dotpay®

Cards

Release 1.2

Dotpay Development Team

September 23, 2020

Contents

Document describes credit card payment integration using direct communication with Dotpay via REST API.

This documentation is available online at: https://www.dotpay.pl/developer/doc/credit-cards/

1 Service address

The service is available on the following addresses:

- for test environment
  https://ssl.dotpay.pl/test_payment/payment_api/v1/
- for production environment
  https://ssl.dotpay.pl/t2/payment_api/v1/

2 Resources

POST /register_order/

This method allows to create a payment operation in Dotpay system on any payment channel. Examples below show payment registration on credit cards channel.

Exemplary request:

```
curl --user login:passwd \
     -H'Accept: application/json; indent=4' \
     -H'content-type: application/json' \
     -XPOST \
     -d @request.json \
     https://ssl.dotpay.pl/test_payment/payment_api/v1/register_order/
```

Status Codes

- 201 Created – created
- 400 Bad Request – error while processing the request

2.1 Basic parameters in the register order method input.

The structure of the data transferred as input to the register order method is described in the table below.

2.1.1 Table 1. Basic parameters of the register order method input

Element	Type	Comments
`order`	object	mandatory; order data
`order.amount`	decimal(10,2)	mandatory; order amount
`order.currency`	string	mandatory; three letter code (ISO 4217) of order currency
`order.description`	string	mandatory; order description
`order.control`	string	optional; order id on seller's side
`seller`	object	mandatory; seller account data
`seller.account_id`	integer	mandatory; Dotpay account number
`seller.url`	string	mandatory; the address to which the payer may be redirected after making the payment
`seller.urlc`	string	optional; the address where notifications about operation status will be sent
`payer`	object	mandatory; payer's data
`payer.first_name`	string	mandatory; payer's first name
`payer.last_name`	string	mandatory; payer's last name
`payer.email`	string	wymagane; payer's email address
`payer.address`	object	optional (unless the configuration of a given channel requires these data); address detail of the payer
`payer.address.street`	string	mandatory if `payer.address` is given; street
`payer.address.building_number`	string	mandatory if `payer.address` is given; building number
`payer.address.flat_number`	string	mandatory if `payer.address` is given; flat number
`payer.address.postcode`	string	mandatory if `payer.address` is given; post code
`payer.address.city`	string	mandatory if `payer.address` is given; city
`payer.address.country`	string	mandatory if `payer.address` is given; three-letter code (ISO 3166-1 alpha-3) of a country
`payment_method`	object	mandatory; payment method data
`payment_method.channel_id`	integer	mandatory; payment channel number, 248 for credit cards. Full list of payment channels is available in basic Implementation documentation
`payment_method.credit_card`	object	credit card data
`payment_method.credit_card.number`	string	credit card number
`payment_method.credit_card.expiration_date`	object	credit card expiration date
`payment_method.credit_card.expiration_date.year`	string (YYYY)	credit card expiration date year
`payment_method.credit_card.expiration_date.month`	string (MM)	credit card expiration month
`payment_method.credit_card.security_code`	string	CVV2/CVC2 code
`payment_method.credit_card.store`	boolean	store credit card data in Dotpay agreement
`payment_method.credit_card.customer_id`	string (4 - 1024 characters)	unique buyer ID generated and stored by seller's system, required for future payments
`payment_method.credit_card.id`	string	Buyer's registered card ID

continues on next page

Table 1 – continued from previous page

Element	Type	Comments
`payment_method.credit_card.operation_type`	string	operation type: e_commerce – first and consecutive payment in one-click model (default value), recurring_init – first transaction allowing later use of recurring payments, recurring – recurring payment (customer doesn't have to be present in order to charge the registered card),
`payment_method.credit_card.security_code_required`	string	allows to control whether CVV/CVV2 security code is required during payment, applies only to consecutive e_commerce. Available values: yes (default) no
`payment_method.credit_card.threeds`	string	allows to control whether 3-D Secure authentication code is required during payment. Applies only to e_commerce model for enrolled cards. Available values: yes (default) no
`request_context.ip`	string	mandatory; payer's ip address
`request_context.language`	string	two-letter code of a language (ISO 639-1) in which the payment is made; yes (default)

2.2 Parameters for 3-D Secure v2 support on the register_order method input

Sending more data than just "required" for a card payment may be of great importance in the final decision of the card issuer to accept the transaction itself.

Note: Based on the additional information sent or the lack thereof, the card issuer may decide on a possible need for additional transaction verification (challenge) or to process transactions without the 3DS code. This, in turn, may speed up and facilitate the payment process itself for the payer and, consequently, have a positive effect on the conversion of card payments.

Therefore, we recommend that you send as much additional data as possible when initiating the payment.

Input data of register_order method for **3DS v2** support are described by the following tables.

2.2.1 Table 2. Parameters in the register_order method input for 3DS v2 support describing the payer's browser

Element	Type	Comments
`request_context.accept`	string	recommended; Accept header from client browser headers description: HTTP ACCEPT Example: `request_context.accept` = text/html, application/xhtml+xml, application/xml;q=0.9, */
`request_context.referer`	string	recommended; Adres strony z której użytkownik został przekierowany (nagłówek HTTP) description: HTTP referer Example: `request_context.referer` = http://www.example.org/referring_page
`request_context.useragent`	string	zalecane; Nagłówek user-agent z nagłówków przeglądarki klienta description: HTTP User-Agent Example: `request_context.useragent` = Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
`request_context.browser.javaenabled`	boolean	recommended; The ability to execute Java code in the client's browser description: navigator.javaEnabled(); Example: `request_context.browser.javaenabled` = 1
`request_context.browser.javascriptenabled`	boolean	recommended; The ability to execute JavaScript code in the client's browser Example: `request_context.browser.javascriptenabled` = 1
`request_context.browser.language`	string	required where `request_context.browser.javascriptenabled` = 1 Browser language in the IETF BCP 47 standard descriptiondescription: navigator.language.slice(0,2) Example: `request_context.browser.language` = pl
`request_context.browser.screencolordepth`	int	required where `request_context.browser.javascriptenabled` = 1 Głębia koloru dla wyświetlania koloru w przeglądarce klienta, pozyskana z screen.colorDepth. description: screen.colorDepth permissible values: *1,4,8,15,16,24,32,48* Example: `request_context.browser.screencolordepth` = 24
`request_context.browser.screenheight`	int	required where `request_context.browser.javascriptenabled` = 1 Screen height in pixels obtained from screen.height. description: screen.height Example: `request_context.browser.screenheight` = 1080

continues on next page

Table 2 – continued from previous page

Element	Type	Comments
`request_context.browser.screenwidth`	int	required where `request_context.browser.javascriptenabled` = 1 Screen width in pixels obtained from screen.width. description: screen.width Example: `request_context.browser.screenwidth` = 1920
`request_context.browser.timezone`	int	required where `request_context.browser.javascriptenabled` = 1 Time zone expressed as the difference in minutes between GMT and local time description: new Date().getTimezoneOffset() Example: `request_context.browser.timezone` = -120

2.2.2 Table 3. Handling of shipping and payer data on the input of register_order method for 3DS v2 support

FIELD NAME	TYPE	DESCRIPTION
`payment_method.customer.is_logged_in`	boolean	Whether payer has register an account before placing an order
`payment_method.customer.registered_since`	string	Payer's registration date on the seller's website, format *YYYY-MM-DD* or *YYYY-MM-DD hh:mm:ss* Optional field, if it is sent, the parameter: `payment_method.customer.order_count` should also be sent. Instead of specifying a specific date in the format *YYYY-MM-DD*, you can use the parameter: `payment_method.customer.registered_since_indicator` instead.
`payment_method.customer.registered_since_indicator`	string (*indicator*)	Payer's registration date on the seller's website, indicator for the `payment_method.customer.registered_since` field Optional, if it's sent, `payment_method.customer.order_count` is also required
`payment_method.customer.account_update`	string	Date of last change of paying account on the seller's website, format *YYYY-MM-DD* Instead of specifying a specific date in the format *YYYY-MM-DD*, you can use the parameter: `payment_method.customer.account_update_indicator` instead.
`payment_method.customer.account_update_indicator`	string (*indicator*)	Date of last change of paying account on the seller's website, indicator for the field `payment_method.customer.account_update`
`payment_method.customer.password_change`	string	Date of last password change for the paying account on the seller's website, format *YYYY-MM-DD* Instead of specifying a specific date in the format *YYYY-MM-DD*, you can use the parameter: `payment_method.customer.password_change_indicator` instead.
`payment_method.customer.password_change_indicator`	string (*indicator*)	Date of last change of password for the paying account on the seller's website, indicator for the field `payment_method.customer.password_change`

continues on next page

Table 3 – continued from previous page

FIELD NAME	TYPE	DESCRIPTION
`payment_method.customer.shipping_address_since`	string	Date from when the payer's delivery address is used, format *YYYY-MM-DD* Instead of specifying a specific date in the format *YYYY-MM-DD*, you can use the parameter: `payment_method.customer.shipping_address_since_indicator` instead.
`payment_method.customer.shipping_address_since_indicator`	string (*indicator*)	Date from which the payer's delivery address is used, the indicator for the field `payment_method.customer.shipping_address_since`
`payment_method.customer.order_count`	int	Number of orders placed by the paying seller on the seller's website from the date of registration Optional, if it's sent, `payment_method.customer.registered_since` is also required
`payment_method.customer.order_count_day`	int	The number of orders placed by the paying seller on the same day
`payment_method.customer.order_count_year`	int	Number of orders placed by the paying seller in the same year
`payment_method.customer.fraud_activity`	boolean	Has the store ever seen suspicious activity on this buyer's account
`payment_method.customer.order`	-	Order
`payment_method.customer.order.total_amount`	string	The value of the entire order
`payment_method.customer.order.id`	string	Order ID in the seller's system. Corresponds to the ID number of the entire order in the store database
`payment_method.customer.order.delivery_type`	string	Delivery method Available values: • COURIER - courier • POCZTA_POLSKA - Poczta Polska • PICKUP_POINT - pickup point like UPS Access point, DHL Parcel Shop • PACZKOMAT - parcel locker • PACZKA_W_RUCHU - paczka w ruchu • PICKUP_SHOP - pickup in shop (click&collect)
`payment_method.customer.order.delivery_address`	-	Delivery address **If the package is delivered to a point / parcel locker / etc, such address and name should be given, not the data of the actual recipient.**
`payment_method.customer.order.delivery_address.city`	string	Delivery address: city
`payment_method.customer.order.delivery_address.street`	string	Delivery address: street
`payment_method.customer.order.delivery_address.building_number`	string	Delivery address: building number
`payment_method.customer.order.delivery_address.flat_number`	string	Delivery address: flat number

continues on next page

Table 3 – continued from previous page

FIELD NAME	TYPE	DESCRIPTION
`payment_method.customer.order.delivery_address.postcode`	string	Delivery address: zip code
`payment_method.customer.order.delivery_address.country`	string	Delivery address: (ISO 3166-1 alpha2) or (ISO 3166-1 alpha3) country code
`payment_method.customer.order.delivery_address.name`	string	Name of recipient / collection point. Examples: `payment_method.customer.order.delivery_address.name` = Point PP:6252652 `payment_method.customer.order.delivery_address.name` = PPP:6252652
`payment_method.customer.order.delivery_address.phone`	string	Payer phone number
`payment_method.customer.order.delivery_address.is_verified`	bool	Delivery address: Whether the delivery address is verified

---

Note: If the store does not want to provide the correct date, it is possible to use an indicator field of replacement type for selected parameters.

2.2.3 Values used for indicator field replacement for selected fields:

VALUE	DESCRIPTION
01	The payer's account does not exist on the seller's website
02	Date of the transaction just ordered
03	Date not older than 30 days ago
04	Date in the range 30 - 60 days ago
05	Date older than 60 days ago

2.2.4 Sample requests for 3DS v2

Exemplary use of parameters described above:

Listing 1: Example 1: using the minimum number of parameters for the 3DS v2 process (json format)

```
{
    "order": {
        "amount": "34.00",
        "currency": "PLN",
        "description": "Payment for order no 3342",
        "control": "xcftg-32432-5325hdf"
    },
    "seller": {
        "account_id": "123456",
```

(continues on next page)

(continued from previous page)

```json
        "url": "https://www.example.com"
    },
    "payer": {
        "first_name": "John",
        "last_name": "Doe",
        "email": "johndoemail@example.com",
        "phone": "123456789",
        "address": {
            "city": "Warszawa",
            "street": "Krucza",
            "building_number": "1a",
            "flat_number": "4",
            "postcode": "00-950",
            "country": "PL"
        }
    },
    "payment_method": {
        "channel_id": "248",
        "credit_card": {
            "number": "4929532027887670",
            "expiration_date": {
                "year": "2022",
                "month": "01"
            },
            "security_code": "670",
            "store": "1",
            "customer_id": "f9c6a4-25473-765gh"
        }
    },
    "request_context": {
        "ip": "127.0.0.1",
        "language": "pl",
        "accept": "text/html, application/xhtml+xml, application/xml;q=0.9, */",
        "referer": "http://www.example.org/referring_page",
        "useragent": "Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML,␣
↪like Gecko) Chrome/84.0.4147.105 Safari/537.36",
        "browser": {
            "javaenabled": 1,
            "javascriptenabled": 1,
            "language": "en",
            "screencolordepth": 24,
            "screenheight": 1024,
            "screenwidth": 1920,
            "timezone": -120
        }
    }

}
```

Listing 2: Example 2: using additional parameters for the 3DS v2 process - one-click payment with a previously saved card (json format)

```json
{
    "order": {
        "amount": "56.20",
        "currency": "PLN",
        "description": "Payment for order no 6542",
        "control": "3426hs5fskdbg6g"
    },
```

(continues on next page)

(continued from previous page)

```
    "seller": {
        "account_id": "123456",
        "url": "https://www.example.com"
    },
    "payment_method": {
        "channel_id": "248",
        "credit_card": {
            "id":
↪"85c14e6e5608cbc69e19acec41730d59052fbcd306364d96c9cdaafacb7a0833d0fa14280ab9a2b2381fad381f65f0
↪",
            "customer_id": "f9c6a4-25473-765gh"
        },

        "customer": {

            "is_logged_in": 1,
            "registered_since": "2019-11-21",
            "order_count": 23,

            "order": {
                "id": "54356723",
                "delivery_type": "PICKUP_POINT",
                "delivery_address": {
                    "name": "Point PP:6252652",
                    "phone": "+48987654321",
                    "street": "Zielona",
                    "building_number": "32",
                    "postcode": "61-321",
                    "city": "Konin",
                    "country": "PL",
                    "is_verified": 1
                }
            },
            "payer": {
                "first_name": "Wieslaw",
                "last_name": "Nowak",
                "email": "paysdfds@example.com",
                "phone": "+48443456766"
            }
        }

    },
    "payer": {
        "first_name": "Adam",
        "last_name": "Kowal",
        "email": "payeremail@example.com",
        "phone": "+48123456789",
        "address": {
            "city": "Konin",
            "street": "Prosta",
            "building_number": "34",
            "flat_number": "7",
            "postcode": "62-500",
            "country": "PL"
        }
    },
    "request_context": {
        "ip": "192.188.3.221",
        "language": "pl",
        "accept": "text/html, application/xhtml+xml, application/xml;q=0.9, */",
        "referer": "http://www.example.org/referring_page",
```

(continues on next page)

(continued from previous page)

```
        "useragent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36␣
↪(KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393",
        "browser": {
            "javaenabled": 1,
            "javascriptenabled": 1,
            "language": "en",
            "screencolordepth": 24,
            "screenheight": 1024,
            "screenwidth": 1920,
            "timezone": -120
        }
    }

}
```

3 One-Click payment

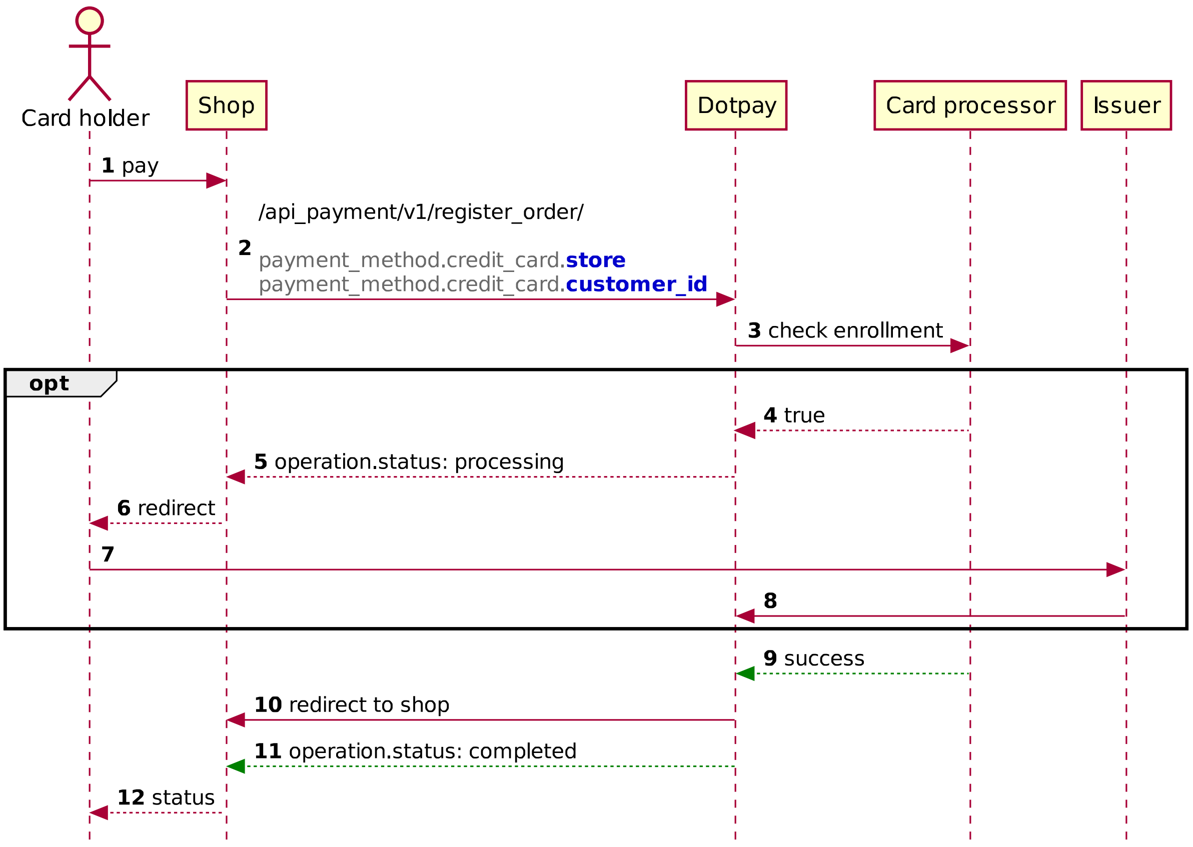
3.1 One Click assumptions

This section describes exemplary credit card (direct and indirect) registration process in One Click model, and consecutive payments where shop passes registered card id.

Shop can send request only when customer has authenticated in shop's system (has to be logged in).

Caution: Keep in mind cards are registered in context of given shop (`id`) group in Dotpay and won't work for other accounts.

3.2 First One Click payment process


Card holder
Shop
Dotpay
Card processor
Issuer
1 pay
/api_payment/v1/register_order/
2
payment_method.credit_card.store
payment_method.credit_card.customer_id
3 check enrollment
opt
4 true
5 operation.status: processing
6 redirect
7
8
9 success
10 redirect to shop
11 operation.status: completed
12 status

Below are examples of first payment initialization in each model:

3.2.1 Direct registration

POST /cards/

```
{
        "seller": {
                "account_id": "123456",
                "url": "https://www.example.com"
        },
        "payer": {
                "first_name": "John",
                "last_name": "Doe",
                "email": "johndoemail@example.com"
        },
        "credit_card": {
                "number": "4929532027887670",
                "expiration_date": {
                        "year": "2020",
                        "month": "01"
                },
                "security_code": "670",
                "customer_id": "f9c6a4-25473"
        },
        "request_context": {
                "ip": "127.0.0.1",
                "language": "pl"
        }
}
```

3.2.2 Registration with payment

`POST /register_order/`

```
{
        "order": {
                "amount": "1.00",
                "currency": "PLN",
                "description": "test",
                "control": "test"
        },
        "seller": {
                "account_id": "123456",
                "url": "https://www.example.com"
        },
        "payer": {
                "first_name": "John",
                "last_name": "Doe",
                "email": "johndoemail@example.com"
        },
        "payment_method": {
                "channel_id": "248",
                "credit_card": {
                        "number": "4929532027887670",
                        "expiration_date": {
                                "year": "2020",
                                "month": "01"
                        },
                        "security_code": "670",
                        "store": "1",
                        "customer_id": "f9c6a4-25473"
                },
                "request_context": {
                        "ip": "127.0.0.1",
                        "language": "pl"
                }
        }
}
```

3.3 First One click payment description

Note: Processing payment card data by seller's system requires – according to Payment Card Industry Data Security Standard (PCI DSS) – additional conditions to be met. In order to receive more information about necessary formalities please contact Sales Department (handlowy@dotpay.pl).

As an alternative card can also be registered using redirection to Dotpay where customer can safely enter card data. This process has been described in technical manual of payment integration

---

Description below applies to registration with payment. In direct registration process is identical but instead of charging the customer only a temporary funds blockade be issued, cancelled when registration process is completed. Operation type will also change from payment to credit_card_registration.

1. Customer chooses payment with credit card, enters it's data and click pay.
2. Shop initializes payment process in Dotpay passing order details such as card data and parameters required for its registration:

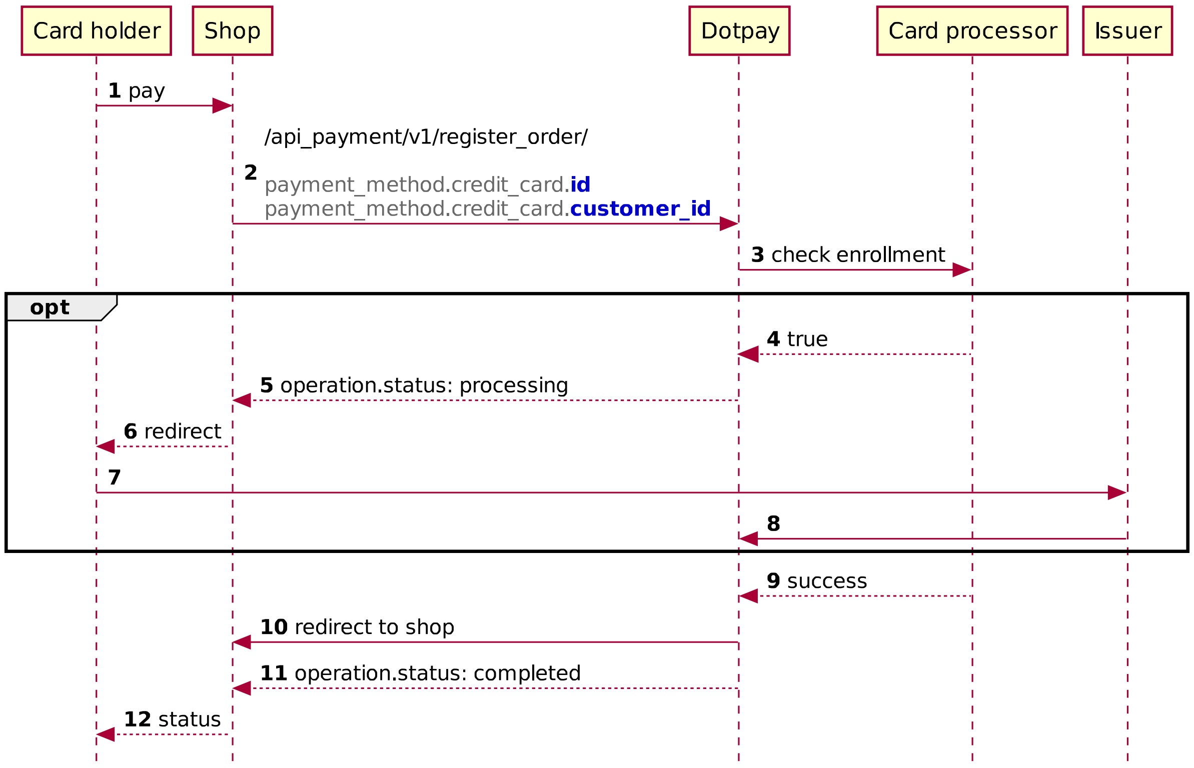
```
{
        "order": {
                "amount": "1.00",
                "currency": "PLN",
                "description": "test",
                "control": "test"
        },
        "seller": {
                "account_id": "123456",
                "url": "https://www.example.com"
        },
        "payer": {
                "first_name": "John",
                "last_name": "Doe",
                "email": "johndoemail@example.com"
        },
        "payment_method": {
                "channel_id": "248",
                "credit_card": {
                        "number": "4929532027887670",
                        "expiration_date": {
                                "year": "2020",
                                "month": "01"
                        },
                        "security_code": "670",
                        "store": "1",
                        "customer_id": "f9c6a4-25473"
                },
                "request_context": {
                        "ip": "127.0.0.1",
                        "language": "pl"
                }
        }
}
```

3. Dotpay checks if card is enrolled for 3-D Secure program.

Attention: Steps 4-8 are optional if card is enrolled for 3-D Secure program (description in *Rozdziale 6*).

4. If it is,
5. Dotpay returns operation details including `redirect` section and `redirect_simplified_url` address.
6. Shop is responsible for redirecting customer to the issuer directly using `redirect` section or indirectly via Dotpay using `redirect_simplified_url`.
7. Customer goes to the issuer site and authorizes with 3-D Secure.
8. Issuer redirects customer to Dotpay
9. Card is charged and registered
10. Customer is redirected to the shop.
11. After receiving urlc notification with operation status
12. shop informs customer about order status.

3.4 Consecutive One Click payment process


Card holder
Shop
Dotpay
Card processor
Issuer
1 pay
/api_payment/v1/register_order/
2
payment_method.credit_card.id
payment_method.credit_card.customer_id
3 check enrollment
opt
4 true
5 operation.status: processing
6 redirect
7
8
9 success
10 redirect to shop
11 operation.status: completed
12 status

3.5 Consecutive One Click payment description

1. Customer chooses payment method, picks registered card and clicks pay.
2. Shop initializes payment process sending order data including registered card id and `customer_id`

```
{
        "order": {
                "amount": "1.00",
                "currency": "PLN",
                "description": "test",
                "control": "test"
        },
        "seller": {
                "account_id": "123456",
                "url": "https://www.example.com"
        },
        "payer": {
                "first_name": "John",
                "last_name": "Doe",
                "email": "johndoemail@example.com"
        },
        "payment_method": {
                "channel_id": "248",
                "credit_card": {
                        "id":
↪"85c14e6e5608cbc69e19acec41730d59052fbcd306364d96c9cdaafacb7a0833d0fa14280ab9a2b2381fad381f65f0
↪",
                        "customer_id": "f9c6a4-25473"
                }
        },
```

(continues on next page)

(continued from previous page)

```
        "request_context": {
                "ip": "127.0.0.1",
                "language": "pl"
        }
}
```

3. Dotpay checks if card is enrolled for 3-D Secure program.

Attention: Steps 4-8 are optional if card is enrolled for 3-D Secure program.

4. If it is,
5. Dotpay returns operation details including `redirect` section and `redirect_simplified_url` address.
6. Shop is responsible for redirecting customer to the issuer directly using `redirect` section or indirectly via Dotpay using `redirect_simplified_url`.
7. Customer goes to the issuer site and authorizes with 3-D Secure.
8. Issuer redirects customer to Dotpay
9. Card is charged.
10. Customer is redirected to the shop.
11. After receiving urlc notification with operation status
12. shop informs customer about order status.

4 Recurring payments

4.1 Recurring payments - Assumptions

This section describes exemplary credit card (direct and indirect) registration process in Recurring model, and consecutive payments where shop initializes payments without customer's presence using previously registered card id.

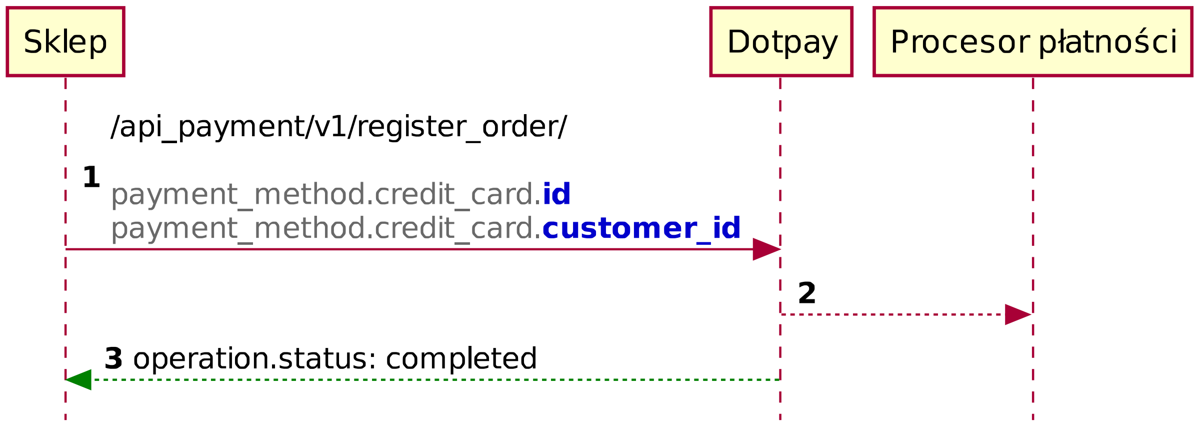
Caution: Keep in mind cards are registered in context of given shop (`id`) group in Dotpay and won't work for other accounts.

4.2 First Recurring payment process

Process is the same as for first One click payment. Only (depending on the account configuration) additional parameter `payment_method.credit_card.operation_type` = recurring_init has to be sent.

Caution: Successful registration does not guarantee consecutive payment attempts will be completed. Customer can unregister card anytime or transaction might fail because of inadequate balance, daily limits, negative authorization and so on.

4.3 Consecutive Recurring payment process


Sklep
Dotpay
Procesor płatności
/api_payment/v1/register_order/
1
payment_method.credit_card.id
payment_method.credit_card.customer_id
2
3 operation.status: completed

4.4 Consecutive Recurring payment process description

1. Shop initializes payment process sending order data including registered card id and `customer_id`

```
{
        "order": {
                "amount": "1.00",
                "currency": "PLN",
                "description": "test",
                "control": "test"
        },
        "seller": {
                "account_id": "123456",
                "url": "https://www.example.com"
        },
        "payer": {
                "first_name": "John",
                "last_name": "Doe",
                "email": "johndoemail@example.com"
        },
        "payment_method": {
                "channel_id": "248",
                "credit_card": {
                        "id":
↪"85c14e6e5608cbc69e19acec41730d59052fbcd306364d96c9cdaafacb7a0833d0fa14280ab9a2b2381fad381f65f0
↪",
                        "customer_id": "f9c6a4-25473"
                }
        },
        "request_context": {
                "ip": "127.0.0.1",
                "language": "pl"
        }
}
```

2. Card is charged
3. and Dotpay send urlc notification with transaction status.

Caution: In case consecutive payment attempts fail, another one should be made not earlier than next day and not more often than daily for not longer than 31 days. Meanwhile shop should take necessary steps to contact the customer to find the cause of the issue.

5 3-D Secure handling (`redirect`)

If payment processing requires redirection to bank / issuer, in response Dotpay will return additional object `redirect` according to the description below.

Element	Type	Comments
redirect	object	complete data required for redirection to the bank / issuer
redirect.url	string	url where customer should be redirected
redirect.method	enumeration (post, get)	redirection http method
redirect.data	object	dictionary (list of <key, values> pairs) of parameters, which need to be sent with redirection to the bank / issuer
redirect.encoding	string	encoding for `request.data` dictionary values

Attention: Redirect data contains signature and need to be sent intact including proper encoding. If data integrity is compromised, payment will be rejected by the bank / issuer.

Note: As an alternative it is possible to redirect (HTTP 302) to the address in `redirect_simplified_url`. In this case redirection to the bank / provider will be handled by Dotpay.

Listing 3: Exemplary response including `redirect.url` and `redirect_simplified_url`:

```
{

        "redirect":{
           "url":"https://ssl.dotpay.pl/test_payment/channel_specific/pv/payment_
↪authentication/M1234-56789/
↪k5bd2c03b5d995boe1862cf775cf8cec114fe36aea928272b0a2b4883a92b14d/",
           "data":{},
           "method":"GET",
           "encoding":"utf-8"
        },
        "redirect_simplified_url":"https://ssl.dotpay.pl/test_payment/channel_
↪specific/pv/payment_authentication/M1234-56789/
↪k5bd2c03b5d995boe1862cf775cf8cec114fe36aea928272b0a2b4883a92b14d/"
 }
```

6 Additional information

6.1 Credit card unregistration

Unregistration can be done in few ways:

1) Client might use link given in payment confirmation emails.
2) Deregistration request might be sent to Dotpay from seller's system via API.

Request should be sent using *DELETE* method to the *https://ssl.dotpay.pl/t2/payment_api/v1/cards/{credit_card_id}/*, where *{credit_card_id}* is card ID which should be removed.

Exemplary request:

DELETE /cards/(**string**: *credit_card_id*)**/**

Response:

```
HTTP/1.1 204 No Content
```

HTTP status codes meaning:

CODE	DESCRIPTION / MEANING
204 No Content	Deleted
404 Not Found	Card not found
400 Bad Request	Request processing error

7 Test environment

Table below contains few exemplary cards which might be used for that purpose. Expiration date is anything from current date to December 2020.

TYPE	NUMBER	CVV2 / CVC2	3DS
Visa	4916 9715 6289 1025	025	No
Visa	4929 5320 2788 7670	670	Yes
MasterCard	5498 5400 7907 4343	343	No

continues on next page

Table 7 – continued from previous page

TYPE	NUMBER	CVV2 / CVC2	3DS
MasterCard	5344 6642 8071 1026	026	Yes

HTTP Routing Table