# PAYMENT GATEWAY API

# PAYMENT GATEWAY API

## *Contents*

**INTRODUCTION** ..... 4

**INTEGRATION USING MODULES OR INTEGRATION PACKAGES** ..... 4

- **MODULES** ..... 4
- **INTEGRATION PACKAGES** ..... 4

**INTEGRATING WITH THE PAYMENT GATEWAY API** ..... 4

**PAYMENTS** ..... 5

- **THE PAYMENT REQUEST** ..... 5
- **THE PAYMENT REQUEST STRUCTURE** ..... 6
- **ORDER ROWS** ..... 7
- **PAYMENT RESPONSE MESSAGE** ..... 8
- **RESCUE PAYMENT** ..... 9
- **CALLBACK** ..... 9
- **CALLBACK, PAYMENT RESPONSE** ..... 9
- **CALLBACK, STORED CARD ALIAS EXPIRED** ..... 9
- **MOBILEPAY PAYMENTS** ..... 10
- **SWISH PAYMENTS** ..... 10
- **VIPPS PAYMENTS** ..... 11
- **CARD PAYMENTS** ..... 13
- **RECURRING CARD PAYMENTS** ..... 14
- **AUTOMATIC RECUR** ..... 16
- **PAYMENT FACILITATORS** ..... 16
- **STORED CARDS** ..... 17

**WEBSERVICES** ..... 18

- **ANNUL** ..... 19
- **CANCELRECURSUBSCRIPTION** ..... 20
- **CONFIRM** ..... 21
- **CREDIT** ..... 22
- **GETPAYMENTMETHODS** ..... 23
- **GETRECONCILIATIONREPORT** ..... 24
- **GETSTOREDCARD** ..... 26
- **LOWERAMOUNT** ..... 27
- **PREPAREPAYMENT** ..... 28
- **QUERY** ..... 30
- **RECUR** ..... 33

**MAC** ..... 35

- **FORMULA** ..... 35

**APPENDIX 1** ..... 36

- **PAYMENT METHODS** ..... 36
- **WEBSERVICES PER PAYMENT METHOD** ..... 37
  - **INDEPENDENT WEBSERVICES** ..... 38
- **THE CUSTOMER ELEMENT** ..... 39
- **TRANSACTION STATUS** ..... 40
- **STATUS CODES** ..... 40
- **GLOSSARY** ..... 43

**APPENDIX 2** ..... 44

- **BASIC CARD PAYMENTS WITH UNKNOWN CUSTOMER** ..... 44

**BASIC CARD PAYMENTS WITH REQUIRED CUSTOMER DATA ..... 44**
**BASIC CARD PAYMENTS WITH FULL CUSTOMER DATA ..... 44**
**BASIC RECURRING CARD PAYMENTS ..... 45**

## INTRODUCTION

This document is for merchants who are interested in accepting card, bank, invoice, payment plan, or e-wallet payments from their customers. The payments will go through our payment gateway, which redirects the customer to their bank, a card service provider, or a payment service provider, where the customer can complete the payment.

There are also other ways to integrate with Svea's payment services, like using a module or integration package, but those are described in other documents.

## INTEGRATION USING MODULES OR INTEGRATION PACKAGES

### MODULES

Svea Bank has developed module plug-ins to a number of e-commerce platforms. For a shop that uses one of those platforms, this is the easiest way to get started. The list of modules can be found at https://www.svea.com/se/sv/foretag/betallosningar/betallosningar-for-e-handel/tech-site.

### INTEGRATION PACKAGES

To create a unified API for all kinds of payments, Svea Bank has developed code libraries for PHP, Java, and C# that the merchant can import and call from their own code. These can be downloaded from GitHub at https://github.com/sveawebpay. Look for the repositories called php-integration, java-integration, and dotnet-integration. There is a button to download as a zip file for those who do not use git.

For card, bank, and e-wallet payments, the integration packages will communicate with the payment gateway. For invoice and payment plan payments, they will bypass the payment gateway and communicate directly with Svea's invoice and payment plan system.

## INTEGRATING WITH THE PAYMENT GATEWAY API

If the merchant wants more control of the integration, they can bypass the modules and integration packages and integrate with the payment gateway API directly. This includes making payment requests directly to Payment Gateway, to integrate with our webservices, and to be prepared to receive callbacks from Payment Gateway. These things are what the rest of this document is about.

## PAYMENTS

To initiate a payment, the merchant sends the customer to Payment Gateway with a POST request. Payment Gateway will then in turn redirect the customer to the end destination, which can be e.g. a bank or a card payment provider. Depending on the payment type, the customer might have to do something there to complete the payment. E.g. if it is a card payment, they might have to fill in their credit card number etc.

After the request has been handled, the customer will normally be sent back to the return URL that was specified in the payment request. It the merchant has specified a callback URL, a separate callback will also be sent, with the same information.

URL for ***test payments*** is:
**https://webpaypaymentgatewaystage.svea.com/webpay/payment**

URL for ***production payments*** is:
**https://webpaypaymentgateway.svea.com/webpay/payment**

### THE PAYMENT REQUEST

The request should contain the following parameters:

| HTTP POST | |
|---|---|
| ***Parameter name*** | *Description* |
| **merchantid** | Your Merchant ID |
| **message** | Payment information, Base64-encoded |
| **mac** | Checksum |

• merchantid - is the store ID that you received from your integrator.
• message - contains the Base64 encoded XML you have generated as described in the section The Payment Request Structure.
• mac – a code that is calculated in order to verify the sender of the call. To generate it, you must have received a secret word from your integrator. See more under the section MAC.

**Example:**

```
<form name='form' method='post' action='https://webpaypaymentgatewaystage.svea.com/webpay/payment'>
<input type='hidden' name='merchantid' value='1100'/>
<input type='hidden' name='message' value='Jmx0Oz94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiJmd
0OyAKJmx0O3BheW1lbnQmZ3Q7IAombHQ7bWVzc2FnZSZndDsgCiZsdDtwYXltZW50TWV0aG9kJmd0O0tPUlQmbHQ7L3BheW1l
bnRNZXRob2QmZ3Q7IAombHQ7Y3VycmVuY3kmZ3Q7U0VLJmx0Oy9jdXJyZW5jeSZndDsKJmx0O2Ft
b3VudCZndDs3MzMwMCZsdDsvYW1vdW50Jmd0OyAKJmx0O2N1c3RvbWVycmVmbm8mZ3Q7dGVzdDEy
MyZsdDsvY3VzdG9tZXJyZWZubyZndDsKJmx0O3JldHVybnVybCZndDtodHRwOi8vbG9jYWxob3N0
OjgwODgvZXBheW1lbnQvcGF5bWVudCZsdDsvcmV0dXJudXJsJmd0OyAKJmx0Oy9tZXNzYWdlJmd0
OyAKJmx0Oy9wYXltZW50Jmd0Ow==='/>
<input type='hidden' name='mac' value='df74ce933f1d367d4100b4d34ad6970760c6040e13d4
9b94b36bd81239c2c3a724435ab5dc4e1065c861944f9a1e56fa1f53f1cd22d71564e69d5256fba24d43'/>
</form>
```

**THE PAYMENT REQUEST STRUCTURE**
The call is structured in XML with the root element <payment>. It contains all information about the payment.

| <payment> | | | | |
|---|---|---|---|---|
| ***Element name*** | ***Content*** | ***Value*** | ***Format*** | ***Mandatory*** |
| **paymentmethod** | Payment method | See Appendix 1 | AN(32) | |
| **lang** | Language on pay page | Supported languages: da, de, en, es, fi, fr, it, nl, no, pl, sv | AN(2) | |
| **currency** | Currency | ISO 4217 alphabetical upper-case (e.g. SEK) | AN(3) | Y |
| **amount** | Total amount in minor currency, including VAT | E.g. 1001 | N | Y |
| **vat** | VAT in minor currency | E.g. 200 | N | |
| **customerrefno** | Merchant ref.nr | Only a-z, A-Z, 0-9, and a few characters for legacy reasons eg - and _ | AN(64) | Y |
| **returnurl** | Return URL | Free text | AN(256) | Y |
| **callbackurl** | Callback URL | Free text | AN(256) | |
| **subscriptiontype** | Type of subscription | See recurring card payments | AN(64) | |
| **simulatorcode** | Desired statuscode. Only used for testing | E,g. 0 | N | |
| **externalpaymentref** | Reference to be sent to card acquirer | Only characters a-z,A-Z and 0-9 | AN(15) | |
| | | | | |
| **excludepaymentmethods** | Methods to exclude | List of <exclude> elements | Complex type | |
| **customer** | Customer data | See Appendix 1 | Complex type | |
| **orderrows** | Orderrows | List of <row> element | Complex type | |
| **payeralias** | Cellular phone number | E.g. 46701234567 | N | |
| **storedcardalias** | The alias for the stored card | GUID | AN(36) | Y |
| **showstorecarddialog** | Allow customer to store carddetails for later use. Defaults to false. | e.g. true | Boolean | |

<lang>
The available languages are:
da – Danish, de – German, en – English (incl. USA), es – Spanish, fi – Finnish, fr – French, it – Italian, no – Norwegian, pl – Polish, sv – Swedish
If the <lang> element is not used, the value will default to the language used in the browser.

Please note that amount is in minor currency, including VAT. For example, if the currency is SEK, 1001 in minor currency equals 10.01 SEK. For the currency of Island, the smallest unit is 1 ISK, which becomes “100” in the xml.

Example: Suppose that the product price is 4 SEK, excluding VAT. If the VAT percentage is 25%, the VAT will be 1 SEK, so the total amount will be 5 SEK. The elements for amount and VAT will be:
<amount>500</amount>
<vat>100</vat>

The <simulatorcode> element is used in the test environment only. It is used to simulate a response code, e.g. 0, which means SUCCESS. In the production environment this element is ignored.

<excludepaymentmethods>
This element holds a list of payment methods that are not to be presented as payment choices. Each payment method is contained in a separate exclude element.

| *Element name* | *Description* | *Value* | *Format* | *Mandatory* |
|---|---|---|---|---|
| **<exclude>** | Payment method | See Appendix 1 | AN(32) | |

Example: <excludepaymentmethods><exclude>SWISH</exclude><excludepaymentmethods>

<externalpaymentref>
This optional element is only valid for the payment method SVEACARDPAY. It is a reference to be sent to the acquirer to be displayed on reconciliations reports. Please note that it may only contain characters a-z, A-Z and 0-9 with a maximum length of 15. If any other characters are given, or length exceeds 15 characters, the payment request will be rejected.

<payeralias>
This element is only valid for the payment methods SWISH, VIPPS and MOBILEPAY. For SWISH and VIPPS, it is mandatory. The payeralias elements contains the customers cellular phone number in international format. For more information, see chapter SWISH, VIPPS and MOBILEPAYPAYMENTS.

<storedcardalias>
This element is optionally used for the payment method SVEACARDPAY and SVEACARDPAY_PF. The step where card data is entered is skipped if storedcardalias is used but 3DSecure authentication is still required.

<showstorecarddialog>
This element is only valid for the payment methods SVEACARDPAY and SVEACARDPAY_PF. If set to true a checkbox will be displayed on the card-payment page with a disclaimer stating that the customer can store their card in order to simplify upcoming payments.

**ORDER ROWS**
Order rows are used to send information about the products. Each order row is represented by a row element <row>. Row elements are all contained in the element <orderrows>.

| **<row>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* | *Mandatory* |
| **name** | Product name | Free text* | AN(256) | Y |
| **description** | Product description | Free text* | AN(512) | |
| **amount** | Price per item in minor currency, including VAT | e.g. 1001 | N | Y |
| **vat** | VAT per item in minor curreny | e.g. 200 | N | |
| **quantity** | Quantity | e.g. 5 | N | |
| **sku** | Article number | Free text* | AN(256) | Y |
| **unit** | Unit | Free text* | AN(64) | |
| **discountamount** | Discount amount in minor currency. | e.g. 200 | N | |
| **discountpercent** | Discount percent; numerical value of the ratio | e.g. 25.5, 32 | N | |
| **vatpercent** | Vat percent; numerical value of the ratio | e.g. 25.5, 32 | N | |

* Free text, no +, be careful with % because of URL encoding

Please note that **amount** is in minor currency, including VAT. For example, if the currency is SEK, 1001 in minor currency equals 10.01 SEK.
The **discountpercent** is the numerical value of the ration. For example: 25.5% will be sent as 25.5; 12%, will be sent as 12 …
The **vatpercent** is the numerical value of the ratio. For example: 7% will be sent as 7; 19.2% will be sent as 19.2

## PAYMENT RESPONSE MESSAGE

After performing the payment, the consumer is redirected back to the store's returnurl through a HTTP POST or a HTTP GET (a variation is when rescue payment is activated, see RESCUE PAYMENT for details). Please contact your premium user to choose the required response type.

The response contains the following parameters:

| **POST/GET Parameters** | |
|---|---|
| **response** | PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48cmVzcG9uc2U+PHRyYW5zYWN0aW9uIGlkPSI1MjM4NjliPjxwYXltZW50bWV0aG9kPkRCTk9SREVBU0U8L3BheW1lbnRtZXRob2Q+PG1lcmNoYW50aWQ+MTEwOTwvbWVyY2hhbnRpZD48Y3VzdG9tZXJyZWZubz5yZUdFUkdIUlJIdGg8L2N1c3RvbWVycmVmbm8+PGFtb3VudD41MDAwMTwvYW1vdW50PjxjdXJyZW5jeT5TRUs8L2N1cnJlbmN5PjwvdHJhbnNhY3Rpb24+PHN0YXR1c2NvZGU+MDwvc3RhdHVzY29kZT48L3Jlc3BvbnNlPg== |
| **mac** | 88cea01f3c1acfcbb40c532ea8be1b0027f77f7d92586619672720ac80d8c6f09bb0c51bbc8b07f2958d64633e36ee938a82fa532d393c0aaa297579829576f8 |
| **merchantid** | 1109 |

• response - Base64 encoded message (that needs to be decoded)

• mac – calculated mac (see section MAC)

• merchantid - your merchantid that you have received from the integrator

The response parameter contains the Base64 encoded XML message that contains a <response> root element. It contains information about the payment status.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transaction** | Optional attribute id containing the Transaction ID | Complex type | Complex type |
| **statuscode** | Status of the payment request | See Appendix 1 | N |

The <transaction> element contains some values:

| **<transaction>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **paymentmethod** | Payment method | See Appendix 1 | AN(32) |
| **merchantid** | ID of the merchant | | N |
| **customerrefno** | ID given by the merchant | Free text | AN(64) |
| **amount** | Total amount in minor currency, including VAT | e.g. 1001 | N |
| **currency** | Currency | ISO 4217 (e.g. SEK) | AN(3) |
| **subscriptionid** | Subscription ID (optional) | e.g. 2002 | N |
| **subscriptiontype** | Subscription type (optional) | e.g. RECURRINGCAPTURE | AN(64) |

Please note that amount is in minor currency, including VAT. For example, if the currency is SEK, 1001 in minor currency equals 10.01 SEK.

In some cases, the transaction will not be assigned an ID. This happens when the payment page is used and, for some reason, the transaction is stopped at an early stage, before the payment method is chosen by the customer. These orders are discarded and will not show up in the transaction list in the admin GUI.

Subscription ID and Subscription type are only returned if it's a subscription payment.

**RESCUE PAYMENT**
If rescue payment is activated for the merchant via a setting in the admin console, a payment that fails at the payment provider, or is cancelled by the customer, will result in a redirect to the PayPage instead of a redirect back to the merchant site. In this case, any selected or excluded payment methods are cleared from the original payment specification, and *all* active payment methods will be shown on the PayPage.

Every new payment attempt will result in a new transaction, hence there might be several failed transactions tied to one <customerrefno>.

**CALLBACK**
A callback is a message that is sent to the merchant. The sending of this message is initiated by the Payment Gateway.

- Certificates, if used on the merchant's system, should be signed by well-known Certificate Authorities.
- The callback URL must use one of these ports: 80 or 443.
- The hostname that should receive that callback must be communicated to Svea Bank for whitelisting.
- If the integrating party uses ip filtering, whitelisting or similar, ip range 193.13.207.0/24 and 193.105.138.0/24 must be allowed for incoming callback messages.

**CALLBACK, PAYMENT RESPONSE**
This callback is an additional response message that is sent to the callback URL of the merchant, with the same format as the response message. It is always sent as a POST. It is for merchants who want to be sure to receive a response even if the normal response message has not been sent. This could happen e.g. if a user that has been redirected to a bank closes their browser before completing the payment, or before being redirected back to the merchant. It may take several minutes before the callback message is sent. If enabled, a callback is always sent, even when the normal response message has also been sent. Even though it is not necessary, we recommend everyone to use the callback functionality if possible.

If a callback URL was not given in the payment message, the default callback URL that may have been configured for the merchant will be used instead. If no callback URL has been given anywhere, no callback will be sent.

**NOTE:** If callback urls are set dynamically (in the payment request) then all domains/hosts used must be communicated to Svea Bank for whitelisting.

**CALLBACK, STORED CARD ALIAS EXPIRED**
A merchant who uses the store card functionality may choose to handle the callback with information about expired card alias, i.e. the card alias cannot be used for further payments.
The callback URL is configured on merchant level in the payment gateway.
The callback URL must not use other ports than: 80 or 443. It is always sent as a POST.
Certificates, if used, on the merchant's system should be signed by well-known Certificate Authorities.
The hostname that should receive that callback must be communicated to Svea Bank for whitelisting.

If the integrating party uses ip filtering, whitelisting or similar, ip range 193.13.207.0/24 must be allowed for incoming traffic.

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<expiredcardalias>
   <storedcardalias>21ff4c0e-b398-42c3-9a60-f971dde92b95</storedcardalias>
</expiredcardalias>
```

**MOBILEPAY PAYMENTS**

The customer will be redirected to a page, MobilePay landing-page, where the user is asked to open the MobilePay app and sign the payment.

After the interaction in the MobilePay app is completed, the customer will be redirected to the url in the element <returnurl>.

If <payeralias> element is provided, the phone number will be prefilled in the MobilePay landing page. The phone number must be fully specified including country code, for example: 004512345678. If the element is not provided or in the wrong format, the payment will still be processed but the prefill will not work.

When integrating MOBILEPAY, the callback must be implemented.

A MOBILEPAY payment example

**Example XML-message**

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>MOBILEPAY</paymentmethod>
<payeralias>004512345678</payeralias>
<currency>DKK</currency>
<amount>500</amount>
<vat>100</vat>
<customerrefno>test_1612427935683</customerrefno>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-admin/admin/merchantresponsetest.xhtml<returnurl>
</payment>
```

**SWISH PAYMENTS**

The customer will be redirected to a page (with a spinner symbol) where the user is asked to open the Swish-app on the device where the app is installed and sign the payment. If the swish app is installed on the same device that was used for the payment the app can be opened directly by clicking a button “Open Swish”.

On this page there is a Cancel button. If clicked the user will be redirected to the <returnurl> with the statuscode 108 which means CANCELLED.

After the interaction in the swish app is completed the customer will be redirected to the url in the element <returnurl> provided the page is not actively closed.

The SWISH payment method requires the <payeralias> element to be present in the payment request. The <payeralias> element contains the customer’s cellular phone number in international format i.e. starting with country code followed by the cellular phone number, see below.

The <customerrefno> element is forwarded to the Swish payment system, to be used for reconciliation. The Swish payment system accepts only alphanumeric characters (A-Z,a-z, 0-9), and in

case the customerrefno contains other characters, these will be removed before sent to the Swish payment system. The length supported is 34 characters.

In case the customer closes the web browser or clicks cancel on the page (as described above) the payment will be in a pending state. The merchant will not get the payment if the redirect flow is broken.

It is highly recommended to implement the callback when using Swish.

A swish payment example

**Example XML-message**

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>SWISH</paymentmethod>
<payeralias>46701234567</payeralias>
<currency>SEK</currency>
<amount>500</amount>
<vat>100</vat>
<customerrefno>unique_id</customerrefno>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
</payment>
```

As an alternative a merchant can use SWISH via a payment facilitator (**SWISH_PF**).

**Difference between SWISH and SWISH_PF**

| SWISH | SWISH_PF |
|---|---|
| Merchant needs to sign a separate agreement with Swish | Merchant needs to sign a separate agreement with Svea Bank |
| Complicated onboarding process | Easy onboarding process |
| The consumers money goes from their account directly to merchants account | Looks the same for the consumer; their money goes from their account to a Svea account and from there to merchants account |
| Svea cannot offer Swish reports to merchant | Svea offers Swish reports to merchant |

**VIPPS PAYMENTS**

The customer will be redirected to a page, Vipps landing-page, where the user is asked to open the VIPPS-app on the device where the app is installed and sign the payment.

On this page there is a Cancel button. If clicked the user will be redirected to the <returnurl> with the statuscode 108 which means CANCELLED.

After the interaction in the vipps app is completed the customer will be redirected to the url in the element <returnurl> provided the page is not actively closed.

The VIPPS payment method will use a <payeralias> element if present in the payment request.
The <payeralias> element contains the customer's cellular phone number, and it must have exactly eight digits. If the <payeralias> doesn't match the criteria statuscode 329 meaning BAD_MOBILE will be returned.
If the there is no <payeralias> the user will, still be redirected to Vipps landing-page where the user must type in the phone number.

The <customerrefno> element is forwarded to the VIPPS payment system, to be used for reconciliation. The VIPPS payment system accepts only alphanumeric characters and minus sign (a-z,

0-9, -), and in case the customerrefno contains other characters, these will be removed. The length supported is 50 characters.
The <message> element is displayed in the VIPPS application; it is not mandatory.

When integrating VIPPS, the callback must be implemented.

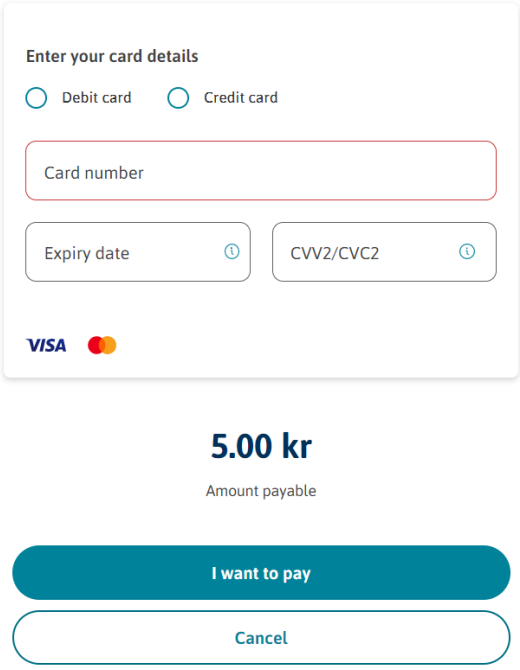
A VIPPS payment example

**Example XML-message**

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>VIPPS</paymentmethod>
<payeralias>48060316</payeralias>
<currency>NOK</currency>
<amount>500</amount>
<vat>100</vat>
<customerrefno>1612427935683</customerrefno>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-admin/admin/merchantresponsetest.xhtml<returnurl>
<message>Buying a mobile</message>
</payment>
```

## CARD PAYMENTS

One way to make card payments is to use the payment method SVEACARDPAY. The customer will be sent to a webpage maintained by the card payment service provider, where the customer is asked to fill in their card number and details.




If the SVEACARDPAY payment is approved, it will get the status AUTHORIZED. After midnight it will automatically be given the status CONFIRMED (unless the merchant is configured for manual confirm). After it is confirmed, it is automatically sent to the card payment service provider for capture. If this goes well, it will reach the final status SUCCESS, which means that the money has been transferred.

SVEACARDPAY supports the following currencies: SEK, DKK, EUR, GBP, NOK, PLN, USD, CHF.

The currency used must be supported by the acquiring agreement.

To choose language on the card payment page, the element <lang> can be used. Available language codes are: sv, da, de, en, fi, fr, no, pl.

SVEACARDPAY and SVEACARDPAY_PF require a number of customer data fields in the customer element. See Appendix 1 for all available customer data. To begin with, one must provide at least one of firstname, or lastname, or companyname. The other required fields are: address, city, zip, country, phone, and email.

For an example XML message, see Appendix 2.

CARD RESPONSE MESSAGE

For card payments, the <transaction> element of the response message contains some additional information:

| *Element name* | *Description* | *Value* | *Format* |
|---|---|---|---|
| **cardtype** | Type of card | e.g. VISA | AN |
| **maskedcardno** | Truncated card no | e.g. 444433xxxxxx3300 | AN(32) |
| **authcode** | Authorization number at EDB | Free text | AN(10) |
| **expiryyear** | Card expiry year | e.g. 13 | N |
| **expirymonth** | Card expiry month | e.g. 04 | N |
| **storedcardalias** | The alias for the stored card | GUID | AN(36) |

<storedcardalias>

This element is returned in case the customer has chosen to store his card details.

**RECURRING CARD PAYMENTS**

Our card payment provider has the capability to store the card details of the customer so that money can be drawn from the card by the merchant at regular intervals. E.g. for a membership fee or for a subscription to a magazine or service.

First, the merchant has to be configured for recurring payments. To enable recurring payments for a particular customer, a subscription must be created. This is done by posting an initial transaction so that the customer can fill in their card number to be stored at the card payment page of the card payment provider. Later payments can be created by posting to the webservice recur. To set up a subscription, add the element <subscriptiontype> to the XML body of the initial transaction. It can have one of two values:

RECURRING
RECURRINGCAPTURE

The merchant should use RECURRING if they only want to create a subscription, as a preparation for future payments, without immediately creating a payment. This is also called account verification, since it is a way to check if the customer's card is valid without drawing money from it. You should always set the initial amount to 0 since no money is to be transferred. A successful initial transaction will reach the "REGISTERED" status, and it will never be captured. To use RECURRING, the merchant must be set to allow account verification by someone at Svea.

The merchant should use RECURRINGCAPTURE if they also want to immediately create a new payment from the new subscription. I.e., two separate things will happen: a subscription will be created, and then a payment will be created from the subscription. The amount of the payment should be present in the request. If the initial payment is approved, it will reach the "AUTHORIZED" status, to be confirmed and captured later.

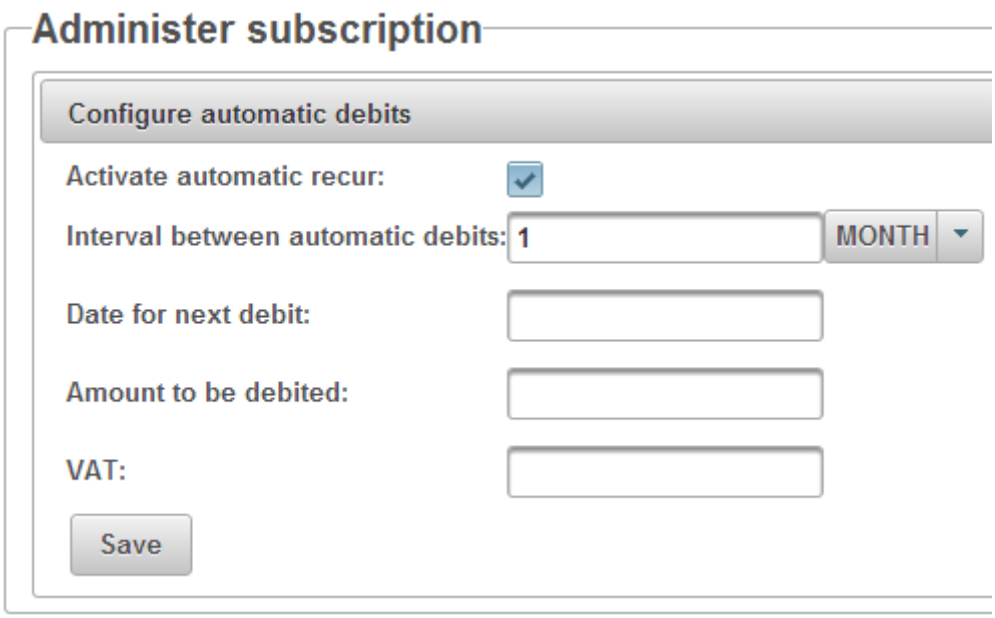
Regardless of the subscription type, if the initial transaction is successful, a subscription will be created. The corresponding subscriptionid is returned in the output XML body as <subscriptionid>. This ID is what the merchant should use when sending recur requests to our webservice. See the chapter about the webservice recur.

A successful recur operation that is made on the subscription will result in a normal SVEACARDPAY payment where the status is set to "AUTHORIZED", and it will be confirmed and captured later.

If no successful recur operation takes place, the subscription will become invalid after 12 months. Whenever a successful recur operation takes place, the subscription is extended to become valid for the coming 12 months.

For example payment XML, see Appendix 2.

**AUTOMATIC RECUR**
Card subscriptions created with the subscriptiontype RECURRING or RECURRINGCAPTURE can be configured to draw money from the card periodically, e.g. once a month. This is done manually on the subscription administration page. Open the subscription, tick the box, and fill in the rest of the information, including the amount that is to be drawn from the card every time. Automatic recur can be turned off again by unticking the box.




Merchants who expect so many customers that manual administration will be impractical may implement their own code for calculating when recurs are to be posted to our webservice.

**PAYMENT FACILITATORS**
A payment facilitator is a super merchant that can have sub merchants and can handle card payment acquirer agreements for those sub merchants, to simplify the integration. At the time being, only two payment facilitators are planned to be set up, internally in Svea.

To make payments through a payment facilitator, one first needs a specific agreement for this. The payment is made using the payment method SVEACARDPAY_PF, which is based on SVEACARDPAY, and the end customer will not notice any difference.

SVEACARDPAY_PF has the same requirements as SVEACARDPAY when it comes to customer data.

For an example payment, see Appendix 2.

**STORED CARDS**
When a card payment is made, it is possible to store the card data of the customer for future use. If the same customer makes a new payment, this data can be retrieved, so that the customer doesn't need to fill in their card number etc. the second time, thus speeding up the payment process and making it more convenient for the customer. This works for both SVEACARDPAY and SVEACARDPAY_PF.

The shop must first be configured for this in Payment Gateway. Contact Svea if you want this.

It must also be activated on the individual payment, and it can be controlled on a payment by payment basis. This is done by adding the following line in the payment request on the initial payment:
<showstorecarddialog>true</showstorecarddialog>
This is an optional element in the payment request. See the chapter THE HTTP PAYMENT REQUEST STRUCTURE.

When the showstorecarddialog has been set to true on a card payment, the card payment page will display an additional checkbox with a text asking the customer if they agree to storing their card details. It also says that the customer can contact Svea to cancel this consent. The card data of the customer will only be stored if the customer checks the checkbox.

If the customer checks the checkbox, the payment response message will contain an extra element called storedcardalias. For more info about this element, see the chapter CARD PAYMENTS, specifically the section called CARD PESPONSE MESSAGE.

The storedcardalias is an alias that represents the card number, and it can be used to make payments with the same card. If the same customer is about to make a new payment, the shop can choose to add this storedcardalias to the payment message. The new payment can then be completed without the need for the customer to enter the card number and data again. The storedcardalias element is described in the chapter THE HTTP PAYMENT REQUEST STRUCTURE.

If the alias is not used within a year, it will expire. Each time the alias is used, the expiry date is renewed, so that the alias can be used for a year from that point. If you want to check if the alias is still valid, you can use the webservice getstoredcard. See the chapter GETSTOREDCARD.

## WEBSERVICES

Send a request consisting of the following parameters:

| HTTP POST | |
|---|---|
| ***Parameter name*** | *Description* |
| **message** | Base64 encoded XML message |
| **merchantid** | Merchant ID |
| **mac** | Checksum |

• merchantid - is the store ID that you received from your integrator.
• message - contains the Base64 encoded XML you have generated as described in the section the query request structure.
• mac - includes the control numbers that have been developed in order to verify the sender of the call. In order to generate this you must have received a secret word from your integrator. See more under section MAC.

THE WEBSERVICE REQUEST
A request contains the three parameters: message, merchantid and mac. The message has to be Base64 encoded.

**Example of a POST:**

```
<form action="https://webpaypaymentgatewaystage.svea.com/webpay/rest/payment" method="post">
<input type="hidden" name="mac" value="dd57e26612b586a1d55efb91b3bc21902e3ae09499bf074d0c0624820fcdc61b
243f24b4f3c9b1feac06ffd3cfe14fb8165a7c83d9fcd550196f7e7fb20e43c2" />
<input type="hidden" name="merchantid" value="1109" />
<input type="hidden" name="message" value="PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4NCjxjYXB0
dXJlPg0KPHRyYW5zYWN0aW9uaWQ+NTIxNjc3PC90cmFuc2FjdGlvbmlkPg0KPC9jYXB0dXJlPg==" />
<input type="submit" value="payment" />
</form>
```

THE WEBSERVICE RESPONSE
The response contains the three parameters: message, merchantid and mac.The message is Base64 encoded. The contents of the message is different for different webservices.

**Example of response:**

```
<?xml version="1.0" encoding="UTF-8"?><response><message>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRG
LTgiPz48cmVzcG9uc2U+PHRyYW5zYWN0aW
9uIGlkPSI1MjE4MzIiPjxjdXN0b21lcnJlZm5vPjU2NTUyNTQ1UkI8L2N1c3RvbWVycmVmbm8+PC90cmFuc2FjdGlvbj48c3RhdHV
zY29kZT4wPC9zdGF0dXNjb2RlPjwvcmVzcG9uc2U+</message>
<merchantid>1109</merchantid>
<mac>70d118d8ff816fbc7244305ada2f335719dc5a680c91f00b88679b348f007ab5cec746d4fd7002e3bfccb8a72c0f524
fe28f2a9bd9efbc392fea8a8386625ae4</mac>
</response>
```

**ANNUL**

The annul request can be used to cancel a payment before it has been captured. It can only be performed on card, invoice, or payment plan transactions having the status AUTHORIZED or CONFIRMED.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/annul**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/annul**

THE ANNUL REQUEST STRUCTURE
The call is structured in XML with the root element <annul>. It contains the transaction id of the transaction to annul.

| **<annul>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **transactionid** | Transaction ID | | N | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<annul>
<transactionid>521677</transactionid>
</annul>
```

THE ANNUL RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transaction** | Transaction ID | Complex type | Complex type |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

The <transaction> element contains an attribute id representing the Transaction ID. The element contains the following information.

| **<transaction>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **customerrefno** | Order ID | Free text | AN(64) |

## CANCELRECURSUBSCRIPTION

This request inactivates an existing recur subscription so that no more recurs can be made on it.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/cancelrecursubscription**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/cancelrecursubscription**

THE REQUEST STRUCTURE
The call is structured in XML with a root element <cancelrecursubscription>. This element contains the subscription ID (not transaction ID) of the subscription you want to inactivate.

| **<cancelrecursubscription>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **subscriptionid** | Subscription ID | | N | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<cancelrecursubscription>
<subscriptionid>1234</subscriptionid>
</cancelrecursubscription>
```

THE CANCELRECURSUBSCRIPTION RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

**CONFIRM**

The confirm request is intended for card transactions. It can only be performed on card transactions having the status AUTHORIZED. This will result in a CONFIRMED transaction that will be captured (settled) on the given capture date. Confirm is mainly used by merchants who are configured to confirm their transactions themselves. Otherwise the transactions are confirmed automatically.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/confirm**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/confirm**

THE CONFIRM REQUEST STRUCTURE
The call is structured in XML with the root element <confirm>. The <confirm> element contains the transaction ID of the transaction to confirm, as well as the capture date. The capture date tells when to capture the transaction. The date format used is the ISO-8601 extended date format, a.k.a. E8601DAw ("yyyy-MM-dd").

| **<confirm>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **transactionid** | Transaction ID | | N | Y |
| **capturedate** | Capture date | | D | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<confirm>
<transactionid>521527</transactionid>
<capturedate>2011-09-21</capturedate>
</confirm>
```

THE CONFIRM RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transaction** | Transaction ID | Complex type | Complex type |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

The <transaction> element contains an attribute id representing the Transaction ID. The element also contains the following information.

| **<transaction>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **customerrefno** | Order ID | Free text | AN(64) |

## CREDIT

The credit request can be used to return money to the customer. Only transactions that have reached the status SUCCESS can be credited.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/credit**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/credit**

THE CREDIT REQUEST STRUCTURE
The call is structured in XML with the root element <credit>. The <credit> element should contain the transaction ID and the amount to credit.

| **<credit>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **transactionid** | Transaction ID | | N | Y |
| **amounttocredit** | The amount that you want to credit in minor currency | E.g. 1001 | N | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<credit>
<transactionid>521527</transactionid>
<amounttocredit>100</amounttocredit>
</credit>
```

THE CREDIT RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transaction** | Transaction ID | Complex type | Complex type |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

The <transaction> element contains an attribute id representing the Transaction ID. The element contains the following information.

| **<transaction>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **customerrefno** | Order ID | Free text | AN(64) |

## GETPAYMENTMETHODS

This request is used to fetch a list of the payment methods available for the merchant.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/getpaymentmethods**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/getpaymentmethods**

THE GETPAYMENTMETHODS REQUEST STRUCTURE
The call is structured in XML and you need a root element <getpaymentmethods>.

| **<getpaymentmethods>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<getpaymentmethods></getpaymentmethods>
```

THE GETPAYMENTMETHODS RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **paymentmethods** | Containing an array of paymentmethod elements | | array |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

The <paymentmethods> element contains the following information.

| **<paymentmethods>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **paymentmethod** | Payment method name | See Appendix 1 | enum |

## GETRECONCILIATIONREPORT

The getreconciliationreport request will return a list of the transactions that were captured or credited at a specified date or during a date span.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/getreconciliationreport**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/getreconciliationreport**

THE GETRECONCILIATIONREPORT REQUEST STRUCTURE
The call is structured in XML with the root element <getreconciliationreport>.

| **<getreconciliationreport>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **date** | Date | Ex: 2011-11-07 | AN | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<getreconciliationreport>
<date>2011-11-07</date>
</getreconciliationreport>
```

If a span of dates are desired the structure will be:

| **<getreconciliationreport>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **fromdate** | Date | Ex: 2014-11-01 | AN | Y |
| **todate** | Date | Ex: 2014-11-30 | AN | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<getreconciliationreport>
<fromdate>2014-11-01</fromdate>
<todate>2014-11-30</todate>
</getreconciliationreport>
```

THE GETRECONCILIATIONREPORT RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **reconciliation** | An array of reconciliationtransaction types | Complex type array | Complex type array |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

Content of the <reconciliation> tag

| **<reconciliation>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **reconciliationtransaction** | Information about a reconciliationtransaction | Complex type | Complex type |

Content of the <reconciliationtransaction> tag

| **<reconciliationtransaction>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transactionid** | Transaction ID | | N |
| **customerrefno** | Merchant's referense | | AN |
| **paymentmethod** | Payment method | See Appendix 1 | AN(32) |
| **amount** | Amount in minor currency. + or - | Eg 1900 or -1900 | N |
| **time** | Time of the transaction | yyyy-MM-dd HH:mm:ss z | AN |

## GETSTOREDCARD

This request can be used to check if the alias of a stored card is valid. The alias can only be used to create payments if it is valid. If the alias is not used within a year, it will expire. Each time the alias is used, the expiry date is renewed, so that the alias can be used for a year from that point.

Please only use this web service if needed. Repetitive polling is not allowed.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/getstoredcard**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/getstoredcard**

THE GETSTOREDCARD REQUEST STRUCTURE
The call is structured in XML with the root element <getstoredcard>.

| **<getstoredcard>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **storedcardalias** | The alias for the stored card | GUID | AN(36) | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<getstoredcard>
  <storedcardalias>21ff4c0e-b398-42c3-9a60-f971dde92b95</storedcardalias>
</getstoredcard>
```

THE GETSTOREDCARD RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **statuscode** | Status of the request | See Appendix 1 | N |
| **cardvalid** | Tells if the stored card alias is valid | true or false | Boolean |

## LOWERAMOUNT

The loweramount operation is intended for card transactions having the status AUTHORIZED or CONFIRMED. It can be used e.g. if the merchant is unable to deliver all of the items that was ordered by the customer, and wants to lower the total amount to pay. If the <amounttolower> is equal to the authorized amount, the transaction status will change to annulled.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/loweramount**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/loweramount**

THE LOWERAMOUNT REQUEST STRUCTURE
The call is structured in XML with the root element <loweramount>. The <loweramount> element contains the transaction id of the transaction you want to annul.

| **<loweramount>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **transactionid** | Transaction ID | | N | Y |
| **amounttolower** | The amount you wish to lower the transaction by. | | N | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<loweramount>
<transactionid>521677</transactionid>
<amounttolower>2000</amounttolower>
</loweramount>
```

THE LOWERAMOUNT RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transaction** | Transaction ID | Complex type | Complex type |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

The <transaction> element contains an attribute id representing the Transaction ID. The element also contains the following information.

| **<transaction>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **customerrefno** | Order ID | Free text | AN(64) |

## PREPAREPAYMENT

The preparepayment request can be used to let Svea Bank do the necessary pre-processing of a payment in advance. This allows the merchant to send the customer to the payment gateway through a HTTP GET instead of a HTTP POST. To complete the payment, the customer visits a certain URL containing the ID of the prepared payment. The customer can either be redirected to this URL by the merchant, or the merchant can give it to the customer, e.g. in an email, so that they can visit it themselves. A prepared payment is valid up to one hour after creation.

Preparepayment is not the standard method for creating payments, and should only be used when there is a specific reason to use it.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/preparepayment**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/preparepayment**

THE PAYMENT REQUEST STRUCTURE
The request is structured in XML and you need a root element <payment>. The <payment> element contains information needed to create a new payment.
Information needed is the same as in a request to our payment gateway except two additional required parameters.

| **Additional parameters** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **lang** | Language according to ISO 639-1 | Ex: sv | AN | Y |
| **Ipaddress** | Customer's IP address | | AN | Y |

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<currency>SEK</currency>
<amount>1000</amount>
<vat>200</vat>
<lang>sv</lang>
<ipaddress>127.0.0.1</ipaddress>
<customerrefno>unique_id</customerrefno>
 <returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
<ssn>460509-2222</ssn>
<orderrows>
    <row>
        <name>ASUS P8P67 PRO Socket-1155</name>
        <amount>1000</amount>
        <description>ATX P67 DDR3 3xPCIe(2.0)x16 CFX SLI SATA 6Gb/s</description>
        <vat>200</vat>
        <quantity>1</quantity>
        <sku>SK-54f-3S23-AB</sku>
        <unit>st</unit>
    </row>
</orderrows>
</payment>
```

THE PREPAREPAYMENT RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **preparedpayment** | Information about your prepared payment | Complex type | Complex type |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

Content of the <preparedpayment> tag

| **<preparedpayment>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **id** | Prepared payment id | e.g 7 | N |
| **created** | Date when the object was created | e.g. 2011-09-27 16:55:01.21 | AN |

When redirecting the customer to let them complete their payment, the prepared payment ID should be appended to the URL.

Example URL for a test redirect:
**https://webpaypaymentgatewaystage.svea.com/webpay/preparedpayment/123456**

Example URL for a production redirect:
**https://webpaypaymentgateway.svea.com/webpay/preparedpayment/123456**

## QUERY

Query is used to get information about a specific transaction. To do this one must know either the transactionid set by Svea Bank, or the customerrefno that has been set by the merchant. These have different URL: s.

Please only use this web service when needed. Repetitive polling is not allowed.

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/querytransactionid**
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/querycustomerrefno**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/querytransactionid**
**https://webpaypaymentgateway.svea.com/webpay/rest/querycustomerrefno**

### THE QUERY REQUEST STRUCTURE

The call is structured in XML and you need a root element <query>. The <query> element contains either the orderid or transaction ID that you want to query.

Query based on the order ID:

| **<query>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **customerrefno** | Order ID | | AN(32) | Y |

**Example:**

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
<customerrefno>temping1238u96896</customerrefno>
</query>
```

Query based on the transaction ID

| **<query>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **transactionid** | Transaction ID | | N | Y |

**Example:**

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
<transactionid>521527</transactionid>
</query>
```

### THE QUERY RESPONSE

The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transaction** | Transaction ID | Complex type | Complex type |
| **statuscode** | Status of the webservice request | See Appendix 1 1 | N |

The <transaction> element contains an attribute id representing the Transaction ID. The element also contains the following information.

| **<transaction>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **<customerrefno>** | Order id | Free text | AN(64) |
| **<merchantid>** | Merchant id | | N |
| **<status>** | Latest transaction status. | e.g SUCCESS | AN(32) |
| **<amount>** | Total amount in minor currency, including VAT | e.g 1000 | N |
| **<currency>** | Currency | ISO 4217 alphabetic (e.g. SEK) | AN(3) |
| **<vat>** | VAT in minor currency | e.g. 1000 | N |
| **<capturedamount>** | Captured amount | e.g. 1000 | N |
| **<authorizedamount>** | Authorized amount | e.g. 1000 | N |
| **<created>** | Timestamp when transaction was created | e.g. 2011-09-27 16:55:01.21 | |
| **<creditstatus>** | Status of the last credit attempt | | AN(32) |
| **<creditedamount>** | Total amount that has been credited | e.g. 1000 | N |
| **<merchantresponsecode>** | Last statuscode response returned to merchant. See Appendix 1 | | N |
| **<paymentmethod>** | Paymentmethod. See Appendix 1 | | AN(32) |
| **<orderrows>** | Element containing <row> elements. | | Complex type |
| **<capturedate>** | When the transaction was captured | e.g. 2011-09-27 16:55:01.21 | |

If you are querying a card payment you will get more information within the <transaction> element.

| *Element name* | *Description* | *Value* | *Format* |
|---|---|---|---|
| **<cardtype>** | Brand of the card | e.g. VISA or MASTERCARD | AN(64) |
| **<eci>** | Enrollement status from MPI. If the card is 3Dsecure enabled or not. | | N |
| **<mdstatus>** | Value calculated from eci as requested by acquiring bank | | N |
| **<expiryyear>** | Expire year of the card | | AN(4) |
| **<expirymonth>** | Expire month of the card | | AN(2) |
| **<chname>** | Cardholder name as entered by cardholder | | AN(100) |
| **<authcode>** | EDB authorization code | | AN(10) |

## RECUR

The recur request creates a new transaction from a subscription. (As we have seen, a subscription is created by making a transaction that has a subscriptiontype.)

URL to POST for test requests:
**https://webpaypaymentgatewaystage.svea.com/webpay/rest/recur**

URL to POST for production requests:
**https://webpaypaymentgateway.svea.com/webpay/rest/recur**

THE RECUR REQUEST STRUCTURE
The request XML should have the root element <recur>. The <recur> element may contain the following elements.

| **<recur>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Content* | *Value* | *Format* | *Mandatory* |
| **customerrefno** | The new unique customerrefno | E.g. 10003 | AN | Y |
| **subscriptionid** | The subscription id | E.g. 1005 | N | Y |
| **currency** | Currency | E.g. SEK | AN | N |
| **amount** | The amount to recur in minor currency | E.g. 1001 | N | Y |
| **vat** | VAT amount | E.g. 250 | N | N |
| **orderrows** | Order rows | List of <row> elements | Complex type | N |

If the subscription type is RECURRING or RECURRINGCAPTURE, the currency for the recur request must be the same as the currency in the initial transaction. To avoid errors, the currency parameter can be omitted.

The amount on the recur request does not need to be the same as the amount on the initial transaction that created the subscription. It can be smaller or larger. VAT is not needed on the recur request.

**Example of XML structure:**

```
<?xml version="1.0" encoding="UTF-8"?>
<recur>
<customerrefno>10003</customerrefno>
<subscriptionid>2003</subscriptionid>
<currency>SEK</currency>
<amount>600</amount>
</recur>
```

THE RECUR RESPONSE
The response is Base64 encoded. See the chapter WEBSERVICES for an example.

The Base64 decoded message contains a <response> tag.

| **<response>** | | | |
|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* |
| **transaction** | Transaction details | Complex type | Complex type |
| **statuscode** | Status of the webservice request | See Appendix 1 | N |

The <transaction> element contains an information regarding the transaction that was created as a result of the recur operation. The element contains a transaction id as attribute as well as the following information.

| **<transaction>** | | |
|---|---|---|
| ***Element name*** | *Description* | *Format* |
| **customerrefno** | Order ID | AN(64) |
| **paymentmethod** | Payment method | AN |
| **merchantid** | Merchant id | N |
| **amount** | Total amount in minor currency | N |
| **currency** | Currency | N |
| **cardtype** | Card type | AN |
| **maskedcardno** | Masked card number | N |
| **expirymonth** | Expiry month | N(2) |
| **expiryyear** | Expiry year | N(2) |
| **authcode** | Authorization code | N |
| **subscriptionid** | Subscription id | N |

## MAC

In this context, MAC stands for Message Authentication Key. In order to verify a request, the merchant needs to create a MAC-string for each request (payment or webservice).

The MAC is calculated by taking the Base64 encoded XML string and appending the "secret word". Hash the resulting string by using SHA-512.

## FORMULA

MAC = SHA512Hash(Base64Encoded(xmlMessage) + secretWord);

Step1

**Example of XML-message:**

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>DBSHBSE</paymentmethod><currency>SEK</currency>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
<amount>50001</amount><customerrefno>testingnrglkjhhohjf</customerrefno>
</payment>
```

Step 2

**Example of Base64 encoded XML-message**

PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4gPHBheW1lbnQ+IDxwYXltZW50b
WV0aG9kPkRCU0hCU0U8L3BheW1lbnRtZXRob2Q+PGN1cnJlbmN5PlNFSzwvY3VycmVuY3k+ID
xyZXR1cm51cmw+aHR0cHM6Ly90ZXN0LnN2ZWFla29ub21pLnNlL3dlYnBheS1hZG1pbi9hZG1pbi
9tZXJjaGFudHJlc3BvbnNldGVzdC54aHRtbDwvcmV0dXJudXJsPiA8YW1vdW50PjUwMDAxPC9hb
W91bnQ+PGN1c3RvbWVycmVmbm8+dGVzdGluZ25yZ2xramhob2hqZjwvY3VzdG9tZXJyZWZubz
4gPC9wYXltZW50Pg==

Step 3

**Example of 'secret word' received from your integrator:**

df74ce933f1d367d4100b4d34ad6970760c6040e13d49b94b36bd81239c2c3a724435ab5dc4e1065c86
1944f9a1e56fa1f53f1cd22d71564e69d5256fba24d43

Step 4

**Example of generated MAC value:**

e70959517368c0873481979922b6d02ec15fba44e27e71abe56eea15c97ec6c970b7f19d2ea3baecec6
aacc1f3d8a0096f2231a37352d228e46e0055f5281586

## APPENDIX 1

### PAYMENT METHODS

| Payment method | Description | Country availability |
|---|---|---|
| **MOBILEPAY** | Card payments | Finland, Denmark |
| **SVEACARDPAY** | Card payments, Svea Bank | All |
| **SVEACARDPAY_PF** | SVEACARDPAY via a payment facilitator | All |
| **SWISH** | Bank payments | Sweden |
| **SWISH_PF** | Bank payments via a payment facilitator | Sweden |
| **VIPPS** | Card payments | Norway |

## WEBSERVICES PER PAYMENT METHOD

| | annul | confirm | confirm partial | credit | loweramount | query |
|---|---|---|---|---|---|---|
| MOBILEPAY | X | X | X | X | | X |
| SVEACARDPAY (CARD) | X | X | X | X | X | X |
| SVEACARDPAY_PF | X | X | X | X | X | X |
| SWISH | | | | X | | X |
| SWISH_PF | | | | X | | X |
| VIPPS | X | X | X | X | X | X |

## INDEPENDENT WEBSERVICES

These webservices are not tied to any particular payment method:

getpaymentmethods
getreconciliationreport
preparepayment

## THE CUSTOMER ELEMENT

For any payment method, the shop may add a customer element to the payment message. For some payment methods this is required, e.g. for card payments (SVEACARDPAY and SVEACARDPAY_PF). In some cases it is unnecessary, e.g. for SWISH and TRUSTLY.

The customer element may contain the following parameters:

| **<customer>** | | | | |
|---|---|---|---|---|
| ***Element name*** | *Description* | *Value* | *Format* | *Mandatory* |
| **ssn** | Social security number or organization number | e.g. 123456-7890 | AN(13) | |
| **firstname** | First name of customer | Free text* | AN(128) | |
| **lastname** | Last name of customer | Free text* | AN(128) | |
| **initials** | Initials of customer | Free text* | AN(32) | |
| **address** | Address | Free text* | AN(256) | |
| **address2** | c/o | Free text* | AN(256) | |
| **housenumber** | House number | Free text* | AN(32) | |
| **zip** | Postal code | e,g. 12345 | N | |
| **city** | City | Free text* | AN(45) | |
| **country** | Country code | e.g. SE | AN(45) | |
| **phone** | Phone number | e.g. 555-123456 | AN(45) | |
| **email** | Email | Free text* | AN(256) | |
| **companyname** | Use if customer is company | Free text* | AN(128) | |
| **companyid** | Id of the company | | N | |
| **vatnumber** | Vat number of the company | e.g. DE123456789 | AN(32) | |
| **industrycode** | Activity classification. For government use 99998, private person use 99999. | e.g. 47523 | N | |
| **iscompany** | If the customer is a company. Defaults to false | e.g. true | Boolean | |
| **unknowncustomer** | Deprecated | e.g. true | Boolean | |

* Free text, no +, be careful with % because of URL encoding

## TRANSACTION STATUS

Below are all possible states that a transaction can be in. During the lifetime of a transaction, the status may change many times, most often ending up as either SUCCESS or FAILED.

The status can be obtained in several different ways, e.g. by looking in the Transaction Details window, or by making a request to the query webservice. The range of possible statuses for a specific transaction depends on its payment method.

| Status | Description |
|---|---|
| **NEW** | The request was authenticated and accepted (syntactically correct) |
| **RECEIVED** | The request is saved, ready to be processed |
| **VALID** | Request has passed all the antifraud checks and all the checks specific to the current payment method (e.g.: currency is not allowed, phone number format, missing mandatory field...) |
| **PENDING** | Payment has been initialized; waiting for the external provider/bank to respond |
| **REGISTERED** | Recurring payment was authorized by the end-user [only card payments] |
| **RESPONSE_RECEIVED** | Callback from the external provider/bank was received |
| **FAILED** | Request did not pass the validation checks (e.g.antifraud, mandatory fields, not allowed value ...); Unexpected error while processing payment (internal or external error); Payment was not accepted by the external provider/bank |
| **ERROR** | Payment status could not be retrieved from the external provider/bank system. |
| **CANCELLED** | Payment was cancelled before payer authorized it |
| **ANNULLED** | Payment was cancelled after payer authorized it |
| **CLEARING** | Payer authorized the request; waiting for the transfer to be confirmed [only Trustly] |
| **AUTHORIZED** | Payer authorized the request. The amount is blocked but not withdrawn from the end-user account [only card payments] |
| **CONFIRMED** | Payment was confirmed; ready to be captured. The confirmation can be done automatically or manually by the merchant [only card payments] |
| **CAPTPENDING** | The process of withdrawing money from end-user account started [only card payments] |
| **CAPTFAILED** | The withdraw could not be performed [only card payments] |
| **SUCCESS** | Payment completed; money was withdrawn from the end-user account |

## STATUS CODES

Status codes are also used in the response message to webservice calls, to convey the result. A webservice call that was successful will have the statuscode SUCCESS in the response message. If the call failed, another status code will be given to act as an error code.

Another way to view status codes is to see them as extra information that clarifies the status of a transaction. All but one of them are error codes and those will most often be seen together with the status FAILED, to explain what went wrong. A transaction with the status SUCCESS will most often have the status code SUCCESS.

| Code | Name | Description |
|---|---|---|
| **0** | SUCCESS | Request performed successfully |
| **1** | REQUIRES_MANUAL_REVIEW | Request performed successfully but requires manual review by merchant |
| **100** | INTERNAL_ERROR | Internal system error |
| **101** | XMLPARSEFAIL | Invalid XML |
| **102** | ILLEGAL_ENCODING | Invalid encoding |
| **104** | ILLEGAL_URL | Invalid URL |
| **105** | ILLEGAL_TRANSACTIONSTATUS | Invalid transaction status |
| **106** | EXTERNAL_ERROR | Failure at third party e.g. the bank |
| **107** | DENIED_BY_BANK | Transaction rejected by bank |
| **108** | CANCELLED | Transaction cancelled |
| **110** | ILLEGAL_TRANSACTIONID | Invalid transaction ID |
| **111** | MERCHANT_NOT_CONFIGURED | Merchant not configured |
| **112** | MERCHANT_NOT_CONFIGURED_AT_BANK | Merchant not configured at bank |
| **113** | PAYMENTMETHOD_NOT_CONFIGURED | Payment method not configured for merchant |
| **114** | TIMEOUT_AT_BANK | Timeout at bank |
| **115** | MERCHANT_NOT_ACTIVE | The merchant is inactivated |
| **116** | PAYMENTMETHOD_NOT_ACTIVE | The payment method is inactivated |
| **117** | ILLEGAL_AUTHORIZED_AMOUNT | Amount cannot be authorized |
| **118** | ILLEGAL_CAPTURED_AMOUNT | Amount cannot be captured |
| **119** | ILLEGAL_CREDITED_AMOUNT | Amount cannot be credited |
| **124** | EXCEEDS_AMOUNT_LIMIT | Amount exceeds the limit |
| **126** | TRANSACTION_NOT_BELONGING_TO_MERCHANT | Transaction does not belong to merchant |
| **127** | CUSTOMERREFNO_ALREADY_USED | Customer reference number already used in another transaction |
| **128** | NO_SUCH_TRANS | Transaction does not exist |
| **129** | DUPLICATE_TRANSACTION | More than one transaction found for the given customerrefno |
| **130** | ILLEGAL_OPERATION | Operation not allowed for the given payment method |
| **131** | COMPANY_NOT_ACTIVE | Company inactivated |
| **133** | SUBSCRIPTION_NOT_ACTIVE | Subscription not active |
| **134** | SUBSCRIPTION_NOT_SUPPORTED | Payment method doesn't support subscriptions |
| **135** | ILLEGAL_DATE_FORMAT | Invalid date format |
| **136** | ILLEGAL_RESPONSE_DATA | Invalid response data |
| **138** | CURRENCY_NOT_CONFIGURED | Currency not configured |
| **139** | CURRENCY_NOT_ACTIVE | Currency not active |
| **140** | CURRENCY_ALREADY_CONFIGURED | Currency is already configured |
| **142** | NO_VALID_PAYMENT_METHODS | No valid payment methods |
| **143** | CREDIT_DENIED_BY_BANK | Credit denied by bank |
| **144** | ILLEGAL_CREDIT_USER | User is not allowed to perform credit operation |
| **146** | CUSTOMER_NOT_FOUND | Customer not found |
| **147** | AGE_LIMIT_EXCEEDED | E.g. the transaction is too old |
| **148** | BROWSER_NOT_SUPPORTED | The browser version is too old |
| **149** | PENDING | The request is still being processed |

| | | |
|---|---|---|
| **150** | CREDIT_PENDING | The credit could not be handled instantly. It is put in a queue it will be processed in due time. |
| **152** | CARD_CLOSED | Card is closed |
| **301** | BAD_TRANSACTION_ID | Invalid transaction ID |
| **303** | BAD_MERCHANT_ID | Invalid merchant ID |
| **304** | BAD_LANG | Invalid language |
| **305** | BAD_AMOUNT | Invalid amount |
| **306** | BAD_CUSTOMERREFNO | Invalid customerrefno |
| **307** | BAD_CURRENCY | Invalid currency |
| **308** | BAD_PAYMENTMETHOD | Invalid payment method |
| **309** | BAD_RETURNURL | Invalid returnurl |
| **311** | BAD_MAC | Invalid mac |
| **316** | BAD_CARDNUMBER_OR_CARDTYPE_NOT_CONFIGURED | Card type not configured for merchant |
| **317** | BAD_SSN | Invalid ssn |
| **318** | BAD_VAT | Invalid vat |
| **319** | BAD_CAPTURE_DATE | Invalid capture date |
| **320** | BAD_CAMPAIGN_CODE | Invalid campaign code |
| **321** | BAD_SUBSCRIPTION_TYPE | Invalid subscription type |
| **322** | BAD_SUBSCRIPTION_ID | Invalid subscription ID |
| **323** | BAD_BASE64 | Invalid Base64 |
| **325** | BAD_CALLBACKURL | Invalid callbackurl |
| **326** | THREE_D_CHECK_FAILED | 3D check failed |
| **327** | CARD_NOT_ENROLLED | Card not enrolled in 3D Secure |
| **328** | BAD_IPADDRESS | Provided IP address is invalid |
| **329** | BAD_MOBILE | Invalid mobile phone number |
| **330** | BAD_COUNTRY | Invalid country |
| **331** | THREE_D_CHECK_NOT_AVAILABLE | Merchant 3D configuration invalid |
| **332** | TIMEOUT | Timeout at Svea |
| **333** | BAD_PERIOD | The reconciliation period is invalid |
| **334** | BAD_ADDRESS_ID | AddressSelector is not valid for this CountryCode |
| **335** | BAD_CUSTOMER_DATA | The supplied customer data is invalid |
| **336** | BAD_UNIT | Invalid unit |
| **337** | BAD_EXTERNAL_PAYMENT_REF | Invalid external payment reference |
| **338** | BAD_STOREDCARDALIAS | Invalid stored card alias |
| **339** | STOREDCARDALIAS_NOT_ACTIVE | Stored card alias inactive |
| **340** | STORED_CARDS_NOT_ENABLED | |
| **341** | ONLY_DEBIT_CARDS_ALLOWED | |
| **342** | TRANSACTION_ALREDY_IN_PROGRESS | |
| **343** | BAD_CANCELURL | Invalid Cancel url |
| **344** | BAD_SIMULATOR_CODE | Invalid value for Simulator Code |
| **345** | BAD_ORDER_ROW | Invalid format for Order Row |
| **346** | BAD_PAYER_ALIAS | Invalid format for Payer Alias-Phone number |
| **347** | BAD_SHOW_STORE_CARD_DIALOG | Invalid value for Show Store Card Dialog |
| **351** | BAD_CARD_BRAND | Operation is not supported for card brand |

| | | |
|---|---|---|
| **360** | CARD_CAPTURE_LOCKED | Operation cannot be perfomed because the payment is locked for capturing. The lock is placed around midnight and will be removed after several minutes |
| **361** | BAD_CAPTUREREQUESTID | Capture request id is missing, not in the correct format or it is not unique |
| **373** | SERVICE_UNAVAILABLE_TRY_LATER | Temporary problem, try later |
| **500** | ANTIFRAUD_CARDBIN_NOT_ALLOWED | Antifraud - cardbin not allowed |
| **501** | ANTIFRAUD_IPLOCATION_NOT_ALLOWED | Antifraud – iplocation not allowed |
| **502** | ANTIFRAUD_IPLOCATION_AND_BIN_DOESNT_MATCH | Antifraud – ip-location and bin does not match |
| **503** | ANTIFRAUD_MAX_AMOUNT_PER_IP_EXCEEDED | Antofraud – max amount per ip exceeded |
| **504** | ANTIFRAUD_MAX_TRANSACTIONS_PER_IP_EXCEEDED | Antifraud – max transactions per ip exceeded |
| **505** | ANTIFRAUD_MAX_TRANSACTIONS_PER_CARDNO_EXCEEDED | Antifraud – max transactions per card number exceeded |
| **506** | ANTIFRAUD_MAX_AMOUNT_PER_CARDNO_EXCEEDED | Antifraud – max amount per cardnumer exceeded |
| **507** | ANTIFRAUD_IP_ADDRESS_BLOCKED | Antifraud – IP address blocked. |
| **600** | SWISH_NOT_ENROLLED | Payer alias is invalid, or payee not enrolled |
| **601** | SWISH_REFUND_ORDER_NOT_FOUND | Original Payment not found or original payment is more than 13 months old |
| **602** | SWISH_REFUND_PAYER_ERROR | Payer alias in the refund does not match the payee alias in the original payment |
| **603** | SWISH_REFUND_PAYER_ORGNR_ERROR | Payer organization number do not match original payment payee organization number |
| **604** | SWISH_REFUND_PAYEE_SSN_ERROR | The Payer SSN in the original payment is not the same as the SSN for the current Payee |

**GLOSSARY**

N = Number
D = Date in the ISO-8601 extended date format, E8601DAw ("yyyy-MM-dd"), e.g. “2011-09-17”
AN = Alphanumeric
MAC = Message Authentication Key
Secret Word = Used in calculating the MAC, unique for each Merchant
Paypage = Payment selection page, where the buyer chooses which payment method to use

## APPENDIX 2

These are example payments with minimal,required and full data . More optional data can be added as desired.

### BASIC CARD PAYMENTS WITH UNKNOWN CUSTOMER

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>SVEACARDPAY</paymentmethod>
<currency>SEK</currency>
<amount>500</amount>
<vat>100</vat>
<customerrefno>unique_id</customerrefno>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
<customer>
<country>SE</country>
<unknownCustomer>true</unknownCustomer>
</customer>
</payment>
```

### BASIC CARD PAYMENTS WITH REQUIRED CUSTOMER DATA

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>SVEACARDPAY_PF</paymentmethod>
<currency>SEK</currency>
<amount>500</amount>
<vat>100</vat>
<customerrefno>unique_id</customerrefno>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
<customer>
<firstname>jane</firstname>
<lastname>doe</lastname>
<email>JaneDoe@test.test</email>
<ssn>460509-2222</ssn>
<address>test</address>
<country>SE</country>
<city>test</city>
<zip>99999</zip>
<industrycode>12345</industrycode>
<companyname>testAB</companyname>
<unknownCustomer>false</unknownCustomer>
</customer>
</payment>
```

### BASIC CARD PAYMENTS WITH FULL CUSTOMER DATA

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>SVEACARDPAY_PF</paymentmethod>
<currency>SEK</currency>
<amount>500</amount>
<vat>100</vat>
```

```
<customerrefno>unique_id</customerrefno>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
<customer>
<firstname>jane</firstname>
<lastname>doe</lastname>
<email>JaneDoe@test.test</email>
<ssn>460509-2222</ssn>
<address>test</address>
<address2>test</address2>
<country>SE</country>
<city>test</city>
<zip>99999</zip>
<houseNumber>13</houseNumber>
<phone>test</phone>
<vatNumber>100</vatNumber>
<initials>JD</initials>
<companyname>testAB</companyname>
<industrycode>12345</industrycode>
<unknownCustomer>false</unknownCustomer>
</customer>
</payment>
```

**BASIC RECURRING CARD PAYMENTS**

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>SVEACARDPAY</paymentmethod>
<currency>SEK</currency>
<amount>0</amount>
<customerrefno>unique_id</customerrefno>
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
<subscriptiontype>RECURRING</subscriptiontype>
<customer>
<firstname>jane</firstname>
<lastname>doe</lastname>
<email>JaneDoe@test.test</email>
<address>test</address>
<address2>test</address2>
<country>SE</country>
<city>test</city>
<zip>99999</zip>
<houseNumber>13</houseNumber>
<phone>test</phone>
<vatNumber>100</vatNumber>
<initials>JD</initials>
<unknownCustomer>false</unknownCustomer>
</customer>
</payment>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment>
<paymentmethod>SVEACARDPAY</paymentmethod>
<currency>SEK</currency>
<amount>500</amount>
<customerrefno>unique_id</customerrefno>
```

```
<returnurl>https://webpaypaymentgatewaystage.svea.com/webpay-
admin/admin/merchantresponsetest.xhtml</returnurl>
<subscriptiontype>RECURRINGCAPTURE</subscriptiontype>
<customer>
<firstname>test</firstname>
<lastname>test</lastname>
<address>test</address>
<city>test</city>
<zip>99999</zip>
<country>SE</country>
<phone>test</phone>
<email>test@test.test</email>
</customer>
</payment>
```