Unlimint

CardPay API Reference

Version: 2.70

Table of Contents

Introduction

Welcome to the CardPay API. You use our APIs to receive payments and check payment status.
For recent changes please see the ***changelog***.

Environments

There are two environments available for integration:

1. Sandbox environment: https://sandbox.cardpay.com
2. Production environment: https://cardpay.com

Sandbox Environment

Sandbox provides full functionality but it only emulates processing, no actual bank transactions are made. It supports the following card PANs:

Card PAN	3-D Secure result	Transaction result
4000000000000002	3-D secure version 1, full authentication	Payment has CONFIRMED status
5555555555554444	3-D secure version 1, full authentication	Payment has DECLINED status
4000000000000044	3-D secure version 1, full authentication	Payment has AUTHORIZED status
4000000000000036	3-D secure version 1, attempt	Payment has CONFIRMED status
4000000000000093	3-D secure version 2, frictionless flow, success	Payment has CONFIRMED status
4000000000000069	3-D secure version 2, frictionless flow, attempt	Payment has CONFIRMED status
4000000000000085	3-D secure version 2, challenge flow, full authentication	Payment has CONFIRMED status
4000000000000077	No 3-D Secure	Payment has CONFIRMED status
5555555555554477	No 3-D Secure	Payment has DECLINED status
4000000000000051	No 3-D Secure	Payment has AUTHORIZED status

If merchant receives non-financial error codes or technical error codes, he needs to contact Cardpay manager.

AUTHORIZED cards must be used to test antifraud system reaction with compulsory transfer payment to preauthorization.

You can use any cardholder name, expiry date and CVV2/CVC2 with these PANs. 3-D Secure is also emulated with a page that doesn't require any password but only shows you 2 buttons. One button is for successful authentication, another is for failed authentication. Note, that when you chose to fail authentication, order is always declined, no matter what PAN was used.

Don't use real cards on Sandbox environment.

To request any changes in settings of Sandbox environment you can directly contact our support via email or Skype.

Production Environment

Once you complete integration with Sandbox environment you will be provided with Production credentials. These are completely different Merchant and User accounts, not related with the ones on Sandbox.

If same user has accounts both on Sandbox and Production, please set different passwords on different environments.

Production always makes real bank transactions, cards from Sandbox are not supported on this environment.

To request any changes in settings of Production environment you need to contact CardPay manager.

API Entry Points

There are three merchant APIs:

1. First one is the Payment API which provides complete server-to-server operations. It is used for sending new orders to the system. It provides a Payment Page Mode. When customer enters card detail on a form located on CardPay payment page or customer enters card details on merchant's site (Gateway Mode).
2. Second API is the Service API. It has additional methods to provide information about payments and to change payment status.
3. Third one is REST API that is still in development and supports only limited number of operations but is intended to replace the first two.

All APIs are available via HTTP protocol using HTTP POST or HTTP GET methods.

Note that according to HTTP standards all parameter values must be URLEncoded before sending and URLDecoded after receiving to escape special characters like ' +, /, &, = ' used in

URLs to separate words, folders and parameters. When API is used for server-to-server communications the reply is printed inside of response body as text as an XML without any encoding.

Payment API Access Modes: Payment Page And Gateway

Depending on account settings Payment API can work in 2 different ways: as a ***Payment Page*** or as a ***Gateway***.

3-D Secure Version 2

3D Secure 2: a seamless integration to increase customer conversion rates and leverage your risk management.

Here's the rundown on Cardpay's new addition to Payment API and the Payment Page with additional customer information to get you started with 3-D Secure 2.

Due to 3-D Secure 2 rules and terms (see more information) Cardpay should collect additional customer information for 3-D Secure 2 verification.

Additional information can be collected by the merchant independently (on his side) and sent in the request (Payment Page and Gateway modes) or the customer should be enter all the necessary information on the additional page. This page will be displayed in cases when this information was not in the request from the merchant.

Additional information in POST request (create payment or recurring) should include the following fields:

- <order trans_type="..."></order>
- <order phone="..."></order>
- <order work_phone="..."></order>
- <order home_phone="..."></order>
- <order ...><card acct_type="..." /></order>
- <order ...><billing addr_line_1="..." /></order>
- <order ...><billing addr_line_2="..." /></order>

Payment Page Mode

Overview

Payment Page Mode is used when Merchant chooses to use our payment page. Customer is redirected from merchant's website to our payment page to enter card details. Customer data entered on our payment page is protected and managed by CardPay and certified to and complies with PCI DSS standard. All customer data is sent via secure connection.

CardPay API Endpoint

- **Test URL:** https://sandbox.cardpay.com/MI/cardpayment.html
- **Live URL:** https://cardpay.com/MI/cardpayment.html

HTML Form

Example HTML Form

```
<?php
//html form template
$template = <<<EOT
<form method="post" action="%cardpay_url%">
    <input type="hidden" name="orderXML" value="%cardpay_orderxml%">
    <input type="hidden" name="sha512" value="%cardpay_sha512%">
    <input type="image" width="170" height="30" src="//www.cardpay.com/images/logo.jpg" alt="Pay Now With CardPay!"/>
</form>
EOT;
```

```
<!DOCTYPE html>
<html>
    <body>
        <form action="https://cardpay.com/MI/cardpayment.html" method="POST">
            <input type="hidden" name="orderXML"
VALUE="PG9yZGVyIHdhbGxldF9pZD0iMjEiIG51bWJlcj0iNDU4MjEwIiBkZXNjcmlwdGlvbj0iQmVzdCBldmVyIHNlZW4gVC1TaGlydH
MsIDExMDIxIiBjdXJyZW5jeT0iVVNEIiBhbW91bnQ9IjI5MS44NiIgZW1haWw9ImN1c3RvbWVyQGV4YW1wbGUuY29tIi8+"/>
            <input type="hidden" name="sha512"
VALUE="b258f153463c8787535a6e90836a4927b2ccddb5af64999ff4b530d74fff2a34959ff77d0b003004d0f9f6d923702c78b579187
287809f820c8e77e0be5c9254"/>
            <input type="image" width="170" height="30" src="https://www.cardpay.com/images/logo.jpg" alt="Pay Now With CardPay!"/>
        </form>
    </body>
</html>
```

To use API in this mode customer must be provided with HTML Form with filled hidden input fields. This Form is posted to the API URL by pressing "Pay" button.

There are only 2 parameters of request: orderXML and sha512.

- orderXML - Base-64 encoded ***XML Request***
- sha512 - see ***Calculating Digest*** section

XML Request

XML Request example using only the minimum required parameters

```
<order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="291.86"
email="customer@example.com"/>
```

Elements and attributes are case in-sensitive. Single quotes may be used instead of double quotes.

Elements in **bold** are mandatory. Attributes in **bold** are mandatory only if Element is present, default value is shown after equals sign.

Following data types are used:

- String(XX) - any Unicode characters, some validation may be applied though. XX is supported length in Unicode characters, data is truncated if exceeded.
- Integer - only decimal digits.
- Decimal - decimal digits with dot as a decimal separator.
- Boolean - boolean flag: *“true”* / *“false”*.

Order Element Structure

Element	Attribute	Type	Description
order	**wallet_id**	Integer	Unique merchant’s ID used by the CardPay payment system
	number	String(50)	Order ID used by the merchant’s shopping cart
	description	String(200)	Description of product/service being sold
	currency	String(3)	***ISO 4217*** currency code
	amount	Decimal	The total order amount in selected currency with dot as a decimal separator, must be less than 100 millions
	email	String(256)	Customer’s e-mail address
	customer_id	String(256)	Customer’s ID in the merchant’s system
	is_two_phase=false	Boolean	If set to *“true”*, the amount will not be captured but only blocked.
	recurring_begin=false	Boolean	If set to *“true”*, the payment can be repeated later using recurring_id from response. See section ***Recurring***

Element	Attribute	Type	Description
	recurring_id	String(32)	Repeating payment sent before. See section ***Recurring***
	generate_card_token=false	Boolean	If set to *"true"*, a ***Card Token*** will be generated and returned in the response or in the notification XMLs
	card_token	String(36)	***Card Token*** used instead of card information
	authentication_request=false	Boolean	If set to *"true"*, amount must not be present in request, no payment will be made, only cardholder authentication will be performed. Also can be used to generate Card Token.
	locale=en	String(2)	Preferred locale for the payment page (***ISO 639-1*** language code). The default locale will be applied if the selected locale is not supported. Supported locales are: "en", "ru", "zh", "ja"
	dynamic_descriptor	String(25)	Short description of the service or product, must be enabled by CardPay manager to be used
	trans_type	String(2)	Identifies the type of transaction being authenticated. Values accepted: 01 = Goods/Service Purchase, 03 = Check Acceptance, 10 = Account Funding, 11 = Quasi-Cash Transaction, 28 = Prepaid Activation and Load. Note: Values derived from the 8583 ISO Standard.
	phone	String(8-18)	Customer phone number. Recommended to send phone number in following format "+1 111111111" with country code and subscriber sections (only digits are accepted) of the number, "+" as prefix and "space" as delimiter. Refer to ITU-E.164 for additional information on format and length.

Element	Attribute	Type	Description
	work_phone	String(8-18)	The work phone number provided by the Cardholder. Required (if available) unless market or regional mandate restricts sending this information. Format: string (8-18 symbols) country code + Subscriber number. Refer to ITU-E.164 for additional information on format and length. Example: +1 111111111
	home_phone	String(8-18)	The home phone number provided by the Cardholder. Required (if available) unless market or regional mandate restricts sending this information. Format: string (8-18 symbols) country code + Subscriber number. Refer to ITU-E.164 for additional information on format and length. Example: +1 111111111
	note	String(100)	Note about the order that will not be displayed to customer
	return_url	String	Overrides default *success URL*, *inprocess URL*, *decline URL* and *cancel URL*. See section ***Return URLs***. return_url can be used separately or together with other url parameters
	success_url	String	Overrides default *success URL* only. See section ***Return URLs***
	decline_url	String	Overrides default *decline URL* only. See section ***Return URLs***
	cancel_url	String	Overrides default *cancel URL* only. See section ***Return URLs***
	inprocess_url	String	Overrides default *inprocess URL* only. See section ***Return URLs***
order/shipping		***shipping*** element	Represents an address where the order will be delivered to. See ***shipping*** element structure below
order/items		List of ***item*** elements	List of order positions (items in the shopping cart). See ***item*** element structure below
order/flights		List of ***flight*** elements	List of flights and passenger information for airlines. See ***flight*** element structure below

Shipping Element Structure

Example of XML Request with Shipping element:

```
<order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="270"
email="customer@example.com">
  <shipping country="USA" state="NY" zip="10001" city="New York" addr_line_1="450 W." addr_line_2="33 Street" phone="+1
111111111"/>
</order>
```

SHIPPING is the address where the order will be delivered. It is used in Anti-fraud System and also can be seen in Payment Manager. But it may be omitted and all the fields of it except country may be omitted too.

Element	Attribute	Type	Description
shipping	**country**	String(3)	***ISO 3166-1*** code of delivery country: 2 or 3 latin letters or numeric code
	state	String(20)	Delivery state or province. May include whitespaces, hyphens, apostrophes, commas and dots
	zip	String(12)	Delivery postal code
	city	String(20)	Delivery city. May include whitespaces, hyphens, apostrophes, commas and dots
	street	String(2-100)	(Deprecated. Replaced by addr_line_1, addr_line_2). Delivery street address. May include whitespaces, hyphens, apostrophes, commas, quotes, dots, slashes and semicolons
	phone	String(5-20)	Valid customer phone number. Example: +1 111111111
	addr_line_1	String(0-50)	First line of the street address or equivalent local portion of the Cardholder shipping address associated with the card used for this purchase. Can include street and house number
	addr_line_2	String(0-50)	Second line of the street address or equivalent local portion of the Cardholder shipping address associated with the card used for this purchase.

Item Element Structure

Example of XML Request with Items element:

```
<order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="270"
email="customer@example.com">
  <items>
    <item name="T-Shirt" description="Funniest print you ever saw" count="1" price="30"/>
    <item name="T-Shirt" description="Coolest print you ever saw" count="4" price="60"/>
  </items>
</order>
```

Item element represents individual order item or service.

Element	Attribute	Type	Description
item	**name**	String(50)	The name of product / service, provided to the customer
	description	String(200)	The description of product / service, provided to the customer
	count	Numeric	The count of product / service, provided to the customer. Any positive number
	price	Decimal	Price of product / service with dot as a decimal separator, must be less than 100 millions

Flights Element Structure

Example of XML Request with Flights element:

```
<order wallet_id="21" number="458210" description="Flight from London to Barcelona with transfer in Rome" currency="USD"
amount="270" email="customer@example.com">
    <flights passenger_name='JAMES BOND' departure_date='31.12.2018' origination_code='LHR' travel_agency_code='121212'
travel_agency_name='Example.com'>
        <flight index='1' flight_number='AG007' destination_code='CIA'/>
        <flight index='2' flight_number='JB007' destination_code='BCN'/>
    </flights>
</order>
```

Flights element represents common airlines information for the whole travel like passenger name or travel agency. Each flight element represents a separate flight. Since flights order matters it is recommended to additionally control it by adding index attribute to each flight element.

Element	Attribute	Type	Description
flights	passenger_name	String(20)	The name of the passenger using only latin alphabet
	origination_code	String(3)	Airport of origination code
	departure_date	String(10)	Departure date in DD.MM.YYYY format
	travel_agency_code	String(8)	Travel agency code
	travel_agency_name	String(25)	Travel agency name using only latin alphabet
	computerized_res_system	String(4)	Computerized reservation system code
	credit_reason_indicator	String(1)	Credit reason indicator
	ticket_change_indicator	String(1)	Ticket change indicator
	is_restricted	Boolean	Restricted ticket indicator
flights/flight	index	Numeric	Sequence number of a flight
	number	String(5)	Flight number
	carrier_code	String(2)	Carrier code
	destination_code	String(3)	Airport of destination code
	service_class_code	String(1)	Service class code
	fare_basis_code	String(6)	Fare basis code
	stop_over_code	String(1)	Stop over code

Recurring

You can begin recurring by sending usual Order with the attribute “recurring_begin” having value*“true”* in it and then repeat payments from the same card without asking the cardholder to enter card details again. To do this you need to get the value of “recurring_id” attribute from the ***Payment Result XML***. When you continue recurring with “recurring_id” Payment Page is not displayed because the same card is used as when recurring began. Instead ***Payment Result XML*** will be sent in response.

Card Token

Card Token feature is almost the same as Recurring, but the difference is that in this case each payment is made with Cardholder present and requires CVV2/CVC2 and 3-D Secure if available. You can obtain Card Token by sending usual Order with the attribute “generate_card_token” having value *“true”* in it. The generated Card Token is sent only in a***Callback*** and only if payment was successful. Each time token is requested, the new one is generated even for the same card. To use Card Token you can send it in “card_token” attribute, so customer will not have to enter card details again, only

CVV2/CVC2 and pass 3-D Secure if needed. Card Token cannot be used more than 1 year after token was generated.

Response

After Payment is completed, the Customer is redirected to***Return URL***.

Callback

See section ***Callbacks***.

Gateway Mode

Overview

In Gateway Mode the Customer enters credit card data on the Merchant's website and order is sent from Merchant to CardPay server-to-server. In this case the Merchant has to collect cardholder data on his website that must be PCI DSS certified for that and then send it as a POST request to the CardPay Payment Endpoint.

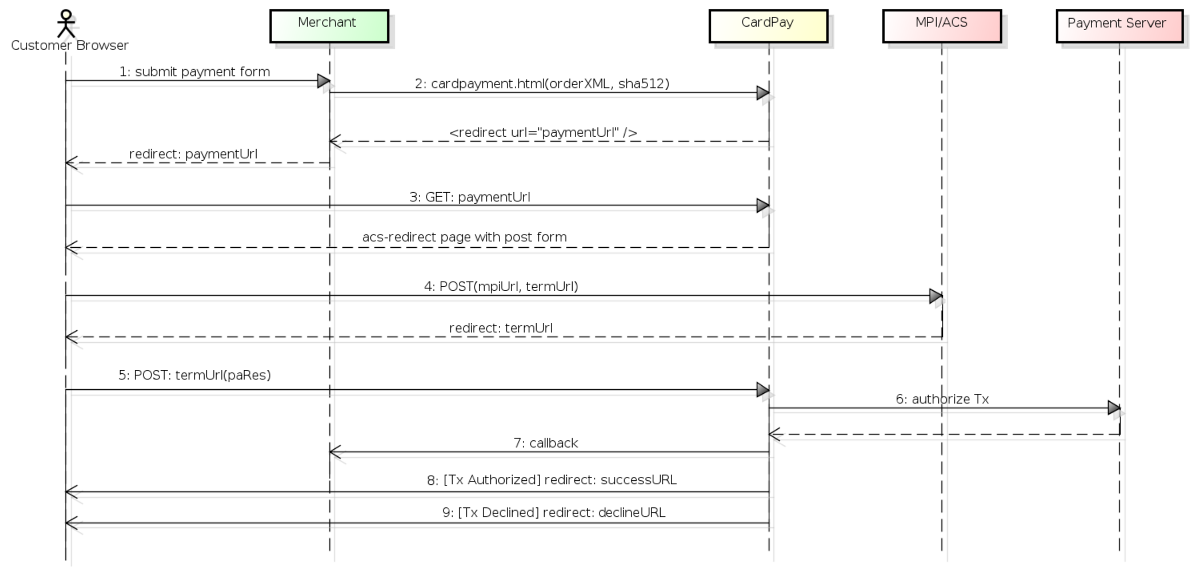
In response to Merchant's request, CardPay server sends URL the Customer should be redirected to. When payment is complete, customer will be redirected back to the shop to one of the ***predefined URLs***. Additionally, ***callbacks*** and/or email notifications will be sent.

CardPay API Endpoint

- **Test URL:** POST: https://sandbox.cardpay.com/MI/cardpayment.html
- **Live URL:** POST: https://cardpay.com/MI/cardpayment.html

In this mode POST method must be used, all data must be sent in a request's body.

Workflow



1. Merchant shows a payment page to the customer. Customer fills card details on the payment page and submits the form.
2. Merchant constructs a payment request (orderXML) with information provided by the customer and sends it to the CardPay payment endpoint. CardPay sends an XML response with the redirect URL to the Merchant: <redirect url="redirectUrl"/>
3. Merchant redirects the customer to the provided URL for card authorization.
4. CardPay shows redirection page to the Customer. Customer interacts with the Issuing Bank's Access Control Server. On this page they may be asked for 3-D Secure password.
5. After that Customer is redirected back to CardPay server with the authentication result (PaRes).
6. Upon authentication completion, payment is processed.
7. Callback message with the transaction status is sent to callback URL, provided by Merchant. Callback URL is set in Merchant account.
8. After that Customer is redirected back to the Merchant's site. Depending on the authorization result the Customer is redirected to:
    - Success URL (8) - if transaction was successful. If not specified, the Customer is redirected to

the Merchant’s website home URL.
  - Decline URL (9) - if transaction has failed. If not specified, the Customer is redirected to the Merchant’s website home URL.

Request Parameters

PHP code example

```
<?php
$ch = curl_init("payment-page-URL");
$params = array();
$params['orderXML'] = $orderXML;    // base64-encoded order XML value
$params['sha512'] = $sha512;
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($params));
curl_setopt($ch, CURLOPT_FAILONERROR, 1);
$response = curl_exec($ch);
curl_close($ch);
```

This is server-to-server request. There are only 2 parameters of request: orderXML and sha512.

- orderXML - Base-64 encoded ***XML Request***
- sha512 - see ***Calculating Digest*** section

XML Request

XML Request example using only the minimum required parameters

```
<order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="291.86"
email="customer@example.com">
    <card num="4000000000000002" holder="MR CARDHOLDER" cvv="123" expires="02/2020"/>
</order>
```

Elements and attributes are case-insensitive. Single quotes may be used instead of double quotes.

Elements in **bold** are mandatory. Attributes in **bold** are mandatory only if Element is present, default value is shown after equals sign.

Elements in ***bold and italic*** are conditional, follow the links for details.

Following data types are used:

- String(XX) - any Unicode characters, some validation may be applied though. XX is supported length in Unicode characters, data is truncated if exceeded.
- Integer - only decimal digits.
- Decimal - decimal digits with dot as a decimal separator.
- Boolean - boolean flag: *“true”* / *“false”*.

Order Element Structure

Element	Attribute	Type	Description
order	**wallet_id**	Integer	Unique merchant's ID used by the CardPay payment system
	number	String(50)	Order ID used by the merchant's shopping cart
	description	String(200)	Description of product/service being sold and ID of customer
	currency	String(3)	***ISO 4217*** currency code
	amount	Decimal	The total order amount in selected currency with dot as a decimal separator, must be less than 100 millions
	email	String(256)	Customer's e-mail address
	customer_id	String(256)	Customer's ID in the merchant's system
	is_two_phase=false	Boolean	If set to *"true"*, the amount will not be captured but only blocked.
	recurring_begin=false	Boolean	If set to *"true"*, the payment can be repeated later using recurring_id from response. See section ***Recurring***
	recurring_id	String(32)	Repeating payment sent before. See section ***Recurring***
	generate_card_token=false	Boolean	If set to *"true"*, a ***Card Token*** will be generated and returned in the response or in the notification XMLs
	card_token	String(36)	***Card Token*** used instead of card information
	authentication_request=false	Boolean	If set to *"true"*, amount must not be present in request, no payment will be made, only cardholder authentication will be performed. Also can be used to generate Card Token.
	dynamic_descriptor	String(25)	Short description of the service or product, must be enabled by CardPay manager to be used

Element	Attribute	Type	Description
	trans_type	String(2)	Identifies the type of transaction being authenticated. Values accepted: 01 = Goods/Service Purchase, 03 = Check Acceptance, 10 = Account Funding, 11 = Quasi-Cash Transaction, 28 = Prepaid Activation and Load. Note: Values derived from the 8583 ISO Standard.
	phone	String(8-18)	Customer phone number. Recommended to send phone number in following format “+1 111111111” with country code and subscriber sections (only digits are accepted) of the number, “+” as prefix and “space” as delimiter. Refer to ITU-E.164 for additional information on format and length.
	work_phone	String(8-18)	The work phone number provided by the Cardholder. Required (if available) unless market or regional mandate restricts sending this information. Format: string (8-18 symbols) country code + Subscriber number. Refer to ITU-E.164 for additional information on format and length. Example: +1 111111111
	home_phone	String(8-18)	The home phone number provided by the Cardholder. Required (if available) unless market or regional mandate restricts sending this information. Format: string (8-18 symbols) country code + Subscriber number. Refer to ITU-E.164 for additional information on format and length. Example: +1 111111111
	note	String(100)	Note about the order that will not be displayed to customer
	return_url	String	Overrides default *success URL* and *decline URL*. See section ***Return URLs***
	success_url	String	Overrides default *success URL* only. See section ***Return URLs***
	decline_url	String	Overrides default *decline_URL* only. See section ***Return URLs***
order/card		***card*** element	Represents a payment card. See ***card*** element structure below

Element	Attribute	Type	Description
order/billing		***billing*** element	Represents an address associated with the card. See ***billing*** element structure below
order/shipping		***shipping*** element	Represents an address where the order will be delivered to. See ***shipping*** element structure below
order/items		List of ***item*** elements	List of order positions (items in the shopping cart). See ***item*** element structure below
order/flights		List of ***flight*** elements	List of flights and passenger information for airlines. See ***flight*** element structure below

Card Element Structure

Example of XML Request with Card element:

```
 <order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="270"
email="customer@example.com">
   <card num="4000000000000002" holder="MR CARDHOLDER" cvv="123" expires="02/2020"/>
</order>
```

Element	Attribute	Type	Description
card	***num***	Integer(13-19)	Customer's card number (PAN). Any valid card number, may contain spaces. Required if 'card_token' is not present, see ***Card Token***
	holder	String(50)	Customer's cardholder name. Any valid cardholder name. Required if 'card_token' is not present, see ***Card Token***
	expires	String(7)	Customer's card expiration date. Format: mm/yyyy. Required if 'card_token' is not present, see ***Card Token***
	cvv	Integer(3)	Customer's CVV2 / CVC2
	acct_type	String(2)	Indicates the type of account. For example, for a multi-account card product. Required if 3DS Requestor is asking Cardholder which Account Type they are using before making the purchase. Required in some markets (for example, for Merchants in Brazil). Values accepted: 01 = Not Applicable, 02 = Credit, 03 = Debit

Billing Element Structure

Example of XML Request with Billing element:

```
 <order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="270"
email="customer@example.com">
```

```
    <card num="4000000000000002" holder="MR CARDHOLDER" cvv="123" expires="02/2020"/>
    <billing country="USA" state="NY" zip="10001" city="New York" addr_line_1="450 W." addr_line_2="33 Street" phone="+1
111111111"/>
</order>
```

BILLING is the customer’s billing address. Depending on account settings Billing may be required. Element structure is identical to ***shipping address element***.

Element	Attribute	Type	Description
billing	**country**	String(3)	***ISO 3166-1*** code of billing country: 2 or 3 latin letters or numeric code
	state	String(20)	Billing state or province. May include whitespaces, hyphens, apostrophes, commas and dots
	zip	String(12)	Billing postal code
	city	String(20)	Billing city. May include whitespaces, hyphens, apostrophes, commas and dots
	street	String(2-100)	(Deprecated. Replaced by addr_line_1, addr_line_2). Billing street address. May include whitespaces, hyphens, apostrophes, commas, quotes, dots, slashes and semicolons
	addr_line_1	String(0-50)	First line of the street address or equivalent local portion of the Cardholder billing address associated with the card used for this purchase. Should include street and house number. Required (if available) unless market or regional mandate restricts sending this information. May include whitespaces, hyphens, apostrophes, commas, quotes, dots, slashes and semicolons.
	addr_line_2	String(0-50)	Second line of the street address or equivalent local portion of the Cardholder billing address associated with the card used for this purchase. Required (if available) unless market or regional mandate restricts sending this information.

Shipping Element Structure

Example of XML Request with Shipping element:

```
 <order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="270"
email="customer@example.com">
    <card num="4000000000000002" holder="MR CARDHOLDER" cvv="123" expires="02/2020"/>
    <shipping country="USA" state="NY" zip="10001" city="New York" addr_line_1="450 W." addr_line_2="33 Street" phone="+1
111111111"/>
</order>
```

SHIPPING is the address where the order will be delivered. It is used in Anti-fraud System and also can be seen in Payment Manager. But it may be omitted and all the fields of it except country may be omitted too. Element structure is identical to ***billing address element***.

Element	Attribute	Type	Description
shipping	**country**	String(3)	***ISO 3166-1*** code of delivery country: 2 or 3 latin letters or numeric code
	state	String(20)	Delivery state or province. May include whitespaces, hyphens, apostrophes, commas and dots
	zip	String(12)	Delivery postal code
	city	String(20)	Delivery city. May include whitespaces, hyphens, apostrophes, commas and dots
	street	String(2-100)	(Deprecated. Replaced by addr_line_1, addr_line_2). Delivery street address. May include whitespaces, hyphens, apostrophes, commas, quotes, dots, slashes and semicolons
	phone	String(5-20)	Valid customer phone number
	addr_line_1	String(0-50)	First line of the street address or equivalent local portion of the Cardholder shipping address associated with the card used for this purchase. Can include street and house number
	addr_line_2	String(0-50)	Second line of the street address or equivalent local portion of the Cardholder shipping address associated with the card used for this purchase.

Item Element Structure

Example of XML Request with Items element:

```
 <order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="270"
email="customer@example.com">
   <card num="4000000000000002" holder="MR CARDHOLDER" cvv="123" expires="02/2020"/>
   <items>
      <item name="T-Shirt" description="Funniest print you ever saw" count="1" price="30"/>
      <item name="T-Shirt" description="Coolest print you ever saw" count="4" price="60"/>
   </items>
</order>
```

Item element represents individual item or service.

Element	Attribute	Type	Description
item	**name**	String(50)	The name of product / service, provided to the customer
	description	String(200)	The description of product / service, provided to the customer
	count	Numeric	The count of product / service, provided to the customer. Any positive number
	price	Decimal	Price of product / service with dot as a decimal separator, must be less than 100 millions

Flights Element Structure

Example of XML Request with Flights element:

```
 <order wallet_id="21" number="458210" description="Flight from London to Barcelona with transfer in Rome" currency="USD"
amount="270" email="customer@example.com">
   <flights passenger_name='JAMES BOND' departure_date='31.12.2018' origination_code='LHR' travel_agency_code='121212'
travel_agency_name='Example.com'>
       <flight index='1' flight_number='AG007' destination_code='CIA'/>
       <flight index='2' flight_number='JB007' destination_code='BCN'/>
   </flights>
</order>
```

Flights element represents common airlines information for the whole travel like passenger name or travel agency. Each flight element represents a separate flight. Since flights order matters it is recommended to additionally control it by adding index attribute to each flight element.

Element	Attribute	Type	Description
flights	passenger_name	String(20)	The name of the passenger using only latin alphabet
	origination_code	String(3)	Airport of origination code
	departure_date	String(10)	Departure date in DD.MM.YYYY format
	travel_agency_code	String(8)	Travel agency code
	travel_agency_name	String(25)	Travel agency name using only latin alphabet
	computerized_res_system	String(4)	Computerized reservation system code
	credit_reason_indicator	String(1)	Credit reason indicator
	ticket_change_indicator	String(1)	Ticket change indicator
	is_restricted	Boolean	Restricted ticket indicator
flights/flight	index	Numeric	Sequence number of a flight
	number	String(5)	Flight number
	carrier_code	String(2)	Carrier code
	destination_code	String(3)	Airport of destination code
	service_class_code	String(1)	Service class code
	fare_basis_code	String(6)	Fare basis code
	stop_over_code	String(1)	Stop over code

Recurring

You can begin recurring by sending usual Order with the attribute “recurring_begin” having value “yes” in it and then repeat payments from the same card without asking the cardholder to enter card details again. To do this you need to get the value of “recurring_id” attribute from the ***Payment Result XML***. When you continue recurring with “recurring_id” you can omit “order/card” tag in orderXML because the same card is used as when recurring began. Also instead of redirect XML for recurring continue ***Payment Result XML*** will be sent in response.

Card Token

Card Token feature is almost the same as Recurring, but the difference is that in this case each payment is made with Cardholder present and requires CVV2/CVC2 and 3-D Secure if available. You can obtain Card Token by sending usual Order with the attribute “generate_card_token” having value *“true”* in it. The generated Card Token is sent only in a***Callback*** and only if payment was successful. Each time token is requested, the new one is generated even for the same card. To use Card Token you can send it in “card_token” attribute, in this case “order/card” tag must contain only “cvv” field. Card Token cannot be used more than 1 year after token was generated.

HTTP Response

Example of XML Response:

```
<?xml version="1.0" encoding="utf"?><redirect url="https://cardpay.com/MI/payments/redirect?token=abc12345"/>
```

In Gateway Mode the Merchant may receive either the result of payment or XML response containing URL a Customer should be redirected to. Customer may open this URL via either GET or POST method.

The response is in XML format containing one `redirect` tag with one `url` attribute:

```
<?xml version="1.0" encoding="utf"?><redirect url="some_url_value"/>
```

After Payment is completed, Customer is redirected to ***Return URL***.

Callback

See section ***Callbacks***.

Calculating Digest

```
<?php
$order = '<order wallet_id="21" number="458210" description="Best ever seen T-Shirt" currency="USD" amount="291.86"
email="customer@example.com"/>';
$secretKey = 'se1cr2et3w0r4d';
$orderXML = base64_encode($order);
$sha512 = hash('sha512', $order . $secretKey);
```

```
String order = "<order wallet_id=\"21\" number=\"458210\" description=\"Best ever seen T-Shirt\" currency=\"USD\"
amount=\"291.86\" email=\"customer@example.com\"/>";
String secretKey = "se1cr2et3w0r4d";
java.security.MessageDigest digest = java.security.MessageDigest.getInstance("SHA-512");
digest.update((order + secretKey).getBytes(StandardCharsets.UTF_8));
String orderXML = java.util.Base64.getEncoder().encodeToString(order.getBytes(StandardCharsets.UTF_8));
String sha512 = org.apache.commons.codec.binary.Hex.encodeHexString(digest.digest());
```

Digest is used for authentication and protecting the message from changes. It is calculated from the order XML and the Secret Word you received with your Wallet ID.

For example if you have following access parameters:

- Wallet ID = 21
- Secret Word = se1cr2et3w0r4d

And you are sending the following order:

```
<order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="291.86"
email="customer@example.com"/>
```

Then you'll need to calculate SHA-512 digest of the following string:

```
"<order wallet_id="21" number="458210" description="Best ever seen T-Shirts" currency="USD" amount="291.86"
email="customer@example.com"/>se1cr2et3w0r4d"
```

That will result in the following:

```
8877da7b52eea2266d9b4f2c7dd4630dd3ea6d1e066a3f2f0921b5163196095a3a7028ba5cf80671471f313719325de1c694f3a9927eaa7cc
6370e6ddbe82d3d
```

And these will be the parameters to be sent to API:

- **orderXML** = PG9yZGVyIHdhbGxldF9pZD0iMjEiIG51bWJlcj0iNDU4MjEwIiBkZXNjcmlwdGlvbj0iQmVzdCBldmVyIHNlZW4gVC1TaGlydHMiIGN1cnJlbmN5PSJVU0QiIGFtb3VudD0iMjkxLjg2IiBlbWFpbD0iY3VzdG9tZXJAZXhhbXBsZS5jb20iLz4=
- **sha512** = 8877da7b52eea2266d9b4f2c7dd4630dd3ea6d1e066a3f2f0921b5163196095a3a7028ba5cf80671471f313719325de1c694f3a9927eaa7cc6370e6ddbe82d3d

Return URLs

Example

The `{...}` placeholders will be replaced with real values specific for particular Order.

```
https://example.com/result?n={number}&s={status}&l={locale}&w={wallet_id}&t={note}
```

Return URLs are the URLs where customer returns by pressing “Back to the shop” or “Cancel” button in Payment Page mode and redirected automatically in Gateway mode.

URL Selection

There can be 4 different URLs:

- *Success URL*: where Customer returns after successful transaction
- *Decline URL*: where Customer returns after failed transaction
- *Cancel URL*: where Customer returns when no transaction was made (only in Payment Page mode)
- *InProcess URL*: where Customer returns when transaction was started but not yet completed

By default they all point to the Shop’s home page. The default URLs can be changed only by CardPay manager, but they can be sent with each request (see Order element attributes).

URL Placeholders

URLs may contain some of the following placeholders to be replaced with actual values:

Placeholder	Description
{id}	will be replaced by CardPay’s order ID
{number}	will be replaced by Shop’s order number
{status}	one of these: *approved*, *pending* or *declined*
{locale}	locale received with order if any, otherwise default one
{wallet_id}	wallet_id received with order
{customer_id}	parameter customer_id provided by the Merchant in XML request
{note}	parameter note provided by the Merchant in XML request
{order_type}	parameter order_type indicates the type of transaction for a which a callback is generated. *! used only for Callback URLs, not for Return URLs.*

If placeholder is in upper case, value will be sent in upper case, except for`note`.

Note, that all values are URLEncoded to be compatible with HTTP specifications.

Payment Result XML

Example:

```
<order id="299150" number="458210" status="APPROVED" description="CONFIRMED" date="15.01.2015 10:30:45" is_3d="true"
approval_code="DK3H25" amount="21.12" currency="USD" />
```

This XML is sent to the Callback URL as HTTP POST parameters `orderXML` and `sha512` constructed in the same way as payment request.

All TAGs and ATTRIBUTE names are sent in lower case.

Attribute	Example	Description	Condition
id	299150	ID assigned to the order in CardPay	Symbol ’-’ is returned when order was cancelled by customer or order was incorrect
refund_id	299151	ID assigned to the refund in CardPay	Only for refund
number	458210	Merchant’s ID of the order	If it was received from Merchant
status	APPROVED	See ***possible values below***	Always present
description	CONFIRMED	CardPay’s message about order’s validation	Always present
date	15.01.2015 10:30:45	Format DD.MM.YYYY hh:mm:ss. For order: date and time when the order was received. For refund: date and time when the refund was received.	Always present
customer_id	11021	Customer’s ID in the merchant’s system	Present if was sent with Order

Attribute	Example	Description	Condition
card_bin or card_num	400000 or …0002 or 400000…0002	Part of card number	Not present by default, ask CardPay manager to enable it if needed, Callback URL must support HTTPS
card_holder	John Silver	Name of cardholder	Not present by default, ask CardPay manager to enable it if needed, Callback URL must support HTTPS
card_expires	02/2017	Card expiration date in MM/YYYY format	Not present by default, ask CardPay manager to enable it if needed, Callback URL must support HTTPS
decline_reason	Insufficient funds	Bank's message about order's decline reason	When transaction was declined
decline_code	05	Optional ***code of the decline***	Included only when transaction is declined and sending of decline codes is enabled by wallet settings
approval_code	DK3H25	Authorization code, provided by bank	Only in case of successful transaction
is_3d	true	Was 3-D Secure authentication made or not	Always present
currency	USD	Transaction currency as received with order	Always present

Attribute	Example	Description	Condition
amount	21.12	Current transaction's amount as received with order, but can be reduced by refunds later. In case of refund notification this amount is before the refund was made	Not present for authentication request
recurring_id	19F0B681E6F74F83AA6AB0162D7BF3A5	ID of recurring that can be used to continue recurring in future	Only for successful recurring begin
refunded	7.04	Refund amount in order's currency	In case of refund notification
note	VIP customer	Note about the order	Present if was sent with Order
ip	IP of customer	IP address of customer	Present if wallet settings has this option enabled. By default the option is not enabled
card_token	067e6162-3b6f-4ae2-a171-2470b63dff00	Generated Card Token	Present if "generate_card_token" flag was set to *"true"* in the payment request and the payment was successful
email	customer@example.com	Customer's e-mail address	Not present by default, ask CardPay manager to enable it if needed

Status Codes

Status field may have one of these values:

Status	Description
APPROVED	Transaction successfully completed, amount was captured
DECLINED	Transaction denied
PENDING	Transaction successfully authorized, but needs some time to be verified, amount was held and can be captured later
VOIDED	Transaction was voided (in case of void notification)
REFUNDED	Transaction was refunded (in case of refund notification)
CHARGEBACK	Customer’s chargeback claim was received (in case of chargeback notification)
CHARGEBACK RESOLVED	Customer’s claim was rejected, same as APPROVED (in case of resolved chargeback notification)

Decline Codes

Code	Description
01	System malfunction
02	Cancelled by customer
03	Declined by Antifraud
04	Declined by 3-D Secure
05	Only 3-D Secure transactions are allowed
06	3-D Secure availability is unknown
07	Limit reached
10	Declined by bank (reason not specified)
11	Common decline by bank
13	Insufficient funds
14	Card limit reached
15	Incorrect card data
16	Declined by bank’s antifraud
17	Bank’s malfunction
18	Connection problem

Callbacks

Handling Callback

```
POST /callback HTTP/1.1
HOST: example.com
Query String Parameters:
orderXML: ...
sha512: ...
```

```
<?php
if (empty($_REQUEST["orderXML"])) {
    Logger::addLog('CardPay callback: orderXML parameter is empty', 4, 1);
    exit;
}
if (empty($_REQUEST["sha512"])) {
    Logger::addLog('CardPay callback: sha512 parameter is empty', 4, 2);
    exit;
}

$cardpaySecretKey = Configuration::get('CARDPAY_SECRET_KEY');
if (empty($cardpaySecretKey)) {
    Logger::addLog('CardPay callback: secret key is empty', 4, 3);
    exit;
}

$cardpayOrderXML = $_REQUEST["orderXML"];
$cardpaySha512 = $_REQUEST["sha512"];

$orderXML = base64_decode($cardpayOrderXML);
$genSha512 = hash('sha512', $orderXML . $cardpaySecretKey);

// if both hashes are the same, the post should come from CardPay Inc
if ($genSha512 != $cardpaySha512) {
    Logger::addLog('CardPay callback: generated SHA512 is not equal to CardPay SHA512', 4, 4, 'generatedSha512', $genSha512);
    exit;
}

// get order XML as object
$orderXMLObj = simplexml_load_string($orderXML);

// order XML fields validation
if (empty($orderXMLObj)) {
    Logger::addLog('CardPay callback: order XML object is empty', 4, 5);
    exit;
}
if (empty($orderXMLObj['number'])) {
    Logger::addLog('CardPay callback: cart id is empty', 4, 6);
    exit;
}
if (empty($orderXMLObj['status'])) {
    Logger::addLog('CardPay callback: order status is empty', 4, 7);
    exit;
}

// order must have APPROVED or PENDING status
// if status is DECLINED then do nothing (exit)
$orderStatus = strtoupper((string)$orderXMLObj['status']);
if (('APPROVED' != $orderStatus) && ('PENDING' != $orderStatus)) {
    // order is declined
    exit;
}

// get cart information
$cartIdStr = (string)$orderXMLObj['number'];
```

After the transaction is processed, ***Payment Result XML*** is sent to callback URL provided by merchant. Callback URL is set in Merchant account.

Callback URL may contain ***URL Placeholders*** like Return URLs do, but note that values set to these placeholders are not signed by digest and should not be used for making decisions, only for optimization. To make any changes in order's status you need to get status, ID and other fields from the `orderXML` after `sha512` digest was validated.

By the security reasons in production environment it is necessary to use HTTPS based callback URL protected by the certificate signed by valid Certificate Authority (CA).

Callbacks are always sent after received order was completed and when it's status was changed. If it was not delivered, callback will be repeated. Callback is marked as delivered if successful code (200) was received. Also you can print some fixed text into your response (for example: "OK") and notify CardPay manager about it then callback is marked as delivered only if successful code (200) and expected text were received.

Intervals for repeating callbacks are following:

- up to 5 minutes after 1st attempt
- 10 minutes after 2nd attempt
- 20 minutes after 3rd attempt
- 30 minutes after 4th attempt
- 45 minutes after 5th attempt
- 1 hour after 6th attempt
- 2 hour after 7th attempt
- 3 hours after 8th attempt
- 5 hours after 9th attempt
- 24 hours after 10th attempt (about 37 hours after 1st attempt)

You can also ask your CardPay manager to set e-mail notification about all cases of callback delivery failures or to disable callback repeating.

Payment result can be also sent by email to merchant and/or customer. It may contain any of the following values:

- Amount and currency
- Confirmation number (authorization code)
- Order number
- Order description

Order Services

The Services are needed to use the Payment Manager functions without using web interface. They provide ability to get the list of orders and change their statuses.

Change Order Status

This method allows you to change order status. It allows to Capture 2-phase transactions, Void transactions when it’s possible or Refund part or full amount of order.

Authentication

To use this service you will need Payment Manager user login and SHA-256 HEX encoded digest of your password. This service will affect only orders available for this user to be modified.

Note: It is strongly recommended to store digest of your password. Do not calculate it from your password for each request!

HTTP Request

Example: For example, if user has login `demo` and password `demo` and you received following ***payment result***:

```
<order id="299150" number="98765" status="PENDING" description="AUTHORIZATION SUCCESS" date="23.12.2015 19:05:38"
is_3d="true" approval_code="DK3H25" amount="4.20" currency="USD" />
```

then your request to **void** payment should look like:

```
POST /MI/service/order-change-status HTTP/1.1
Host: sandbox.cardpay.com
Query String Parameters:
id: 299150
status_to: void
client_login: demo
client_password: 2a97516c354b68848cdbd8f54a226a0a55b21ed138e207ad6c5cbb9c00aa5aea
```

Service Endpoint

This service is available on following entry points URL:

- **Test URL:** https://sandbox.cardpay.com/MI/service/order-change-status
- **Live URL:** https://cardpay.com/MI/service/order-change-status

Request attributes must be sent as HTTP request parameters. According to HTTP specification all parameter values must be URLEncoded.

Method “payout” can be applied only to successful payments, created less than 6 months ago.

Parameters in ***bold and italic*** required only for “refund” and “payout” operations.

Parameter	Type	Description	Example
client_login	String	User login. It is the same as for Payment Manager	demo
client_password	String(64)	SHA-256 HEX-encoded digest generated from the same password used in Payment Manager	2a97516c35…
id	Integer	ID of Order to be changed	299150
status_to	"capture" or "void" or "refund" or "payout"	Action to be performed	void
amount	Decimal	Amount of refund or payout. May be omitted for refund, all remaining amount is refunded then	21.12
reason	String	Description for refund or payout	Out of stock

XML Response

Order status response example:

```
 Content-Type: text/xml;charset=UTF-8

<response is_executed="yes" refund_id="299151">
   <order id="299150" status_to="refund" currency="USD" refund_amount="42.38" remaining_amount="132.54"
status="APPROVED" />
</response>
```

Reply comes as XML content in the HTTP Response body.

Element	Attribute	Type	Description	Example
response	**is_executed**	"yes" or "no"	Indicates was the request successful or not	yes
	refund_id	String	ID of created Refund in case of "refund" operation	299151
	payout_id	String	ID of created Payout in case of "payout" operation	299151
	details	String	The reason why request was insuccessful	Login Failed
response/**order**		***Order element***	Order change status details	

Order Element Structure

Attribute	Type	Description	Example
id	String	Requested order's ID	Numeric
status_to	"capture" or "void" or "refund" or "payout"	Requested action	void
currency	String(3)	Currency of refunded amount, ***ISO 4217*** currency code	EUR
refund_amount	Decimal	Refunded amount in the currency of processing	21.12
remaining_amount	Decimal	Amount left after refund was made	0.00
status	String	***Status Code*** of modified or created order	APPROVED

Note: when status of refund or payout is PENDING it means that it was not processed yet, the final status will be set later and it can be declined.

REST API

The CardPay API uses HTTP verbs and a RESTful endpoint structure. Request and response payloads are formatted as JSON.

API Endpoints

The API is available at the following URLs:

- **Test URL:** `https://sandbox.cardpay.com/MI/api/v2`
- **Live URL:** `https://cardpay.com/MI/api/v2`

Authentication

Authorization header example

```
GET /MI/api/v2/payments HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
```

If user has login `demo` and password `demo`, the Base64 encoded string of "demo:demo" is: ZGVtbzpkZW1v

To use this service you will need Payment Manager user login and password. This service will affect only orders available for this user to be seen.

This service uses Basic access authentication. To authenticate you need to add `Authorization` header to HTTP request:

```
Authorization: Basic AUTHORIZATION_TOKEN
```

AUTHORIZATION_TOKEN string is a Base64 encoded credentials string. The credentials string format is: `login`:`password`.

JSON Document Format

The request data is sent in the HTTP Request body having the `Content-Type: application/json`.

Unless otherwise noted, objects defined by this specification**SHOULD NOT** contain any additional members. Client and server implementations **MUST** ignore members not recognized by this specification. This allows the API to evolve.

A JSON object MUST be at the root of JSON API request and response containing data. This object defines a document's **"top level".**

A document contain at least one of the following top-level members:

- data : the document's "primary data"
- errors : an array of ***error objects***
- meta : a meta object that contains non-standard meta-information.

A document MAY contain any of these top-level members:

- links : a links object related to the primary data.

The top-level links object MAY contain the following members:

- self : the link that generated the current response document.
- related : a related resource link when the primary data represents a resource relationship. pagination links for the primary data.

The document's "primary data" is a representation of the resource or collection of resources targeted by a request.

Primary data MUST be either:

- a single resource object, a single resource identifier object, or null, for requests that target single resources
- an array of resource objects, an array of resource identifier objects, or an empty array ([]), for requests that target resource collections

A logical collection of resources MUST be represented as an array, even if it only contains one item or is empty.

"Resource objects" appear in a JSON API document to represent resources.

A resource object SHOULD contain at least the following top-level members:

- id
- type

Exception: The id member is not required when the resource object originates at the client and represents a new resource to be created on the server.

Errors

Processing Errors

A server MAY choose to stop processing as soon as a problem is encountered, or it MAY continue processing and encounter multiple problems. For instance, a server might process multiple attributes and then return multiple validation problems in a single response.

When a server encounters multiple problems for a single request, the most generally applicable HTTP error code SHOULD be used in the response. For instance, 400 Bad Request might be appropriate for multiple 4xx errors or 500 Internal Server Error might be appropriate for multiple 5xx errors.

Error Objects

Error objects provide additional information about problems encountered while performing an operation. Error objects MUST be returned as an array keyed by errors in the top level of a JSON API document.

An error object MAY have the following members:

- id : a unique identifier for this particular occurrence of the problem.
- links : a links object containing the following members:
    - about : a link that leads to further details about this particular occurrence of the problem.
- status : the HTTP status code applicable to this problem, expressed as a string value.
- title : a short, human-readable summary of the problem that SHOULD NOT change from occurrence to occurrence of the problem, except for purposes of localization.
- detail : a human-readable explanation specific to this occurrence of the problem.
- source : an object containing references to the source of the error, optionally including any of the following members:
    - pointer : a JSON Pointer [RFC6901] to the associated entity in the request document [e.g. “/data” for a primary data object, or “/data/attributes/title” for a specific attribute].
    - parameter : a string indicating which URI query parameter caused the error.
- meta : a meta object containing non-standard meta-information about the error.

Payments

The *payments* namespace contains resource collections for payments.

This service is available on following entry points URL:

- **Test URL:** GET: https://sandbox.cardpay.com/MI/api/v2/payments
- **Live URL:** GET: https://cardpay.com/MI/api/v2/payments

Note that only GET method is allowed.

Operation

GET /MI/api/v2/payments

Get the list of payments for a period of time. This service will return only payments available for this user to be seen.

Request

Payments request example

```
GET /MI/api/v2/payments HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
Query String Parameters:
startMillis: 1438981200000
endMillis: 1439381873000
walletId: 123
maxCount: 10
```

See ***Request Attributes*** section bellow. These attributes must be sent as HTTP parameters.

Response

Payments response example

```
{
  "data": [
    {
      "id": "12345",
      "number": "order00017",
      "state": "DECLINED",
      "date": 1438981200400,
      "customerId": "11021",
      "declineReason": "Insufficient funds",
      "declineCode": "13",
      "currency": "USD",
      "amount": 21.12,
      "description": "Special order - REF.ID: 13579",
      "note": "VIP customer",
      "email": "customer@example.com",
      "is3d": false,
      "rrn": "000012345678",
      "maskedPan": "400000...0002",
            "issuingCountryCode": "US"
        },
        {
            "id": "12346",
            "number": "order00018",
            "state": "COMPLETED",
            "date": 1505687851000,
            "currency": "USD",
            "amount": 42.24,
            "description": "Special order - REF.ID: 135790",
            "note": "Platinum customer",
            "email": "customer2@example.com",
            "authCode": "DK3H25",
            "is3d": true,
            "rrn": "000012345679",
            "maskedPan": "400000...0002",
            "issuingCountryCode": "US"
        }
    ],
    "hasMore": false
}
```

Reply comes as JSON content in the HTTP Response body.

Element	Description	Type	Example
data	List of ***payment*** elements	See ***Payment Element Structure***	
hasMore	Indicates if there are more payments for this period than was returned	Boolean	true

Operation

Payment info request example

```
 GET /MI/api/v2/payments/12347 HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
```

Payment info response example

```
{
    "data": {
        "type": "PAYMENTS",
        "id": "12347",
        "created": 1522740708000,
        "updated": 1522740709206,
        "state": "COMPLETED",
        "rrn": "000018872018",
        "merchantOrderId": "955987",
        "description": "Special order - REF.ID: 13579"
    }
}
```

GET /MI/api/v2/payments/:id

Use this call to get the state of the payment by it's **id**.

Operation returns a "data" element containing ***Order Status Object***

Refunds

The *refunds* namespace contains resource collections for refunds.

This service is available on following entry points URL:

- **Test URL:** GET: https://sandbox.cardpay.com/MI/api/v2/refunds
- **Live URL:** GET: https://cardpay.com/MI/api/v2/refunds

Note that only GET method is allowed.

Operation

GET /MI/api/v2/refunds

Get the list of refunds for a period of time. This service will return only refunds available for this user to be seen.

Request

Refunds request example

```
GET /MI/api/v2/refunds HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
Query String Parameters:
startMillis: 1455045922000
endMillis: 1455545922000
walletId: 123
maxCount: 10
```

See ***Request Attributes*** section bellow. These attributes must be sent as HTTP parameters.

Response

Refunds response example

```
{
 "data": [
  {
   "id": "12348",
   "number": "949225",
   "state": "COMPLETED",
   "date": 1455045922100,
   "currency": "EUR",
   "amount": 14.14,
   "description": "Special order - REF.ID: 13579",
   "email": "test1@example.com",
   "authCode": "DK3H25",
   "is3d": true,
   "originalOrderId": "12350",
   "rrn": "12345678",
   "maskedPan": "400000...0002",
   "issuingCountryCode": "US"
  },
    {
      "id": "12349",
      "number": "214250",
      "state": "COMPLETED",
      "date": 1455045922200,
      "currency": "EUR",
      "amount": 8.84,
      "description": "Special order - REF.ID: 13580",
      "email": "test2@example.com",
      "authCode": "dpsncc",
      "is3d": false,
      "originalOrderId": "12351",
      "rrn": "12345678",
      "maskedPan": "400000...0077",
      "issuingCountryCode": "US"
    }
  ],
  "hasMore": false
}
```

Reply comes as JSON content in the HTTP Response body.

Element	Description	Type	Example
data	List of ***refund*** elements	See ***Refund Element Structure***	
hasMore	Indicates if there are more refunds for this period than was returned	Boolean	true

Operation

Refund info request example

```
 GET /MI/api/v2/refunds/12352 HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
```

Refund info response example

```
{
  "data": {
    "type": "REFUNDS",
    "id": "12352",
    "created": 1523540268000,
    "updated": 1523540269247,
    "state": "COMPLETED",
    "rrn": "12345679",
    "merchantOrderId": "890081",
    "description": "Special order - REF.ID: 13581"
  }
}
```

GET /MI/api/v2/refunds/:id

Use this call to get the state of the refund by it's **id**.

Operation returns a "data" element containing ***Order Status Object***

Payouts

The *payouts* namespace contains resource collections for payouts.

This service is available on following entry points URL:

- **Test URL:** POST, GET: https://sandbox.cardpay.com/MI/api/v2/payouts
- **Live URL:** POST, GET: https://cardpay.com/MI/api/v2/payouts

Payout request example:

```
POST /MI/api/v2/payouts HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
Query String Parameters:
walletId: 12345
```

```
{
  "data": {
    "type": "PAYOUTS",
    "timestamp": "2015-08-28T09:09:49Z",
    "merchantOrderId": "PO01242324",
    "amount": 128.97,
    "currency": "USD",
    "card": {
      "number": "4000000000000002",
      "expiryMonth": 7,
      "expiryYear": 2019
    },
    "description": "X-mass gift for you, my friend",
    "note": "Payout Ref.12345",
    "email": "customer@example.com"
  }
}
```

Operation:

```
POST /MI/api/v2/payouts
```

Create Payout Order.

Request

Properties and elements names are case-sensitive.

Properties and elements in **bold** are mandatory. Properties in **bold** are mandatory only if containing element is present.

Properties in ***bold and italic*** are conditional, follow the links for details.

Request attribute walletId must be sent as HTTP parameter.

Parameter	Type	Description	Example
walletId	String	Unique merchant’s ID used by the CardPay payment system	12345

Note that some request fields are not mandatory but still may be required by Bank. Please contact CardPay manager for more info on the requirements.

Request Object

Property	Type	Description	Example
type	String	always "PAYOUTS"	PAYOUTS
timestamp	String	Represents the date and time in ***ISO 8601*** format when the order was sent to CardPay	2015-08-28T09:09:49Z
merchantOrderId	String(50)	Represents the ID of the order in merchant’s system	PO01242324
amount	Decimal	Represents the amount to be transferred to the customer’s card, must be less than 100 millions	128.97
currency	String(3)	***ISO 4217*** currency code of the transaction. Must match wallet currency	USD
card	***card***	Represents a debit or credit card. Required if ‘cardToken’ is not present, see ***Card Token***	
cardToken	String(36)	***Card Token*** used instead of card information	067e6162-3b6f-4ae2-a171-2470b63dff00
billing	***billing***	Represents the customer’s billing address	
description	String(200)	Transaction description	
dynamicDescriptor	String(25)	Short description of the service or product, must be enabled by CardPay manager to be used	
note	String(50)	Note about the order, not shown to the customer	Payout Ref.12345
recipientInfo	String(500)	Payout recipient information, see ***Recipient Information***	John Smith
email	String(256)	Customer’s e-mail address, see ***Recipient Information***	customer@example.com

Card Object

Property	Type	Description
number	int(13-19)	Customer’s card number (PAN). Any valid card number, may contain spaces
expiryMonth	int(2)	Customer’s card expiration month in format: mm , contact CardPay manager, this field may be required by Bank
expiryYear	int(4)	Customer’s card expiration year in format: yyyy , contact CardPay manager, this field may be required by Bank

Both expiryMonth and expiryYear must be present or absent at the same time.

Billing Object

Property	Type	Description
country	String(3)	***ISO 3166-1*** code of delivery country: 2 or 3 latin letters or numeric code
state	String(20)	Billing state or province
zip	String(12)	Billing postal code
city	String(20)	Billing city
street	String(2-100)	Billing street address
phone	String(5-20)	Customer phone number

Billing is the customer’s billing address. Billing may be required by Bank, contact CardPay manager for details.

Card Token

Payout request with Card Token example:

```
POST /MI/api/v2/payouts HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
Query String Parameters:
walletId: 12345
```

```
{
  "data": {
    "type": "PAYOUTS",
    "timestamp": "2015-08-28T09:09:49Z",
    "merchantOrderId": "PO01242324",
    "amount": 128.97,
    "currency": "USD",
    "cardToken": "067e6162-3b6f-4ae2-a171-2470b63dff00",
    "description": "X-mass gift for you, my friend",
    "note": "Payout Ref.12345",
    "email": "customer@example.com"
    }
  }
```

You can obtain Card Token by sending Payment with the attribute 'generate_card_token' having value *"true"* in it. Payout then can be made to the same card used for Payment. To do that you need to send property 'cardToken' instead of 'card' and 'recipientInfo'. Either 'cardToken' or 'card' property must be present, not both at the same time. Card Token cannot be used more than 1 year after token was generated.

Recipient Information

Property 'recipientInfo' may be required by Bank. In most cases it's Cardholder's name, contact CardPay manager for requirements. It must be omitted when property 'cardToken' is present. Property 'email' may also be required by Bank, contact CardPay manager.

Response

Payout response example

```
{
  "data": {
    "type": "PAYOUTS",
    "id": "12353",
    "created": 1523605624890,
    "updated": 1523605625142,
    "state": "COMPLETED",
    "rrn": "000018872019",
    "merchantOrderId": "PO01242324",
    "description": "Special order - REF.ID: 13579"
  },
  "links": {
    "self": "https://sandbox.cardpay.com/MI/api/v2/payouts/12353"
  },
  "meta": {
    "request": {
      "timestamp": 1523610589000,
      "type": "PAYOUTS",
      "merchantOrderId": "PO01242324",
      "amount": 128.97,
      "currency": "USD",
      "note": "Payout Ref.12345",
      "description": "X-mass gift for you, my friend",
      "email": "customer@example.com",
      "card": {
        "number": "400000...0002",
        "expiryMonth": 7,
        "expiryYear": 2019
      }
    }
  }
}
```

Property	Description	Type
data	Contains the payout order details	***Order Status Object***. Property “type” is always "PAYOUTS".
links	Contains the link to payout order	***links*** element structure below
meta	May contain extra information, e.g. ***payout request***	object

The server reply comes as JSON content in the HTTP Response body and HTTP status code 201 Created. Additionally, server will return the URL of the created payout order in the Location HTTP header and in "links.self" JSON property of the response object.

The "data" property will contain ***Order Status*** with the ***State*** of the operation. If the payout operation was **successful**, the state will be "COMPLETED". If it **failed**, the state will be "DECLINED". In case if the payout was **accepted**, but not performed yet, the state will be "AUTHORIZED". In this case the final result will be sent in a ***Callback*** or can be checked a few minutes later using GET method of this API.

The "meta" property may contain additional information, e.g. original request with masked card number.

Links Object

Element	Attribute	Type	Description
links	self	String	A link to the newly created Payout order

Operation

```
GET /MI/api/v2/payouts
```

Get the list of payouts for a period of time. This service will return only payouts available for this user to be seen.

Request

Payouts request example

```
GET /MI/api/v2/payouts HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
Authorization: Basic ZGVtbzpkZW1v
Query String Parameters:
startMillis: 1438980200000
endMillis: 1438981200000
walletId: 123
maxCount: 10
```

See ***Request Attributes*** section bellow. These attributes must be sent as HTTP parameters.

Response

Payouts response example

```
{
 "data": [
  {
   "id": "2200421",
   "number": "ORD437282",
   "state": "DECLINED",
   "date": 1438980200010,
   "declineReason": "Incorrect card data",
   "declineCode": "15",
   "currency": "EUR",
   "amount": 12.34,
   "description": "X-mass gift for you, my friend",
   "note": "Some note",
   "email": "customer@example.com",
   "is3d": false,
   "rrn": "9a28135e-61be-2a97-gac5-54253da2fb2a",
   "maskedPan": "400000...0002",
   "issuingCountryCode": "US"
  },
  {
   "id": "2200422",
   "number": "ORD437283",
   "state": "COMPLETED",
   "date": 1438980200020,
   "currency": "EUR",
   "amount": 56.78,
   "description": "Special order - REF.ID: 13579",
   "note": "Some note 2",
   "email": "customer@example.com",
   "authCode": "dpsncc",
   "is3d": false,
   "rrn": "9a28135e-21be-2397-4ac5-55256d72abea",
   "maskedPan": "400000...0002",
   "issuingCountryCode": "US"
  }
 ],
 "hasMore": false
}
```

Reply comes as JSON content in the HTTP Response body.

Element	Description	Type	Example
data	List of ***payout*** elements	See ***Payout Element Structure***	
hasMore	Indicates if there are more payouts for this period than was returned	Boolean	true

Operation

Payout info request example

```
GET /MI/api/v2/payouts/12354 HTTP/1.1
Host: sandbox.cardpay.com:443
Content-Type: application/json
```

```
Authorization: Basic ZGVtbzpkZW1v
```

Payout info response example

```
{
   "data": {
      "type": "PAYOUTS",
      "id": "12354",
      "created": 1523605624000,
      "updated": 1523605625142,
      "state": "COMPLETED",
      "rrn": "000018872019",
      "merchantOrderId": "PO01242324",
      "description": "Special order - REF.ID: 13579"
   }
}
```

GET /MI/api/v2/payouts/:id

Use this call to get the status of the specified payout by it's**id**.

Operation returns a "data" element containing ***Order Status Object***

Common Objects

Order Report Request Attributes

Parameter	Type	Description	Example
startMillis	Long	Epoch time in milliseconds when requested period starts (inclusive), default is 24 hours before endMillis	1438336800000
endMillis	Long	Epoch time in milliseconds when requested period ends (not inclusive), must be less than 7 days after startMillis, default is current time	1438941600000
number	String	Get list of orders by Order Number	order00017
walletId	Integer	Limit result with single WebSite orders	101
maxCount	Integer	Limit number of returned orders, must be less than 10,000, default is 1,000	100

Order Report Response Object

Attribute	Description	Type	Example
id	ID assigned to the order in CardPay	String	299150
number	Merchant’s ID of the order	String	order00017
state	***Order State***	String	COMPLETED
date	Epoch time when this order started	Long	1438336812000
customerId	Customer’s ID in the merchant’s system	String	11021
declineReason	Bank’s message about order’s decline reason	String	Cancelled by customer
declineCode	***Code of the decline***	String	02
currency	Transaction currency	String(3)	USD
amount	Initial order amount	Decimal	21.12
description	Order description. In case of refunds it contains refund reason	String	Special order - ID: 13579
note	Note about the order	String	VIP customer
email	Customer’s e-mail address	String	customer@example.com
authCode	Authorization code, provided by bank	String	DK3H25
is3d	Was 3-D Secure authentication made or not	Boolean	true
refundedAmount	Only for payments: refund amount in order’s currency	Decimal	7.04
originalOrderId	Only for refunds: CardPay ID of original payment order	String	299151
rrn	RRN (Retrieval Reference Number), supplied by the acquiring financial institution	String	000012345678
maskedPan	Masked PAN (shows first 6 digits and 4 last digits)	String	400000…0002
issuingCountryCode	Country code of issuing country	String	US

Order Status Object

Successful Order state example

```
{
    "type": "PAYMENTS",
    "id": "12355",
    "created": 1522738766000,
    "updated": 1522740001760,
    "state": "COMPLETED",
    "rrn": "000018872018",
    "merchantOrderId": "1",
    "description": "Special order - REF.ID: 13579"
}
```

Failed Order state example

```
{
    "type": "PAYMENTS",
    "id": "12356",
    "created": 1523333446000,
    "updated": 1523333469383,
    "state": "DECLINED",
    "decline": {
        "code": "10",
        "description": "Declined by bank (not classified)"
    },
    "merchantOrderId": "848868"
}
```

Property	Type	Description
type	String	Order type
id	String	ID of the newly created Payment order in CardPay system
created	String	Payment order creation date, timestamp
updated	String	Payment order last update date, timestamp
state	***Order State***	Payment order state in CardPay system.
decline	Object	If order processing was not successful, the reason of decline will be provided
decline.code	String	CardPay ***decline code***
decline.description	String	Description of the CardPay ***decline code***
rrn	String(40)	This value will be set to the *Retrieval Reference Number (RRN)* supplied by the acquiring financial institution. This number can be provided to the cardholder as a receipt reference number for a successful transaction.
merchantOrderId	String(50)	Represents the ID of the order in merchant’s system
description	String	Order description. In case of refunds it contains refund reason

Decline code and description will be present only when processing was not successful.

Order States

State field of the order may have one of the following values:

State	Description
NEW	Order was submitted successfully to CardPay system and transaction was created in CardPay system
IN_PROGRESS	Transaction is being processed
AUTHORIZED	Transaction was successfully authorized but needs some time to be verified, amount was held and can be captured later
DECLINED	Transaction was rejected
COMPLETED	Transaction was successfully completed, amount was captured (for payments)
CANCELLED	Transaction was cancelled by customer
REFUNDED	Transaction was fully refunded
VOIDED	Transaction was voided
CHARGED_BACK	Customer’s chargeback claim was received
CHARGEBACK_RESOLVED	Customer’s chargeback claim was rejected, equals to COMPLETED

Changelog

2019-08-14 - 2.70

- 3-D Secure 2 API changes added

2018-04-18 - 2.61

- Added `Flights` tag for airlines
- Email and card expiration date now can be returned in a Callback

2018-03-14 - 2.60

- Unused decline code 12 was removed
- Card Token expiration condition changed

2017-03-16 - 2.59

- “Dynamic descriptor” parameter was added

2016-12-05 - 2.57

- [XML API] `/service/order-change-status` now returns status and ID of created order

2016-11-03 - 2.56

- Japanese locale was added

2016-09-05 - 2.54

- Examples were updated

2016-08-03 - 2.53

- Maximum amount was increased to 100,000,000 for all operations

2016-07-01 - 2.52

- “Card Token” feature was added
- “Authentication Request” feature was added

2016-05-31 - 2.49

- Order state “CHARGEBACK RESOLVED” was added

2016-05-05 - 2.48

- Order state “CLAIMED” was removed from ***Order States*** as no longer supported

2016-04-26 - 2.47

- [REST API] is now enabled by default

2016-03-29 - 2.44

- [XML API] `/service/order-report` service is no longer supported

2016-03-10 - 2.43

- Added description of environments and the list of PANs for Sandbox
- `External Payment Page` mode was renamed to `Gateway` mode
- [REST API] is now disabled by default, it can be enabled by CardPay manager
- [REST API] getting the list of orders is described for all order types
- Quotation marks are allowed in `street` field of Order
- Customer IP was added to Callback
- [REST API] report parameters are not mandatory now

2015-12-23 - 2.42

- [REST API] parameter `number` was added to GET request
- [REST API] response descriptions and examples were fixed

2015-11-23 - 2.41

- Callback section was updated

2015-10-15 - 2.40

- [XML API] `/service/order-change-status` service documentation updated
- [REST API] added operation `/api/v2/payments`
- [REST API] added operation `/api/v2/refunds`
- [REST API] added operation `/api/v2/refunds/:id`

2015-09-16 - 2.39

- [REST API] documentation structure updated
- [REST API] added operation: `/api/v2/payments/:id`
- Added orderXML attributes recurring_begin and recurring_id
- Added Payment Result attribute recurring_id

2015-08-12 - 2.38

- [REST API] added operation `/api/v2/payments`
- Callback section was updated

2015-05-15 - 2.35

- List of Decline Codes was fixed

2015-04-21 - 2.33

- Order’s attribute customer_id was added

2015-03-31 - 2.32

- Decline code 07 added (‘Limit reached’)

2015-03-24 - 2.30

- API documentation updated