secupay

Flex 2.0 API Payment Schnittstelle

June 10, 2013

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, 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
```

2.2 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.

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 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.

2.5 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
}
```

2.5.1 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.

2.5.2 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.

2.5.3 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.

2.6 language

To get different languages for the resulting iframe and error messeages, you can use the language parameter.

For now, it accepts de_DE and en_US as values and defaults to de_DE if not specified.

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": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace"
    }
}
```

return:

```
{
    "status": "ok",
    "data": [
        "creditcard",
        "debit"
    ],
    "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
- payment_type - identifies the payment type, for example: creditcard or debit
- 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 4. 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.

Submission of payment data takes place in the provided iFrame.

```
{
    "data": {
      "apikey": "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace",
      "demo": "0",
      "payment_type": "creditcard",
      "apiversion": "2.11",
      "amount": 100,
      "url_success": "https://shop.example.com/success",
      "url_failure": "https://shop.example.com/failed",
      "url_push": "https://shop.example.com/push.php"
    }
}

{
    "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.

4 Push API

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

Any status change 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
```

- 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

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=6801
    fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace&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
```

We recommend to check for the Referrer api.secupay.ag and if possible only accept messages with https (port 443). All received data should also be escaped.

5 Parameter

```
apikey                  => "6801fxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx7ace"
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"
street                  => "Test Street"
housenumber             => "5t"
zip                     => "12345"
city                    => "TestCity"
country                 => "DE"
telephone               => "+4912342134123"
dob_value               => "01.02.1903"
email                   => "test@ema.il"
ip                      => "172.31.6.49"
amount                  => "1199"
currency                => "EUR"
purpose                 => "Test Order #1"
basket                  => <see definition #5.2>
userfields              => <see definition #5.3>
order_id                => "100203"
note                    => "default note text"
apiversion              => "2.11"
```

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

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",
        "name": "Testname1",
        "model": "test model",
        "ean": "2309842",
        "quantity": "2",
        "price": "50",
        "total": "100",
        "tax": "19"
    },
    {
        "article_number": "1235",
        "name": "Testname2",
        "model": "test model",
        "ean": "2309843",
        "quantity": "4",
        "price": "199",
        "total": "796",
        "tax": "19"
    }
]
```

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"
]
```

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 => Cannot 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.
```

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 => Ungültiger Betrag
0006 => nicht abgeschlossene Zahlung kann nicht rückabgewickelt werden
0007 => Keine Zahlart verfügbar
0008 => Hash wurde schon verarbeitet
0009 => Scoring ungültig
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 => Fehlender Parameter
0019 => Konnte den Nutzer nicht erstellen
0020 => Nutzername/E-Mail bereits vergeben
0021 => Nutzername oder Passwort ungültig
0022 => Userauth Token ungueltig
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
```