secupay

Flex 2.0 API Payment Interface

Version 2.3.21

2. Februar 2016

Contents

1 Common

This Documentation describes the flex.API v2.3 for processing online payments.

1.1 the simple.secure.easy system

We enable online payment processing in eCommerce Shops and Portals.

If the customer is using incorrect data, or has a low solvency we wont accept the payment, and decline it with a appropriate error message. The customer should then be forwarded to the checkout page, to chose another payment method.

To process demo payments successfully, your contract needs the proper conditions. If the requests are getting denied because of error 0007 or error 0012, please consult our support team for activation.

1.2 Demo Requests

To mark a request as an demo, you can send the property “demo”. This property prevents any real-life transactions.

2 System description

2.1 REST

The secupay Flex API, subsequent named flex.API, is roughly a REST Interface. Therefor its fairly easy to use via https requests. The hostname of the api is api.secupay.ag, the protocol is https, which leads to the following base url for request:

```
https://api.secupay.ag/
```

For safety reasons only port 443/https is allowed on the api host, you can not use port 80, such requests will get rewritten to port 443. For initial development, you should use our dist system. This ensures clean separation between development and productive environments. Additionally no transactions are getting charged and there is no cost for any request.

```
https://api-dist.secupay-ag.de
```

2.1.1 test environment

To try your connection to the flex.API we set up a test environment.

It is possible to test all flex.API-calls like in the live system without effecting your live contract whilst programming.

After your successful testing process you just need to exchange the URL and APIkey to work on the live system. The other flex.API-calls do not change.

Use the follwing URL for the test environment:

```
https://api-dist.secupay-ag.de
```

The transactions are not going to be booked and the requests will not be charged.

For the API-key just send a request to our customer service.

2.2 Paths

Functions of the API are available under the following path. They are namespaced by using, for example, /payment/<FUNCTION>.

Examples:

• namespace: payment, function: init

```
https://api.secupay.ag/payment/init
```

• namespace: payment, function: gettypes

```
https://api.secupay.ag/payment/gettypes
```

2.3 HTTP-header

All requests need an HTTP Header to define different input/output formats. Furthermore, it is possible to control the language of the reported errors

2.3.1 Content-Type and Accept

We support 2 output formats, JSON and XML. Within the secupay system, we work with JSON, which gets converted to XML if you prefer it.

HTTP-Header Content-Type This controls which data you are sending to the api. To indicate what kind of data you will send, use the header "Content-Type".

To send xml to us:

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

To send json to us:

```
Content-Type: application/json; charset=utf-8;
```

Every Input to the api should be proper utf-8, if anything arrives with a different encoding, we will make it utf8, which could lead to improper conversions of special characters.

HTTP-Header Accept This header defines the content type of the answer.

Example to get back xml:

```
Accept: text/xml;
```

And for json:

```
Accept: application/json;
```

Usually you could send multiple formats in one accept header, however at this point we don't support this. Sending just one content type in your request is appreciated. The headers accept and content-type can be mixed. Its possible to post in json to us, and get xml as result.

2.4 language

The flex.API supports multiple languages. So there are, for example, when incorrect data transmitted to the API, error codes that can be intercepted by the client program. On the other hand, there are also qualified error messages to be displayed to the user directly into the corresponding surface. A typical validation errors is an empty entry in a required field

Currently valid values

```
de_DE
```

as well as

```
en_US
```

2.5 Charset

Only UTF-8 is used as charset by the flex.API for incoming and outgoing data. A validation of incoming data regarding to the charset does not take place. The client program is responsible for the required conversion of data.

2.6 Data structure

Independent of namespace and format all incoming and outgoing data are in a consistent structure. All requests with an http-body encapsulate the user data in the data. Returns of the flex.API are divided into three parts: status, data and errors.

example:

```
{
    "status": "ok",
    "data": {
        "hash": "...",
        "iframe_url": "..."
    },
    "errors": null
}
```

2.6.1 Return: Status

Status gives back if the request was successful. There are three possible status:

ok - the request to the flex.API was successful and the requested action was done.

error - the requested action could not be done. The cause was a technical one. Such failures point to bugs of the flex.API or the Client. The developer has to solve this problem.

failed - the requested action could not be done. In most cases a validation error causes this problem and the form with the error messages should be shown to the user.

2.6.2 Return: Data

The part data contains the used data. The structure of this part depends to the application. Mostly it is a name-value-assignment or a list of name-value-assignments.

Note: The data returned here can be extended in the future to include additional value pairs. This must not lead to errors in the processing

2.6.3 Return: Errors

This part is only filled when Status is set “failed“ or “error“. In this case it contains a list with error codes and error messages.

3 Payment

3.1 Get available payment types

```
https://api.secupay.ag/payment/gettypes
```

This request gives back a list of all available payment types. This list corresponds to the apikey and can change dynamically. The payment types are linked to the corresponding contracts for every customer of secupay. We advise that application for listing of the available payment methods to avoid unwanted error because the selected method of payment is not available. This request is always in real-time.

request:

```
{
        "data": {
                "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace"
        }
}
```

return:

```
{
    "status": "ok",
    "data": [
        "prepay",
        "debit",
        "invoice",
        "creditcard",
        "transfer"
    ],
    "errors": null
}
```

3.2 Initializing / submission of transactions

```
https://api.secupay.ag/payment/init
```

At the beginning the payment transaction has to be declared to the secupay system. Within this the information of the transaction will be sent by the customer system. The secupay system initializes the payment transaction by saving the transaction data and by creating a hash. This hash is used for the following communication. The required information is changing for different payment types and shop systems. Many parameter are optional or are even not considered in some cases. For initializing the payment the technical parameters are the most important.

• apikey - unique key for the contract in the secupay system

demo - No payments will be carried out as long as the value is set on ”1”.

• amount - the amount in the smallest unit of currency (euro cent)

• payment_type - identifies the payment type, for example: creditcard or debit / Examples you will find from part 3.2.1

• url_success - The url used to redirect the user if the transaction was successful.

• url_failure - The url used to redirect the user if the transaction failed or is canceled.

• url_push - The url used to receive status updates.

Additional parameter are described in chapter 5. The flex.API checks the data for completeness. If all required data is present the hash and iFrame link for the transaction will be returned.

Cash data are requested usually by secupay in a separately provided iFrame These responses to the flex.API with the hash and a link to open the iFrames to enter. Below is first carried out by secupay the approval process. Thereafter, the flex.API returns the hash and the status of the transaction.

The parameter ”labels” is optional. If it is not specified, the standard texts of secupay are used.

Submission of payment data takes place in the provided iFrame.

```
{
     "data": {
        "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
        "demo": "1",
        "payment_type": "debit",  <- see examples
        "payment_action": "sale",
        "url_success": "http://shop.example.com/success.php",
        "url_failure": "http://shop.example.com/failure.php",
        "url_push": "http://shop.example.com/push.php",
        "apiversion": "2.11",
        "shop": "APITest Shop",
        "shopversion": "1.0",
        "moduleversion": "1.0",
        "language": "de_DE",
        "title": "Herr",
        "firstname": "Test FN",
        "lastname": "Test LN",
        "company": "Test Company",
        "street": "Test Street",
        "housenumber": "5t",
        "zip": "12345",
        "city": "TestCity",
        "country": "DE",
        "telephone": "+4912342134123",
        "dob_value": "01.02.1903",  <- mandatory for payment_type invoice
        "email": "test@ema.il",
        "ip": "172.31.6.49",
        "amount": "199",
        "currency": "EUR",
        "purpose": "Test Order #1",
        "order_id": "100203",
        "delivery_address": {
            "firstname": "Test FN",
            "lastname": "Test LN",
            "company": "Test Company",
            "street": "Test Street",
            "housenumber": "5t",
            "zip": "12345",
            "city": "TestCity",
            "country": "DE"
            },
        "basket": [
             {
```

```
                "item_type": "shipping",
                "name": "standard delivery",
                "tax": "19",
                "total": "1324"
              },
              {
                "item_type": "article",
                "article_number":"3211",
                "quantity": "2",
                "name": "Testname 1",
                "ean": "4123412341243",
                "tax": "19",
                "total": "1324",
                "price": "1000"
               },
               {
                 "item_type": "article",
                 "article_number":"48875",
                 "quantity": "2",
                 "name": "Testname 1",
                 "ean": "4123412341236",
                 "tax": "19",
                 "total": "1324",
                 "price": "1000"
                }
          ]
     }

}
```

answer

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303"
    },
    "errors": null
}
```

The customer can finalize the transaction in the provided iFrame. It is possible to embed the URL or use a redirect. The layout of the input forms is designed to adapt to the available space. The ”payment_action” parameter controls the processing of the transaction by us, for

the time being, there are the values ”sale” and ”authorization”. Sale is a direct payment. To perform the transaction later, you have to transmit “authorization” here.

3.2.1 Special features for prepay

There is no iFrame with the payment method prepay. All necessary information is included directly in the response to the init request. This information must be provided to your customer in a suitable form to enable the payment transaction can be settled later.

For prepay (advanced payment) change the payment_type in prepay in the example for the submission of a transaction.

```
{
    "data": {
        ...
        "payment_type": "prepay",
        ...
```

Answer

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303",
        "purpose": "TA 123456",
        "payment_data": {
            "accountowner": "secupay AG",
            "accountnumber": "0123456789",
            "bankcode": "01234567",
            "iban": "DE00012345678912345678",
            "bic": "COBADEFF103"
        }
    },
    "errors": null
}
```

• purpose - the purpose to be used for the transfer, to allow an assignment of the subsequent payment of the transaction .

• payment_data - Bank account to be used for the transfer

• iframe_url - has no use for prepay transactions

3.2.2 Example for debit

For debit change the payment_type in debit in the example for the submission of a transaction.

- experience - optional see part 5.6 auf Seite 32

```
{
    "data": {
        ...
        "payment_type": "debit",
        "experience": {
            "positive": "1",
            "negative": "0"
        },
        ...
```

The answer contains the following data.

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303"
    },
    "errors": null
}
```

3.2.3 Example for Invoice

For invoice change the payment_type in invoice in the example for the submission of a transaction.

- experience - optional see part 5.6 auf Seite 32
- dob_value - mandatory field for invoice buying

```
{
    "data": {
         ...
        "payment_type": "invoice",
        "dob_value": "01.02.1903",
        "experience": {
            "positive": "1",
            "negative": "0"
        },
         ...
```

The answer contains the following data.

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303"
    },
    "errors": null
}
```

3.2.4 Example for Creditcard

For creditcard change the payment_type in creditcard in the example for the submission of a transaction.

- experience - optional see part 5.6 auf Seite 32

```
{
    "data": {
         ...
        "payment_type": "creditcard",
        "experience": {
            "positive": "1",
            "negative": "0"
        },
         ...
```

The answer contains the following data.

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303"
    },
    "errors": null
}
```

3.2.5 Example for Transfer

For the method of payment ”transfer” als payment details must be transmit in the init request.

```
{
    "data": {
        ...
        "payment_type": "transfer",
        ...
        "purpose": "Test Order #1",
        "payment_data": {
            "accountowner": "Test LN",
            "iban": "DE00012345678912345678",
            "bic": "COBADEFF103"
        },
        ...
```

Answer

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303",
        "purpose": "TA 123456",
        "payment_data": {
            "accountowner": "secupay AG",
            "iban": "DE00012345678912345678",
            "bic": "COBADEFF103"
        }
    },
    "errors": null
}
```

- iframe_url - has no usage for transfer transactions

3.3 Generating APIKey via API

This feature provides a way to dynamically generate a APIkey. For normal store operating this function should not be necessary and is valid only for explicitly activated APIKeys.

```
https://api.secupay.ag/payment/requestapikey
```

```
{
    "data": {
        "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
        "user": {
            "title": "Herr",
            "company": "Firma",
            "firstname": "Test FN",
            "lastname": "Test LN",
            "street": "Test Street",
            "housenumber": "5t",
            "zip": "12345",
            "city": "TestCity",
            "telephone": "+4912342134123",
            "industry": "4812",
            "dob_value": "01.02.1903",
            "email": "test@ema.il",
            "homepage": "www.test.de"
        },
        "payout_account": {
            "accountowner": "Test Inhaber",
            "iban": "DE00012345678912345678",
            "bic": "COBADEFF103"
        },
        "project": {
            "name": "APITest Shop"
        },
        "iframe_opts": {
            "show_basket": true,
            "basket_title": "Ihr Warenkorb",
            "submit_button_title": "Kostenpflichtig Bestellen!"
        },
        "payin_account": false
    }
}
```

3.3.1 Specific Parameters

Except for iframe_opts all parameters are mandatory.

• user - Contract data of project owner

• payout_account - Account to which the transactions are to be paid

• project - Project -specific properties , for now only the name of the project

• iframe_opts - defines a list of standards that affect the appearance of the iframe

– show_basket - Set in a Basket Iframe displays true / false

– basket_title - The title of the basket

– submit_button_title - The text of the Submit Button

– logo_base64 - The logo to be filed for the iFrame as base64 representation

– cession - Speech in iFrame , ' personal ' or ' formal '

– logo_base64 as well as cession unless communicated explicitly taken from the existing contract.

• payin_account - Query a virtual account as the destination for remittances (this requires a corresponding option in the contract). This parameter returns bank details, IBAN and BIC for the virtual account that's linked with the given contract.

logo_base64 and cession will be taken over from the existing contract if not explicitly transmitted

Answer


```
{
    "status": "ok",
    "data": {
        "apikey": "48411xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx744e",
        "contract_id": "1234",
        "payin_account":{
            "accountowner": "Vorname Name  des Vertragspartners",
            "iban": "DE00012345678912345678",
            "bic": "COBADEFF103"
         }
    },
    "errors": null
}
```


3.4 Status check

```
https://api.secupay.ag/payment/status
```

This function queries the status and further information for a hash. In the future, the answer can be expanded to include more information.

```
{
        "data": {
                "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "hash":"tujevzgobryk3303"
        }
}
```

Answer

```
{
        "status": "ok",
        "data": {
                "hash": "tujevzgobryk3303",
                "payment_status": "acccepted",
                "status": "accepted",
                "created": "2013-06-10 10:27:42",
                "demo": 0, "trans_id": "1831201",
                "amount": 100,
                "opt": {

                }
        },
        "errors": null
}
```

The data contained in the part of the answer status has to suit after forwarding at url_success on accepted. An exception may be made for pre-authorized transactions. It is recommended, that after the forwarding is done at url_success, by a status query the status to verify ”accepted”. The property ”opt” is filled with payment-type specific data. - for example: on Invoice, it gets filled with the bank account to inform the customer via e-mail. At the moment, this property is only filled for invoice transactions, but will also be available for other payment types in the future. It is therefore necessary that new implementations handle this property in general.

3.5 Capture pre-authorized payments

Pre-authorized payments have to perform the payment, be gecleared, this can be done in 2 ways, either by post on

```
https://api.secupay.ag/payment/<hash>/capture
```

or via GET on

```
https://api.secupay.ag/payment/<hash>/capture/<apikey>
```

Via GET Version the user receives an overview of the transaction in the browser and the payment can be start with a button. In the POST version has to be submitted as usual the apikey in data Array.

3.6 Cancellation pre-authorized / not submitted final payment

There are two ways for cancellation – either with user interaction via GET or as a pure server to server transfer. May be canceled for the time being only transactions which are pre-authorized, or were not submitted final.(For debits, this is generally the day following, about 7 clock a.m.)

POST:

```
https://api.secupay.ag/payment/<hash>/cancel
```

GET:

```
https://api.secupay.ag/payment/<hash>/cancel/<apikey>
```

In the post variant is returned for success or failure in accordance with status failed /ok.

3.7 Invoice

With the payment method invoice there are in addition to the normal API functions which have the option to set the date of shipment.

Capture (shipping date of an invoice)

To set the shipping date of an invoice, the following URL must be called

```
https://api.secupay.ag/payment/{hash}/capture/{apikey}
```

The values of the hash and the apikey from the merchant should be replaced accordingly.

Example for an URL:

```
https://api.secupay.ag/payment/oxaijkeaucbl2041/
capture/6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace
```

Alternatively, a set of shipping date via POST is possible without direct user interaction, in this variant of the date of request is used as the date of shipment. For this has to be sent to the following URL a POST request:

```
https://api.secupay.ag/payment/{hash}/capture/
```

In Data object of inquiry in addition to the parameters APIkey still needs to be submitted:

```
{
        "data": {
                "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "tracking": {
                        "provider": "DHL",
                        "number": "TC123456789"
                },
                "invoice_number": "RN 0001"
        }
}
```

The parameter “tracking” (shipping service and tracking number) and “invoice_number” are optional. The mentioned ”invoice_number” is included in a subsequent status request in the response.

Additional return parameters at a status request

In a status request on Invoice transactions you get returned additional fields in opt Property:

Answer

```
{
        "status": "ok",
        "data": {
                "hash": "tujevzgobryk3303",
                "status": "accepted",
                "payment_status": "accepted",
                "created": "2013-06-10 10:27:42",
                "demo": 0,
                "trans_id": "1831201",
                "amount": 100,
                "opt": {
                 "recipient_legal": "secupay AG, Goethestraße 6,
                 01896 Pulsnitz",
                 "payment_link":  "https://api.secupay.ag/
                 payment/tujevzgobryk3303",
                 "payment_qr_image_url":  "https://api.secupay.ag/
                 qr?d=http %3A%2F%2F....",
                        "transfer_payment_data": {
                                "purpose": "TA 123456 DT 20130610",
                                "accountowner": "secupay AG",
                                "iban": "DE00012345678912345678",
                                "bic": "COBADEFF103",
                                "accountnumer": "0123456789",
                                "bankcode": "01234567",
                                "bankname": "Test Bank"
```

```
                },
                "invoice_number": "RN 0001"
            }
        },
        "errors": null
}
```

Payment of the invoice by the end customer To pay the invoice, the customer has several options. These are offered to him in an iFrame. The URL for this is the same as the returned iFrame URL in init. This URL can be directly integrated via QR code or via links in emails respectively invoice PDFs. So the customer has an easy way to pay the invoice.

3.8 Recurring payments (subscription)

To be able to perform recurring payments, you first have to unlock your contract for it. Then there are 2 methods to an initial subscription perform payment. The preferred option is the requests to /Payment/getSubscription for an already successful payment. Alternatively you can although sent in the init of the first subscription transaction a subject “subscription”, than you get back an answer with the parameter “subscription_id”. With this “subscription_id” you can trigger the new transactions. Important is only that there is the object, the purpose can be empty.

The Subscription-Id can be used to create a new payment using cash data entered by the payer.

The interaction between you and secupay runs from here without any interaction by the end costumer.

Init Request:

```
{
        ...
        "data": {
                "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "subscription": {
                        "purpose": "ABO Monatlich bei ..."
                },
                "amount": 2350,
                "purpose": "Ihre Bestellung bei ..."
        }
}
```

3.8.1 Creating a subscription through a given hash

If you want to create a subscription to a payment already made subsequently, this is possible with the action getSubscription. In this connection the POST to /Payment/getSubscription, the used APIkey and the hash of the original transaction will be send. In response you will receive an Subscription-Id with the subscription payments could be made.

POST

```
https://api.secupay.ag/payment/getSubscription
```

```
{
        "data": {
                "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "hash": "tglswdmhziwb84171",
                "subscription": {
                        "purpose": "ABO Monatlich bei ..."
                }
        }
}
```

Answer

```
{
        "status": "ok",
        "data": {
                "subscription_id": 1234
        },
        "errors": null
}
```

3.8.2 Using the Subscription-ID

Purpose The purpose of subscription transactions can be defined at serveral places. Here, the following in order of precedence

1. purpose in POST on payment/subscription
2. purpose in subobject “subscription” on payment[init|getSubscription]
3. purpose of the origin transaction

To minimize the chargeback risk, please pay attention to use a meaningful purpose which the customer can clearly recognise.

3.8.3 Perform subscription payment

To produce a subscription transaction off the /payment/init returned Subscription-Id, the following POST requestion /payment/subscription has to be performed.

POST

```
https://api.secupay.ag/payment/subscription
```

In data object of inquiry in addition to those parameters ”amount”, ”subscription_id” and ”apikey” are received:

```
{
        "data": {
                "apikey":"6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "subscription_id": 1234,
                "amount": 2350,
                "purpose": "Ihre Bestellung bei ..."
        }
}
```

Answer

```
{
        "status": "ok",
        "data": {
                "hash": "tujevzgobryk3303"
        },
        "errors": null
}
```

3.8.4 Special case of recurring invoice transactions

In a recurring invoice transaction the customer should be received a message with the new payment method with the indication of the transfer data which is returned as a response to the transaction creating

3.9 postponement of transactions

Transactions can be postponed f.e. till the performance is completed. In this case you withhold the payment to the stakeholder of a transaction.

```
https://api.secupay.ag/payment/init
```

Init see part 3.2

```
 {
        ...
        "data": {
                "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "set_accrual": "1"
        }
}
```

answer

gives back the success True or the failure False of the accruel

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303",
        "iframe_url": "https://api.secupay.ag/payment/tujevzgobryk3303",
        "accrual": true
    },
    "errors": null
}
```

The cancellation of the postponement occurs via POST through the folowing URL

POST

```
https://api.secupay.ag/payment/reverseaccrual
```

```
{
        data": {
                "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "hash": "tujevzgobryk3303"
        }
}
```

3.10 addition of a stakeholder position to an existing transaction

For a transaction always the last amount is used for an apikey/name-combination. This function can only be executed, till ”set_accrual”: ”1” is not abolished with ”reverseaccrual”.

```
https://api.secupay.ag/payment/additem
```

```
{
        "data": {
        "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
        "hash": "tujevzgobryk3303",
```

```
        "basket": [
             {
                "item_type": "stakeholder_payment",
                "apikey": "37373xxxxxxxxxxxxxxxxxxxxxxxxxxxxx2fe2",
                "name": "plattform fee",
                "total": "1324"
             }
         ]
        }
}
```

Answer

```
{
    "status": "ok",
    "data": {
        "hash": "tujevzgobryk3303"
    },
    "errors": null
}
```

4 Push API

For security reasons the API offers the option to confirm the transaction by pure server to server communication.

When submitting the transaction thr parameter url_push will be transmitted. These will be send when the status changes to the transactions of the new status via HTTP POST.

This allows, for example, the use of an inventory control system which automatically gets the current status of the payment process and the shipment of packages is simplified and made more secure.

Any relevant status changes of the transaction in the secupay system will generate a push message per HTTP POST to a chosen URL (see parameter url_push).

URL:

```
https://push.example.net/push_client.php
```

```
POST /push_client.php HTTP/1.1

hash=jtnjpfgrbrqk3300&amount=1199&status_id=6&status_description=
abgeschlossen&changed=1365444092&payment_status=accepted&
apikey=6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace&hint=
```

- hash: hash of transaction
- amount: transaction amount
- status_id: id of the exact status in the secupay system
- status_description: name of the exact status in the secupay system
- changed: UTC Unix timestamp at which the change occurred
- apikey: the apikey which was used for the transaction, it is recommended to compare it against your apikey before the message is accepted
- hint: (todo) can provide additional information
- payment_status
- simplified status, provides categories for the exact status
  - accepted - transaction was successful
  - authorized - transaction is authorized
  - denied - transaction was denied or canceled
  - issue - there is a problem with the transaction, for example a charge-back has occurred
  - void - the transaction was voided or refunded

  - issue_resolved - the problem with the transaction has been solved, for example, Settled chargeback

- (additional for subscription payments)
  - subscription_id - The ID used for the subscription payment

The push notification only needs to be acknowledged by the receiving side. For this purpose, they have to respond as a simple text containing the full text inquiry including ack=Approved .

To acknowledge the push message display ack=Approved and attach the received data.

Example:

```
ack=Approved&hash=jtnjpfgrbrqk3300&status_id=6&status_description=
abgeschlossen&changed=1365444092&payment_status=accepted&
apikey=6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace&hint=
```

If the apikey or hash do not match ack=Disapproved and the received data should be displayed. You can add the parameter error with a suitable error message.

Example:

```
ack=Disapproved&error=no+matching+order+found+for+hash&
hash=jtnjpfgrbrqk3300&status_id=6&
status_description=abgeschlossen&changed=1365444092&
payment_status=accepted&
apikey=6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace&hint=
```

We recommend to check for the Referrer api.secupay.ag or in the development system api-dist.secupay-ag.de and if possible only accept messages with https (port 443). All received data should also be escaped. For the above example, the API expect the following response:

```
$ref = $_SERVER['HTTP_REFERER'];
if (strpos($ref, 'api.secupay.ag') !== FALSE) {
    echo" Der Push ist von $ref";
}
```

5 Parameter

```
apikey                          => "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace"
payment_type            => "debit"
demo                            => "0"
url_success                     => "http://shop.example.com/success.php"
url_failure                     => "http://shop.example.com/failure.php"
url_push                        => "http://shop.example.com/shop/push"
language                        => "de_DE"
shop                            => "APITest Shop"
shopversion                     => "1.0"
moduleversion           => "1.0"
title                           => "Herr"
firstname                       => "Test FN"
lastname                        => "Test LN"
company                         => "Test Company"
industry                        => "4812"
street                          => "Test Street"
housenumber                     => "5t"
zip                                     => "12345"
city                            => "TestCity"
country                         => "DE"
telephone                       => "+4912342134123"
dob                                     => "01.02.1903"
email                           => "test@ema.il"
ip                                      => "172.31.6.49"
amount                          => "1199"
currency                        => "EUR"
purpose                         => "Test Order #1"
basket                          => <siehe extra Definition #5.2>
userfields                      => <siehe extra Definition #5.3>
delivery_address        => <siehe extra Definition #5.4>
order_id                        => "100203"
note                            => "default note text"
apiversion                      => "2.11"
labels                          => <siehe extra Definition #3.2>
set_accrual                => false <siehe extra Definition #3.9>
merchant_customer_id  => NULL  <siehe extra Definition #5.5>
```

5.1 Additional information for parameters

• amount - the value is expected in the lowest unit of the chosen currency. if nothing is specified euro cent will be used.

5.2 Basket

5.2.1 basket structure

The basket is structured as an array of objects. This is an easy way to send variable parameter for each item in the basket. This information can assist us in the customer support and to calculate the risk of the order. The following structure is supported by the secupay system and can be extended if needed.

```
[
        {
                "article_number": "1234",
                "item_type": "article",
                "name": "Testname1",
                "model": "test model",
                "ean": "2309842",
                "quantity": "2",
                "price": "50",
                "total": "100",
                "tax": "19"
        },
        {
                "article_number": "1235",
                "item_type": "article",
                "name": "Testname2",
                "model": "test model",
                "ean": "2309843",
                "quantity": "4",
                "price": "199",
                "total": "796",
                "tax": "19"
        },
        {
                "item_type": "shipping",
                "name": "standard shipping fee",
                "total": "500",
                "tax": "19"
        },
]
```

5.2.2 basket item types

Further differentiation of the basket item types can be accomplished via the parameter item_type.

The following types of basket item types are possible:

• article – goods- / service position

• shipping – shipping costs

• donation – donation share

• stakeholder_payment - Amount for third parties (sets a corresponding activation of the contract in advance)

For stakeholder payment in addition to the parameter API key is required to identify the intended recipient and handed over the amount in the parameter total.

```
[
        {
                "item_type": "stakeholder_payment",
                "apikey":  "37373xxxxxxxxxxxxxxxxxxxxxxxxxxxxx2fe2",
                "name": "plattform fee",
                "total": "499"
        },
]
```

5.3 Userfields

Userfields can be used to add individual information, in the future it may be used to sort or filter transactions in the secupay system. It could be sensible to add notes or special properties of the order or the customer.

```
[
        "userfield_1": "test 1",
        "userfield_2": "test 2...",
        "userfield_3": "test 3"
]
```

5.4 Delivery adress

The delivery_address is an object that receives the elements of the delivery address.

The correct transmission of the delivery address helps us support the transactions and the risk assessment of the order.

Subsequent structure is recorded natively by us and can be extended if necessary.

```
[
        "firstname": "Test FN",
        "lastname": "Test LN",
        "company": "Test Company",
        "street": "Test Street",
        "housenumber": "5t",
        "zip": "12345",
        "city": "TestCity",
        "country": "DE"
]
```

5.5 transfer of shop client's customer number

To reduce the rejection rate it is favourable to be able to clearly assign those shop customers for which experiences are existing.

Through the transmission of the customer number this will be possible.

```
https://api.secupay.ag/payment/init
```

Init see part 3.2

```
 {
        ...
        "data": {
                "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
                "merchant_customer_id": "1"
        }
}
```

5.6 Experiences / evaluation

experience - Inform us about your payment experiences with your customer (requires the corresponding activation for the respective contract)

positive - here you can inform us about how often the customer payed punctually

- 0 - no positive experience with the customer
- 1 - one positive completed transaction / sale
- 2 or current value - at least 2 positive completed transactions / sales or number of those transactions

negative - here you can inform us about negative indicators of your customer

- 0 -no negative experience with the customer

- 1 - the customer currently has overdue open positions
- 2 or current value - in the past the customer has caused return debits or he payed only after several remindings

```
{
    "data": {
        ...
        "experience": {
            "positive": "1",
            "negative": "0"
        },
        ...
```

5.7 own labels / marking shopping cart title, sending and cancellation button

With the transfer of the labels in the init (see part 3.2) you can change the marking of the shopping cart title, the sending and cancellation button on your own.

```
...

         "labels": {
                "en_US": {
                       "basket_title": "Your Order",
                       "submit_button_title": "Submit",
                       "cancel_button_title": "Return to Basket"
                              },
                "de_DE": {
                    "basket_title": "Ihre Bestellung",
                    "submit_button_title": "Daten Übermitteln",
                    "cancel_button_title": "Zum Warenkorb"
                 }
          }
```

5.8 industry / branch code

- here you specify the MCC Code for creditcard payments suitable for your commerce. The MCC Codes you will find in the list under the URL `https://github.com/greggles/mcc-codes`

6 Error codes

An error message consists of an error code and an message. Language of the message depends on the used language parameter. At the moment “en_us”(default) and “de_de” are supported. list of possible english error messages:

```
0001 => Invalid apikey
0002 => Invalid hash
0003 => Cannot capture unauthorized payment
0004 => Cannot cancel/void unaccepted payment
0005 => Invalid amount
0006 => Cannot refund unaccepted payment
0007 => No payment type available
0008 => Hash has already been processed
0009 => Scoring invalid
0010 => Payment denied by Scoring
0011 => Payment couldnt be finalized
0012 => Selected payment type is not available
0013 => Apikey mismatch
0014 => Cannot capture specified payment
0015 => Cannot cancel/void specified payment
0016 => Cannot refund specified payment
0017 => Invalid Paymentdata
0018 => Missing Parameter
0019 => Couldnt create the user
0020 => Username/email unavailable
0021 => Username or password invalid
0022 => Userauth Token invalid
0023 => Could not connect the card to the user
0024 => Invalid value for parameter
0025 => Cannot process specified payment
0026 => Cannot create payment information
0027 => Invalid data
0028 => Payment Provider declined the payment
0029 => No transaction available for this terminal
0030 => Cannot create transaction
0031 => Unknown parking zone
0032 => The password you entered is too short
0033 => The passwords are not equal
0034 => The data provided is not sufficient
0035 => There was an error creating the Apikey
0036 => The Plate is not valid
0037 => Payment data missing
0038 => Cannot find any active ticket for this zone and car
```

```
0039 => There is already valid ticket for this car in this zone
0040 => There is not any tariff valid for your parking time
0041 => Password recovery failed
0042 => Your password has been recovered
0043 => Error recovering the password
0044 => Failed to save your payment data
0045 => Your payment data has been saved
0046 => Referenced payment is not accepted
0047 => Cannot set accrual
0048 => Cannot reverse accrual
0049 => Failed to add Data
0050 => Your bank doesn't support SEPA direct debit
```

German error messages:

```
0001 => Ungültiger apikey
0002 => Ungültiger hash
0003 => nicht autorisierte Zahlung kann nicht eingezogen werden
0004 => nicht abgeschlossene Zahlung kann nicht storniert werden
0005 => Ungüliger Betrag
0006 => nicht abgeschlossene Zahlung kann nicht rückabgewickelt werden
0007 => Keine Zahlart verfügbar
0008 => Hash wurde schon verarbeitet
0009 => Scoring ungülig
0010 => Zahlung durch Scoring abgelehnt
0011 => Zahlung konnte nicht abgeschlossen werden
0012 => ausgewählte Zahlart ist nicht verfügbar
0013 => Apikey stimmt nicht überein
0014 => Zahlung konnte nicht abgeschlossen werden
0015 => Zahlung konnte nicht storniert bzw erstattet werden
0016 => Zahlung konnte nicht erstattet werden
0017 => Zahlungsmitteldaten ungültig
0018 => Fehlernder Parameter
0019 => Konnte den Nutzer nicht erstellen
0020 => Nutzername/E-Mail bereits vergeben
0021 => Nutzername oder Passwort ungültig
0022 => Userauth Token ungültig
0023 => Konnte die Karte nicht verknüpfen
0024 => Ungültiger Wert für einen Parameter
0025 => Angegebene Zahlung konnte nicht verarbeitet werden
0026 => Zahlungsinformationen konnten nicht erstellt werden
0027 => Daten fehlerhaft
0028 => Fehler beim einreichen der Transaktion beim Zahlungsanbieter.
0029 => Für dieses Terminal ist keine Transaktion hinterlegt.
0030 => Konnte die Transaktion nicht erstellen
0031 => Unbekannte Parkzone
0032 => Das eingegebene Passwort ist zu kurz
0033 => Die eingegebenen Passwörter sind nicht gültig
0034 => Übermittelte Daten nicht vollständig
0035 => Fehler beim anlegen des APIKeys
0036 => Das Kennzeichen ist ungültig
0037 => Zahlungsmitteldaten fehlen
0038 => Es konnte kein aktives Ticket für diese Zone und dieses Fahrzeug
                gefunden werden
0039 => Es existiert bereits ein Ticket für diese Zone und
                dieses Fahrzeug
0040 => Für die angegebene Parkzeit existiert kein gültiger Tarif
```

```
0041 => Wiederherstellung des Passworts ist gescheitert.
0042 => Ihr Passwort wurde wieder hergestellt.
0043 => Fehler beim wiederherstellen des Passworts
0044 => Konnte die Zahlungsmitteldaten nicht speichern.
0045 => Ihre Zahlungsmitteldaten wurden gespeichert.
0046 => Referenzierte Zahlung ist nicht akzeptiert
0047 => Kann accrual nicht setzen
0048 => Kann accrual nicht entfernen
0049 => Hinzufügen der Daten ist fehlgeschlagen
0050 => Die angegebene Bank nimmt nicht am SEPA-Lastschriftverfahren teil.
```