanation to HTTP PUT : HTTP PUT(faked over POST with _method="put" hidden param)
2018-10-25	● Updated Mastercard test card to a new one ● Added Mastercard test card with 22 beginning
2018-04-02	● Updated Visa test card CVC code
2017-01-24	● Initial document - successor of Everypay Payment API documentation.