SimplePay 1.0.x

Technical details (JAVA, ASP .NET, PHP, Python, Ruby, Node.js)

LiveUpdate, BackRef, IPN, IOS, IDN, IRN functions

2017. 03. 17.

TABLE OF CONTENT

1. Introduction

In the documentation "SimplePay_1.0.x_Payment_EN_xxxxxx.pdf" we presented the necessary development regarding SimplePay online payment methods based on the functional sample code of SDK but the technical details were not discussed there.

This documentation only discusses additional details, provides insight to the technical background of the process, but it does not include the basics and the payment process.

As a result, the information below cannot be interpreted without reference to the documentation referred to above, that is, **it is absolutely necessary to begin the development by reading that.**

In order to understand the process and the development the following chapters of the referred documentation are vital. These are necessary for the development, testing and activation irrespective of the way of implementation.

- Brief summary
- Transaction and its statuses
- Logos and guides
- Data transmission declaration
- Testing
- Support

This documentation provides help for those who carry out the development in a program language other than PHP. The basic sample code necessary for understanding can be found in this documentation in all programming languages what was defined in the subtitle of this document.

2. LiveUpdate

LiveUpdate means the start of payment transaction. During this a form is created which has to be submitted by POST method to SimplePay.

2.1. Data fields of form

During LiveUpdate implementation we have added data to the form by means of SDK setField() function.

When we did this, in view of the SimplePay, transparent and brief explanation, we did not check from field to field the type and size of the data as well as the possible limitations.

In the future we will check every necessary and optional field as it may be important what type of data are included in the given fields.

2.1.1. Data of order

The data of order are unique data globally characteristic of the transaction. They include data necessary for the use of SimplePay system.

Name of variable	Required	Size	Description
MERCHANT	Yes	16	Merchant's unique ID in the SimplePay system
ORDER_REF	Yes	256	Order's unique ID in the merchant's system
LANGUAGE	No	2	Language of payment page and letters (according to ISO 639-1), possible values: CZ, DE, EN, IT, HR, HU, PL, RO, SK. The default is HU

ORDER_DATE	Yes	19	Date of order in the format of Y-m-d H:i:s (2016-03-20 18:52:42)
PRICES_CURRENCY	Yes	3	Currency of price, possible values: HUF, EUR, USD
ORDER_SHIPPING	No	8	Amount of delivery cost which is added to the total price. In the case of HUF, without decimals (in the case of EUR and USD, two decimal digits where the decimal is the dot character). Zero, or a bigger number.
DISCOUNT	No	8	Amount of discount which is deducted from the total price. In the case of HUF, without decimals (in the case of EUR and USD, two decimal digits where the decimal is the dot character). Zero, or a bigger number which is smaller than the total price.
PAY_METHOD	No	8	Payment method. In case of credit card CCVISAMC, in case of a bank transfer WIRE. If the variable not exists the transaction will start as a credit card payment. In this case no need to add this variable to hash string.
ORDER_TIMEOUT	No	4	Time in seconds which can be spent by the customer on the payment page. It is practical to provide around 300-900 seconds. If the customer exceeds this time limit, he/she will automatically be redirected to TIMEOUT_URL.
TIMEOUT_URL	Yes	256	The customer will be redirected to this URL if he/she exceeds the value of ORDER_TIMEOUT, or if he/she clicks on "Cancel" on the payment page of SimplePay.
BACK_REF	Yes	256	The website that closes the transaction for the information of the customer. The customer will be redirected to this page after a successful or unsuccessful payment.

2.1.2. Basket data

Each transaction should at least include one product. Each product has to be provided with the same fields, that is, if the merchant or its developer would like to leave out one of the non-required fields he/she has to leave out that field in the case of all products.

Name of variable	Required	Size	Description
ORDER_PNAME	Yes	128	Name of product
ORDER_PCODE	Yes	64	Unique ID of product
ORDER_PINFO	No	128	Description of product
ORDER_PRICE	Yes	8	Price of product. In the case of HUF, without decimals (in the case of EUR and USD, two decimal digits where the decimal is the dot character).

ORDER_QTY	Yes	3	Quantity, number of product. It is a positive, whole number.
ORDER_VAT	Yes	2	VAT rate of product. It will be added to the product's price.

The ORDER_PCODE field must include the unique ID of the product. The content of this field has to be unique in your system.

If there is more than one piece in the basket of a product it is better to add it once to the basket and set the number in the quantity (ORDER_QTY) field.

If you provide gross price, then you should enter 0 value to the ORDER_VAT field. If you provide net price, then you should provide the VAT rate in the ORDER_VAT field with a whole number without percent sign, e.g. 27.

When providing the price, it is recommended to provide a gross amount without decimal digits in HUF currency (in the case of EUR and USD, two decimal digits where the decimal is the dot character) with a VAT value of 0.

If you provide fraction with more decimal digits than needed (e.g. you obtain the net amount and the rate of VAT by dividing from the gross amount) then the system will round, because of which the amount in the online store may differ from that of the payment page.

2.1.3. Billing information

Name of variable	Required	Size	Description
BILL_FNAME	No	64	First name
BILL_LNAME	No	64	Last name
BILL_EMAIL	Yes	64	Email address
BILL_PHONE	No	32	Phone number
BILL_COMPANY	No	128	Company name
BILL_FISCALCODE	No	64	VAT number
BILL_COUNTRYCODE	No	2	Country
BILL_STATE	No	64	County
BILL_CITY	No	128	City
BILL_ADDRESS	No	128	Address
BILL_ADDRESS2	No	128	Address
BILL_ZIPCODE	No	32	Zip code

2.1.4. Delivery information

Name of variable	Required	Size	Description
DELIVERY_FNAME	No	64	First name
DELIVERY_LNAME	No	64	Last name
DELIVERY_EMAIL	No	64	Email address
DELIVERY_PHONE	No	32	Phone number
DELIVERY_COUNTRYCODE	No	2	Country
DELIVERY_STATE	No	64	County
DELIVERY_CITY	No	128	City
DELIVERY_ADDRESS	No	128	Address
DELIVERY_ADDRESS2	No	128	Address
DELIVERY_ZIPCODE	No	32	Zip code

2.2. Character encoding

You have to use a character encoding in UTF-8 in each case. Save the files with the 'without UTF-8 BOM' encoding.

If the character encoding is not appropriate, the hash value generated from transaction details will be corrupted because of which the SimplePay system cannot interpret the transaction and discards it.

Send all texts including accented characters in some textual field for testing the character encoding, e.g. „ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP". If the transaction starts appropriately with this and you see the above text in a suitable format, the character encoding is appropriate.

2.3. LiveUpdate implementation

In the majority of cases it is sufficient to execute an implementation by the sample code of SDK but in some cases it may be needed to create an own unique source code.

This chapter is directed at those developers who intend to solve the SimplePay LiveUpdate integration in this way.

The chapter is basically identical to those described in the LiveUpdate implementation in SDK documentation but expands it with source code details.

However, it is more in that it discusses those details which are executed by the sample files (e.g. hash calculation), thus it has not been explained earlier but if someone wants to encode it, it is necessary to understand it.

Within the entire process you initiate the credit card payment with the payment transaction details under LiveUpdate.

At this point the followings have to be performed:

- create a form and upload it with the transaction's basket data

- send the form by POST method to the URL of SimplePay
https://secure.simplepay.hu/order/lu.php

2.3.1. HTML form

A sample of the HTML form to be created can be seen below.

The form must be sent to the following URL by POST method:

test transaction

https://sandbox.simplepay.hu/payment/order/lu.php

live transaction

https://secure.simplepay.hu/payment/order/lu.php

MERCHANT and **SECRET_KEY** data that are needed to start a transaction can be found on the SimplePay's merchant control panel (https://sandbox.simplepay.hu/admin/).

The header section of "Account/Technical" page contains the general data of the account. You can find here the Merchant ID (MERCHANT).
The "Base data" tab shows the key for hash computing (SECRET_KEY).
These properties of the accounts are unique per currencies.

You can choose between the unique currency accounts in the menu on the right (Select another account).

The value of MERCHANT is added to the generated form; this data is not secret. However, SECRET_KEY is secret, this is the key to authenticate the call both on the merchant's and the SimplePay's page. This is unique in the case of each merchant. The authenticating signature is calculated on the basis of this and it is known only by SimplePay and the merchant. This data is necessary to generate HMAC signature which will be presented in detail later. As both parties (the merchant and the SimplePay) know it, and the authentication is carried out on the basis of this, thus

DO NOT send the value of SECRET_KEY during LiveUpdate.

The characteristics of the form's data fields were presented in detail earlier. The fields basically are composed of a name and a value such as the currency:

```
<input type='hidden' name='PRICES_CURRENCY' id='PRICES_CURRENCY' value='HUF' />
```

```
<html>
<head><meta charset="UTF-8"></head>
<form action='https://sandbox.simplepay.hu/payment/order/lu.php' method='POST' id='SimplePayForm' accept-ch
arset='UTF-8'>
<input type='hidden' name='MERCHANT' id='MERCHANT' value='PUBLICTESTHUF' />
<input type='hidden' name='ORDER_REF' id='ORDER_REF' value='101010514600323871720' />
<input type='hidden' name='ORDER_DATE' id='ORDER_DATE' value='2016-04-07 12:33:07' />
<input type='hidden' name='ORDER_PNAME[]' id='ORDER_PNAME' value='Lorem 1' />
<input type='hidden' name='ORDER_PCODE[]' id='ORDER_PCODE' value='sku0001' />
<input type='hidden' name='ORDER_PINFO[]' id='ORDER_PINFO' value='ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP' />
<input type='hidden' name='ORDER_PRICE[]' id='ORDER_PRICE' value='331' />
<input type='hidden' name='ORDER_QTY[]' id='ORDER_QTY' value='1' />
<input type='hidden' name='ORDER_VAT[]' id='ORDER_VAT' value='0' />
<input type='hidden' name='PRICES_CURRENCY' id='PRICES_CURRENCY' value='HUF' />
<input type='hidden' name='ORDER_SHIPPING' id='ORDER_SHIPPING' value='0' />
<input type='hidden' name='DISCOUNT' id='DISCOUNT' value='0' />
<input type='hidden' name='PAY_METHOD' id='PAY_METHOD' value='CCVISAMC' />
<input type='hidden' name='LANGUAGE' id='LANGUAGE' value='HU' />
<input type='hidden' name='ORDER_TIMEOUT' id='ORDER_TIMEOUT' value='300' />
<input type='hidden' name='TIMEOUT_URL' id='TIMEOUT_URL' value='https://weboldalam.tld/timeout.php?order_re
f=101010514600323871720&order_currency=HUF' />
<input type='hidden' name='BACK_REF' id='BACK_REF' value='https://weboldalam.tld/backref.php?order_ref=1010
10514600323871720&order_currency=HUF' />
<input type='hidden' name='BILL_FNAME' id='BILL_FNAME' value='Tester' />
<input type='hidden' name='BILL_LNAME' id='BILL_LNAME' value='SimplePay' />
<input type='hidden' name='BILL_EMAIL' id='BILL_EMAIL' value='email@example.com' />
<input type='hidden' name='BILL_PHONE' id='BILL_PHONE' value='36201234567' />
<input type='hidden' name='BILL_ADDRESS' id='BILL_ADDRESS' value='First line address' />
<input type='hidden' name='BILL_ZIPCODE' id='BILL_ZIPCODE' value='1234' />
<input type='hidden' name='BILL_CITY' id='BILL_CITY' value='City' />
<input type='hidden' name='BILL_STATE' id='BILL_STATE' value='State' />
<input type='hidden' name='BILL_COUNTRYCODE' id='BILL_COUNTRYCODE' value='HU' />
<input type='hidden' name='DELIVERY_FNAME' id='DELIVERY_FNAME' value='Tester' />
<input type='hidden' name='DELIVERY_LNAME' id='DELIVERY_LNAME' value='SimplePay' />
<input type='hidden' name='DELIVERY_PHONE' id='DELIVERY_PHONE' value='36201234567' />
<input type='hidden' name='DELIVERY_ADDRESS' id='DELIVERY_ADDRESS' value='First line address' />
<input type='hidden' name='DELIVERY_ZIPCODE' id='DELIVERY_ZIPCODE' value='1234' />
<input type='hidden' name='DELIVERY_CITY' id='DELIVERY_CITY' value='City' />
<input type='hidden' name='DELIVERY_STATE' id='DELIVERY_STATE' value='State' />
<input type='hidden' name='DELIVERY_COUNTRYCODE' id='DELIVERY_COUNTRYCODE' value='HU' />
<input type='hidden' name='ORDER_HASH' id='ORDER_HASH' value='d1ecf3e3818faadd56b2eef28962ad7d' />
<button type='submit'>Start SimplePay online payment</button>
</form>
</html>
```

In the case of products, data must be provided differently. As the basket may include more products and the same parameters must be provided, we use arrays to upload the content of the basket. Such arrays are the following:

ORDER_PNAME
ORDER_PCODE
ORDER_PINFO
ORDER_PRICE
ORDER_QTY
ORDER_VAT

If e.g. ORDER_PNAME does not have a value in the case of any product, you should send it with empty value.
The product's data are found in the same location of the arrays, that is, the first element of the description array or the price array logically belongs to the first element of the name array, etc.

In the HTML source code the array is indicated by "[]" characters after the name of the variable. The array elements of the form are indexed with numbers and the field's name has to be followed by square brackets. Logically it is not different from a PHP array.

Example for the definition of HTML array:

```
<input type="hidden" value="Product_1" id="ORDER_PNAME"
name="ORDER_PNAME[]">
<input type="hidden" value="Product_2" id="ORDER_PNAME"
name="ORDER_PNAME[]">
<input type="hidden" value="100" id="ORDER_PRICE" name="ORDER_PRICE[]">
<input type="hidden" value="200" id="ORDER_PRICE" name="ORDER_PRICE[]">
```

Example for the definition of PHP array:

```
$array['ORDER_PNAME'][] = 'Product_1';
$array['ORDER_PNAME'][] = 'Product_2';
$array['ORDER_PRICE'][] = '100';
$array['ORDER_PRICE'][] = '200';
```

Whether we send and receive the first form by POST method, then display it by means of print_r() function, or if we use the array created in PHP code in the same way; we will get the same result:

```
Array
(
          [ORDER_PNAME] => Array
          (
                    [0] => Product_1
                    [1] => Product_2
          )
          [ORDER_PRICE] => Array
          (
                    [0] => 100
                    [1] => 200
          )
)
```

In case of HTML array, it is not appropriate to provide index or use associative array.

Correct use:
<input type="hidden" name="ORDER_PNAME[]" value=" Product_1" />
<input type="hidden" name="ORDER_PNAME[]" value="Utazó táska" />

Incorrect use:
<input type="hidden" name="ORDER_PNAME**[0]**" value=" Product_1" />
<input type="hidden" name="ORDER_PNAME**[1]**" value="Utazó táska" />

2.3.2. Required fields necessary to start transactions

Fields in the following form are the minimum necessary ones to be able to start a transaction.

We have presented in detail the fields and their content in the possible data of LiveUpdate form.

In the following chapter, we will describe the calculation of ORDER_HASH field necessary for authentication.

```
<form action='https://sandbox.simplepay.hu/payment/order/lu.php' method='POST' id='SimplePayFo
rm' accept-charset='UTF-8'>
<input type='hidden' name='MERCHANT' id='MERCHANT' value='PUBLICTESTHUF' />
<input type='hidden' name='ORDER_REF' id='ORDER_REF' value='101010514872470318621' />
<input type='hidden' name='ORDER_DATE' id='ORDER_DATE' value='2017-02-16 12:10:31' />
<input type='hidden' name='ORDER_PNAME[]' id='ORDER_PNAME' value='Lorem 1' />
<input type='hidden' name='ORDER_PCODE[]' id='ORDER_PCODE' value='sku0001' />
<input type='hidden' name='ORDER_PRICE[]' id='ORDER_PRICE' value='1108' />
<input type='hidden' name='ORDER_QTY[]' id='ORDER_QTY' value='1' />
<input type='hidden' name='ORDER_VAT[]' id='ORDER_VAT' value='0' />
<input type='hidden' name='PRICES_CURRENCY' id='PRICES_CURRENCY' value='HUF' />
<input type='hidden' name='TIMEOUT_URL' id='TIMEOUT_URL' value='http://mywebsite.tld/timeout.p
hp?order_ref=101010514872470318621&order_currency=HUF' />
<input type='hidden' name='BACK_REF' id='BACK_REF' value='http://mywebsite.tld/backref.php?ord
er_ref=101010514872470318621&order_currency=HUF' />
<input type='hidden' name='BILL_EMAIL' id='BILL_EMAIL' value='sdk_test@otpmobil.com' />
<input type='hidden' name='ORDER_HASH' id='ORDER_HASH' value='3cd244760ab9a23205dea4ef08c7840e
' />
<button type='submit'>Start SimplePay</button>
</form>
```

2.3.3. Hash calculation, authentication

For the authentication of the provided data, a HMAC_MD5 HASH that is unique per transaction is necessary which is generated with the secret key belonging to your account.

This generated HASH has to be provided as the last parameter in the LiveUpdate in the following way:

<input type="hidden" name="ORDER_HASH"
value="3cd244760ab9a23205dea4ef08c7840e" />

The content of value is logically unique in each transaction .

The source string belonging to the hash can be calculated that we add the length of the field's content before the value of each field in bytes. So if the value of a variable is "HUF" and it has three characters, then the source string will be "3HUF".

As we do not need the number of characters but its length in bytes, then it may provide different values, for example in the case of accented characters.

13PUBLICTESTHUF
192016-04-08 11:00:01
7Kártya
31ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP

If you do not use a character encoding in UTF-8, you have to encode the data into UTF-8 format before the calculation of characters because the length of text may vary. In this case it is very important that you have to send the UTF-8 encoded content in the form fields because if you only applied it during the hash calculation but you has not sent it, then the SimplePay system cannot receive the transaction because of the difference.

The fields in the following table are added to the HASH calculation. The fields are required and their order is determined according to the first column.

The order of fields is determined according to the first column. If a not mandatory field (ORDER_PINFO, ORDER_SHIPPING, DISCOUNT, PAY_METHOD) not included to the form, no need to put it into the hash string.

#	Name of field	Mandatory	Value of field	Length	Source string value
1	MERCHANT	Yes	PUBLICTESTHUF	13	13PUBLICTESTHUF
2	ORDER_REF	Yes	1010105146011598 78253	21	211010105146011598782 53
3	ORDER_DATE	Yes	2016-04-08 11:46:27	19	192016-04-08 11:46:27
4	ORDER_PNAME	Yes	Lorem ipsum	11	11Lorem ipsum
5	ORDER_PCODE	Yes	sku0001	7	7sku0001
6	ORDER_PINFO	No	ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP	31	31ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP
7	ORDER_PRICE	Yes	331	3	3331
8	ORDER_QTY	Yes	1	1	11
9	ORDER_VAT	Yes	0	1	10
10	ORDER_SHIPPING	No	0	1	10
11	PRICES_CURRENCY	Yes	HUF	3	3HUF
12	DISCOUNT	No	0	1	10
13	PAY_METHOD	No	CCVISAMC	8	8CCVISAMC

The following source string can be linked from the following data:

13PUBLICTESTHUF211010105146011598782531920 16-04-08 11:46:2711Lorem ipsum7sku000131ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP3331111010 3HUF108CCVISAMC

If a required variable does not have a value, e.g. ORDER_PNAME, then the number of bytes will be 0. In this case 192016-04-08 11:46:2707sku0001 will be on this part of the string.

The merchant account used for the test has the following IDs:

MERCHANT: PUBLICTESTHUF
SECRET_KEY: FxDa5w314kLlNseq2sKuVwaqZshZT5d6

The hash value is generated with an encryption key (SECRET_KEY).

We demonstrate this function below. The sample codes are available on all programming languages defined in the subtitle of this document.

NOTE: examples always include the minimum sample code necessary for the function's implementation.

In the case of all examples, the variable **sourceString** includes the earlier linked data and the variable **secretKey** includes the SECRET_KEY value.

Value of sourceString

13PUBLICTESTHUF21101010514601159878253192016-04-08 11:46:2711Lorem ipsum7sku000131ÁRVÍZTŰRŐ TÜKÖRFÚRÓGÉP33311110103HUF108CCVISAMC

Value of secretKey

FxDa5w314kLlNseq2sKuVwaqZshZT5d6

JAVA SOLUTION

```
javax.crypto.spec.SecretKeySpec;
javax.crypto.Mac

private static String calculateHmacMd5AsBytes(String secretKey, String sourceString) {
              Mac mac = Mac.getInstance("HmacMD5");
              mac.init(new SecretKeySpec((secretKey).getBytes("UTF-8"), "HmacMD5"));
              return ByteHelper.fromByteArray(mac.doFinal(sourceString.getBytes("UTF-8")));
}
```

ASP .NET SOLUTION

```
var secretKey = System.Text.Encoding.UTF8.GetBytes("secretKey");
var sourceString = System.Text.Encoding.UTF8.GetBytes("sourceString");
var hmacMD5 = new System.Security.Cryptography.HMACMD5(secretKey);
var hash = hmacMD5.ComputeHash(sourceString);
```

PHP SOLUTION

```
$hash = hash_hmac('md5', $sourceString, $secretKey);
```

PYTHON SOLUTION

```
# -*- coding:Utf-8 -*-
import hmac
digest = hmac.new(secretKey)
digest.update(sourceString)
hash = digest.hexdigest()
```

RUBY SOLUTION

```
require 'OpenSSL'
digest = OpenSSL::Digest.new('md5')
hash = OpenSSL::HMAC.hexdigest(digest, secretKey, sourceString)
```

NODE.JS SOLUTION

```
var crypto = require('crypto');
var hmac = crypto.createHmac('md5', secretKey)
var hash = hmac.update(sourceString, 'utf8').digest('hex')
```

The value of the generated hash with the above data and key will be the following in any programme language: **51f48bfda333a8c477bbbedd18a1f787**

More products in the basket
If there are more products in the basket, then the data of all products must be calculated in the hash string. In this case the product data are calculated per type, that is, first each ORDER_PNAME, then each ORDER_PCODE, etc.

In case of two products, this is calculated as follows

'ORDER_PNAME' = 'Lorem ipsum',
'ORDER_PCODE' = 'sku0001',
'ORDER_PINFO' = 'ÁRVÍZTŰRŐ',
'ORDER_PRICE' = 123,
'ORDER_QTY' = 0,
'ORDER_VAT' = 1

'ORDER_PNAME' = 'Dolor sit amet',
'ORDER_PCODE' = 'sku0002',
'ORDER_PINFO' = 'TÜKÖRFÚRÓGÉP',
'ORDER_PRICE' = 456,
'ORDER_QTY' = 0,
'ORDER_VAT' = 1

Thus, all data of the transaction must be in the following order before arranging the source string.

[0] => PUBLICTESTHUF
[1] => 101010514601278769072
[2] => 2016-04-08 15:04:36
[3] => Lorem ipsum
[4] => Dolor sit amet
[5] => sku0001
[6] => sku0002
[7] => ÁRVÍZTŰRŐ
[8] => TÜKÖRFÚRÓGÉP
[9] => 123
[10] => 456
[11] => 1
[12] => 1
[13] => 0
[14] => 0
[15] => 0
[16] => HUF
[17] => 0
[18] => CCVISAMC

The hash string from the above data will be the following:

13PUBLICTESTHUF211010105146012787690721920 16-04-08 15:04:3611Lorem ipsum14Dolor sit amet7sku00017sku000213ÁRVÍZTŰRŐ17TÜKÖRFÚRÓGÉP31233456111110101 03HUF108CCVISAMC

Value of the generated hash: **6ed529adde57070bf64ce05efa559307**

2.3.4. Hash check

For checking it is recommended to use the free service on the following page at http://hash.online-convert.com/md5-generator

Copy the source string into the field "**Text you want to convert to a MD5 hash**", then enter the key used for encoding (SECRET_KEY) in the field "**Shared secret key used for the HMAC variant**".

As long as the hash value generated in this page is not the same as the hash value used by you in the form, you will be expected to receive "Invalid Signature" error message from the SimplePay system.

2.4. Debugging

During LiveUpdate if you provide data with such an error that the SimplePay system cannot interpret the transaction, you will immediately be redirected to the website from where the transaction arrived. In this case the payment is not successful.

The error message in the URL will be available in the "**err**" variable.

In the case of tests to be executed during the development, it is better to take into consideration the following. If an error occurs, you must debug based on the following list.

This list does not include the source code's many error possibilities related to syntax, server environment, network connection but it focuses on the possible problem sources in relation to the development of SimplePay online payment.

Since these are construction defects it is recommended to check them at least once even if the code runs well.

- Log checks
- Appropriate character encoding (UTF-8)
- Merchant code (MERCHANT), there is no space before or after (it is recommended to use a trim() function)
- Encrypting key (SECRET_KEY), there is no space before or after (it is recommended to use a trim() function) DO NOT transmit this data!
- Providing the currency suitable for the merchant code (MERCHANT)
- "Invalid Account" error message, inappropriate or lacking merchant account data (MERCHANT).
- "Invalid signature" error message, Hash encoding error in the online store. It can arise from the lack of SECRET_KEY, truncated use, or if there is a space in the beginning or the end. It may refer to a character encoding problem (not UTF-8), or the lack of required fields.
- Corrupted URLs in the BACK_REF and TIMEOUT_URL fields. It points to non-existent or non-available file, or there is a space in the URL. Interchanging HTTP and HTTPS.
- Lacking or not uploaded required data fields
- The order ID is not unique in the merchant's system

3. BackRef

The BackRef is an informative page. This is a location in the merchant's system where the customer is redirected from the payment page in case of a successful or unsuccessful payment. In case of a successful payment, this page is indicative only and it is not the end of the transaction. In case of an unsuccessful payment, this is the end of the transaction.

The BackRef URL had to be provided in the BACK_REF variable upon starting the transaction. The SimplePay system links the provided values to the data regarding the transaction's result.

Data linked to BACK_REF URL may be the following

Name of field	Content of field
RC (return code)	A return code indicating the transaction's result. If it begins with 000 or 001, the transaction is successful, in case of other numbers it is unsuccessful, and the appropriate message must be displayed on the page.
RT (return text)	The textual version of the return code indicating the transaction's result.
3dsecure	It is currently provided only for compatibility purposes. Its value is NO in each case.
date	The exact date of bank authorization
payrefno	SimplePay transaction ID
ctrl	A value authenticating BackRef call. On the basis of this it may be checked if the call actually comes from the SimplePay system.

Within the entire process we receive the transaction returning from the SimplePay payment page to the online store's website under the Back Ref and notify the customers about the result.

At this point the developer's task is to:

- process and authenticate the data received by GET method
- notify the customer about the transaction's result
- store data according to the individual logic of the online store

3.1. Authentication

The URL is authenticated by the previously introduced HASH calculation method. The only difference is that the content's length is not calculated per variable but the entire URL is provided as one value, namely, one string.

Sample BackRef string

https://weboldalam.tld/backref.php?order_ref=10101051461157026966 4&order_currency=HUF&RC=000&RT=000+%7C+OK&3dsecure=NO&date=2016-04-20+14%3A57%3A38&payrefno=99016530&ctrl=a5a268fd200eaef93e87a3f1403ce65f

If it is treated as one string, then the last 38 characters (&ctrl=**a5a268fd200eaef93e87a3f1403ce65f**) must be cut. By cutting the ctrl variable from the entire string for the authentication, the following value remains in the case of the above sample. This will be the base of the hash string.

https://weboldalam.tld/backref.php?order_ref=10101051461157026966 4&order_currency=HUF&RC=000&RT=000+%7C+OK&3dsecure=NO&date=2016-04-20+14%3A57%3A38&payrefno=99016530

The length of this string must be calculated in a way detailed in the description of LiveUpdate. In this case the length will be 165 which we link to the beginning of the string and thus we create a source string.

165https://weboldalam.tld/backref.php?order_ref=10101051461157026966 4&order_currency=HUF&RC=000&RT=000+%7C+OK&3dsecure=NO&date=2016-04-20+14%3A57%3A38&payrefno=99016530

It is only possible to authenticate BackRef with merchant data used to start the transaction.

The merchant account has the following IDs:
MERCHANT: PUBLICTESTHUF
SECRET_KEY: FxDa5w314kLlNseq2sKuVwaqZshZT5d6

From this point the hash generation is the same as it has been described in the case of LiveUpdate.

Value of **sourceString**

165https://weboldalam.tld/backref.php?order_ref=10101051461157026966 4&order_currency=HUF&RC=000&RT=000+%7C+OK&3dsecure=NO&date=2016-04-20+14%3A57%3A38&payrefno=99016530

Value of **secretKey**
FxDa5w314kLlNseq2sKuVwaqZshZT5d6

Thus, the value of the generated hash will be **a5a268fd200eaef93e87a3f1403ce65f**, which is the same as the value provided in ctrl variable in the BackRef call.

If the generated and received values are the same, you can be sure in the authenticity of the received values.

3.1.1. Risks of errors during authentication

The SimplePay system calculates the hash value sent in the ctrl variable regarding the entire URL encoded URL. Consequently, the authentication must be carried out on the merchant's page before each reconstruction, otherwise the calculated value will not be the same as the one received in the ctrl variable.

3.2. Information

BackRef page serves to provide information to the customers. If the authentication was successful, the received data can be interpreted.

The information must include the transaction's result (successful, unsuccessful), the SimplePay transaction ID (payrefno variable) and the merchant transaction ID.

The content of RC variable is 000 or 001 in case of successful transaction. In all other cases the transaction is unsuccessful.

3.2.1. Information in case of unsuccessful credit card transaction

In case of an unsuccessful payment, this is the end of the transaction.

Reason for failure may be among others that the credit card does not exist or has expired. It is possible that there is not enough money on the card, or there is money but the customer has reached the daily limit set by them.

The actual reason for the failure may be provided by the bank which issued the card for the customer.

After the unsuccessful payment, the following compulsory pieces of information must be displayed
1. The result, that is, "Unsuccessful payment"
2. The following information: Please check the validity of data provided during the transaction. If all data are correct, please contact your bank which issued the card to request the reason for the refusal.
3. SimplePay reference number which is the SimplePay's transaction ID. The SimplePay may provide a response on the basis of this regarding any questions relating to the transaction
4. Merchant's order ID
5. Date of transaction

Sample information

Unsuccessful transaction.
Please check the validity of data provided during the transaction.
If all data are correct, please contact your bank which issued the card to request the reason for the refusal.
SimplePay reference number: **99013817**
Order ID: **101010514582296094262**
Date: **2016-03-17 16:46:52**

3.2.2. Information in case of a successful transfer

In case of a successful card check, the BackRef page **is not the end of the transaction but it is only informative** for the customer.

The order may be fulfilled only after the reception of the IPN message described in chapter 5.

After the successful payment, the following **compulsory pieces of information** must be displayed
1. The result, namely, "Successful payment", or "Successful transaction", or "Successful card check", or optionally "Waiting for confirmation"
2. SimplePay reference number which is the SimplePay's transaction ID. The SimplePay may provide a response on the basis of this regarding any questions relating to the transaction
3. Merchant's order ID
4. Date of transaction

Sample information

Successful card check.
SimplePay reference number: **99013247**
Order ID: **101010514577073006166**
Date: **2016-03-11 15:41:43**

Fraud monitoring runs in the background at this point. Only if it is successful does the system send out the IPN message. The transaction ends with the IPN message. At this time, the merchant can fulfil the order.

It is very important to know that the IPN message may arrive in the background at the same time (or even earlier) as the customer is redirected to the BackRef page.

NOTE: You always have to fulfil the order based on the IPN.

This also means that if separate statuses belong to these event in your system, then the IPN must in each case must have higher status than BackRef, namely, the **status of BackRef must never overwrite the status of IPN.**

3.2.3. Information in case of a successful transfer

The process in case of a transfer differs only in payment method up to the point when the payment is started. In this case the payment page is only informative with data necessary for the transfer.

On the information page the customer does not have to do anything (the transfer can be carried out from his/her net bank), thus it is not sure that he/she will be redirected to the merchant's website.

If he/she clicks on "Back" button and returns to the merchant's website, then he/she must be notified. The notification in this case differs from the credit card payment.

The BackRef page is not the end of the transaction but it is only informative for the customer.

The order may be fulfilled only after the reception of the IPN message described in chapter 5.

After the wire transfer, the following **compulsory pieces of information** must be displayed
1. At this point only an order has been placed, there cannot be information about payment, that is, "Successful order".
2. It has to be indicated that the order will be fulfilled on a later date, that is, "The order will be fulfilled after the arrival of the transfer"
3. SimplePay reference number which is the SimplePay's transaction ID. The SimplePay may provide a response on the basis of this regarding any questions relating to the transaction
4. Merchant's order ID
5. Date of transaction

Sample information

Successful order.
The order will be fulfilled after the arrival of the transfer
SimplePay reference number: **99014232**
Order ID: **101010514599378142316**
Date: **2016-03-29 12:16:55**

4. Timeout

Cancelled transaction means that after starting LiveUpdate the customer arrives at the SimplePay payment page but due to some reasons he/she wants to return to the merchant's website. In this case he/she may click on "Cancel" button with which he/she may initiate this return.

In case of a **timeout** after starting LiveUpdate the customer arrives at the SimplePay payment page but until the time provided in the ORDER_TIMEOUT variable expires, he/she cannot initiate the payment. In this case at the exceeding of the limit the payment page automatically redirects the customer to the merchant's website.

In both cases the customer will return to the URL provided in the TIMEOUT_URL variable during the LiveUpdate.

It is very important to take into consideration during providing information that **payment has not been made in either cases** as the payment has been cancelled before providing the credit card details and making the payment. Therefore, here it is not possible to notify the customer about unsuccessful payment.

After a cancelled or timed-out transaction, the following **compulsory pieces of information** must be displayed
1. The result, that is, "Cancelled transaction", or "Timed-out transaction"
2. The following information: "You cancelled the payment, or the maximum time of transaction has expired. "
3. Optionally: merchant's order ID and date can be displayed

Sample information

Aborted transaction
You cancelled the payment, or the maximum time of transaction has expired.
Order ID: **10101051458143998**
Date: **2016-03-16 16:00:07**

5. IPN

The IPN (Instant Payment Notification) communication is the final part of the successful payment process. The IPN call is sent from the SimplePay system to the URL provided by the merchant with POST method.

The URL can be provided under the panel "Basic data" of the menu "Account/Technical" in the merchant's control panel (https://sandbox.simplepay.hu/admin/).

At this point the credit card transaction has gone through the bank's card check and the fraud monitoring. It is very important that the IPN be sent out only in case of successful payments. If the authorization is unsuccessful, it already returned to BackRef as an unsuccessful payment and the transaction ended there.

If the card authorization was successful but blocked during fraud monitoring, the SimplePay's customer service gets in touch with the merchant and it does not authorize the transaction (does not send IPN message) until there is a potential risk of abuse.

In case of a wire transfer, the IPN is sent if the money has arrived from the customer to the account of OTP Mobile Ltd. and based on the transfer's notice it can be assigned to the transaction made on the informative page.

The transaction status at the time when the first IPN is sent: **COMPLETE**

It may be different if it applies a two-step payment and it requests IPN message about the card authorization. In this case the status is the same as the value of the PAYMENT_AUTHORIZED on the BackRef page.

It follows from the above that the IPN always indicates successful payments

IPN can be sent out also manually, especially for testing and debugging purposes. This is possible only in case of a successful transaction.

If you click on the order reference number in the transaction list on the merchant's control panel, you will open the page showing transaction details. Here, by clicking on the blue "Resend IPN" button you can send it.

The result of sending can be seen on the same page under "Communication" tab among "IPN messages". As the request will be included in a task list when the request is started it is not sure that the result of sending will promptly be displayed in the list.

By opening the IPN message you may check if the website which processes the IPN message in the given URL is available in your system. If the value of the response's HTTP code is not 200, it is not available for some reason.

Here you can check data sent to your website and the response your website sent to the IPN message.

At this point the developer's task is to:

- authenticate the call
- provide feedback to the SimplePay about the successful receipt

5.1. Authentication

The value of all provided fields must be processed, except for the variable called HASH. Pay special attention to the fact that the IPN message may include arrays, e.g. product details. In this case the arrays have to be read and their content have to be added to the string.

For example, in case of the following array the hash string may be composed as follows

```
Array
(
    [CURRENCY] => HUF
    [IPN_PNAME] => Array
        (
            [0] => Product_1
            [1] => Product_2
        )

    [IPN_PCODE] => Array
        (
            [0] => SKU0001
            [1] => SKU0002
        )

    [IPN_TOTALGENERAL] => 8319
)
```

Part of a hash string - sample
3HUF9Product_19Product_27SKU00017SKU00024831 9

The logic of hash string composition and the calculation of hash are the same as the method described during the LiveUpdate.

If the value received in the HASH variable is the same as the calculated value, the authentication is successful.

5.2. Confirmation of the IPN message.

If the authentication was successful, a confirmation must be provided to the SimplePay system.

On the basis of this confirmation, the SimplePay system obtains information that the merchant has received and processed the IPN message about the success of transaction. The confirmation is vital as if it does not take place or it is not appropriate it attempts to send the IPN again.

After the first IPN, the logic of resend after non-received IPN is the following:

- after 5, 10, 15, 30, 45 minutes
- after 1, 2, 3, 6, 9, 12, 18 hours
- after 1, 1.5, 2, 2.5, 3 days

After these, the server no longer sends out IPN message regarding the given transaction. Of course, IPN messages may be sent out manually from the merchant's admin interface.

During the confirmation and after receiving and validating the IPN message, a response in the following format must be written which include the current date and the hash value calculated based on a part of data received in IPN:

<EPAYMENT>2016040813427|8a23a1f99f95eaafddf4d35a9671ef3b</EPAYMENT>

The following pieces of information are needed for the confirmation

#	Variable	Source	Content
1	IPN_PID [0]	It is included in the IPN message.	ID of the first product
2	IPN_PNAME[0]	It is included in the IPN message.	Name of the first product
3	IPN_DATE	It is included in the IPN message.	Date in the IPN in YmdHis format, e.g. **20160408113426**
4	DATE	Current time of the merchant's server	Timestamp generated at the confirmation in a YmdHis format, e.g. **20160408113427**

In order to calculate the hash of the confirmation the four values in the above table must be used in the above order as follows:

IPN_PID[0] = 42
IPN_PNAME[0] = Product_1
IPN_DATE = 2016040813426
DATE = 2016040813427

The IPN_DATE and the generated date (DATE) may often be the same but it is not always true. If the confirmation does not take place in the same second or the clocks of the servers are not synchronized, this value may be different.

From the above data hash string will be the following:

2429Product_11420160408134261420160408134 27

The merchant account has the following IDs:
MERCHANT: PUBLICTESTHUF
SECRET_KEY: FxDa5w314kLINseq2sKuVwaqZshZT5d6

The hash is generated as per those described in the LiveUpdate which are the following based on the above data: 8a23a1f99f95eaafddf4d35a9671ef3b

The confirmation has to be displayed in the following form. The date is a generated date (DATE) which counts into the hash string, while hash is a value generated for the confirmation. The two data are separated with a | symbol.

NOTE: There is no space in the confirmation

<EPAYMENT>20160408134 27|8a23a1f99f95eaafddf4d35a9671ef3b</EPAYMENT>

The transaction's status will be COMPLETE at the arrival of the IPN message.

5.3. Check of the IPN's success

You can check the success of IPN confirmation on the merchant's admin interface. If you open the specific payment in the transaction list, you will find it in "Communication" menu among "IPN messages".

If the status of the submitted message is not OK, you have to open and check the confirmation your website sent out in detail.
The most frequent risks of error:

- the IPN URL is not available
- the http status code is not 200
- EPAYMENT part is not included in the message

6. Modification and query of transactions

A common element in the functions (IOS, IDN, IRN) of the following chapters that it modifies an existing transaction, or queries its data.

Everything takes place in the background outside the browser between the merchant's and the SimplePay's server. The provided data must be sent to the SimplePay's server with a POST call in the background which executes and immediately replies to it.

During this, the necessary data must be sent to the URL belonging to the given communication and the received output must be processed.

In case of all implementations, two pieces of data are needed; the URL where it sends the call and the request to be sent in pairs of name-value.

The following example only shows the communication; its content is not important currently.

The value of URL in the example:
https://sandbox.simplepay.hu/payment/order/ios.php

The name-value pairs of the data to be sent:

MERCHANT = PUBLICTESTHUF
REFNOEXT = 101010514615913074586
HASH = 9607a566c832821b8447eea204e6da1e

The response to the call with the above data has an XML format with the following content

```
<order>
        <order_date>2016-04-25 13:35:07</order_date>
        <refno>99016764</refno>
        <refnoext>101010514615913074586</refnoext>
        <order_status>COMPLETE</order_status>
        <paymethod>Visa/MasterCard/Eurocard</paymethod>
        <hash>76621655113b7fd00d605075d28023f1</hash>
</order>
```

JAVA SOLUTION

```
HttpPost httpPost = new
HttpPost("https://sandbox.simplepay.hu/payment/order/ios.php");

List<BasicNameValuePair> formparams = new ArrayList<BasicNameValuePair>();
formparams.add(new BasicNameValuePair("MERCHANT", "PUBLICTESTHUF"));
formparams.add(new BasicNameValuePair("REFNOEXT", "101010514615913074586"));
formparams.add(new BasicNameValuePair("HASH", "9607a566c832821b8447eea204e
6da1e"));

httpPost.setEntity(new UrlEncodedFormEntity(formparams));
HttpResponse response = httpClient.execute(httpPost);
BufferedReader rd = new BufferedReader(new InputStreamReader(response.getEntit
y().getContent()));

StringBuffer result = new StringBuffer();
String line = "";
while ((line = rd.readLine()) != null) {
  result.append(line);
}
Pack resultXml = new XmlPack(result.toString()).setInt("responseCode", response.getSt
atusLine().getStatusCode());
return resultXml;
```

ASP .NET SOLUTION

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
namespace ConsoleApplication
{
   class Program
   {
      static void Main(string[] args)
      {
         var postData = new Dictionary<string, string>();
         postData.Add("MERCHANT", "PUBLICTESTHUF");
         postData.Add("REFNOEXT", "101010514615913074586");
         postData.Add("HASH", "9607a566c832821b8447eea204e6da1e");

         var pspClient = new SimplePSPClient();
         Task.Run(async () =>
         {
            var xml = await pspClient.GetString(new Uri("https://sandbox.simplepay.hu/payment/order/ios.php"), postData);
         });
                                    Console.ReadLine();
      }
   }
   public class SimplePSPClient
   {
      private HttpClient client;
      public SimplePSPClient()
      {
         client = new HttpClient();
      }
      public async Task<string> GetString(Uri uri, Dictionary<string, string> postData)
      {
         var content = new FormUrlEncodedContent(postData.ToList());
         var response = await client.PostAsync(uri, content);

         if (response.IsSuccessStatusCode)
         {
            var data = await response.Content.ReadAsStringAsync();

            return data;
         }
         else
            return null;
      }
   }
}
```

PHP SOLUTION

```
$url = 'https://sandbox.simplepay.hu/payment/order/ios.php';
$data = array(
  'MERCHANT' => 'PUBLICTESTHUF',
  'REFNOEXT' => '10101051461591307458 6',
  'HASH' => '9607a566c832821b8447eea204e6da1e'
);
```

by using cURL

```
$curlData = curl_init();
curl_setopt($curlData, CURLOPT_URL, $url);
curl_setopt($curlData, CURLOPT_POST, true);
curl_setopt($curlData, CURLOPT_POSTFIELDS, http_build_query($data));
curl_setopt($curlData, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curlData, CURLOPT_USERAGENT, 'curl');
curl_setopt($curlData, CURLOPT_TIMEOUT, 60);
curl_setopt($curlData, CURLOPT_FOLLOWLOCATION, true);
$resultXml = curl_exec($curlData);
curl_close($curlData);
$result = (array) simplexml_load_string($resultXml);
```

By using file_get_contents()

```
$options = array(
            'http' => array(
            'method' => 'POST',
            'header' =>
                        "Accept-language: en\r\n".
                        "Content-type: application/x-www-form-urlencoded\r\n",
            'content' => http_build_query($data, '', '&')
));
$context = stream_context_create($options);
$result = file_get_contents($url, true, $context);
```

PYTHON SOLUTION

```
# -*- coding:Utf-8 -*-
import httplib, urllib
merchant = "PUBLICTESTHUF"
renoext = "10101051461591307458 6"
hash = "9607a566c832821b8447eea204e6da1e"
params = urllib.urlencode({'MERCHANT': merchant, 'REFNOEXT': renoext, 'HASH': hash})
headers = {"Content-type": "application/x-www-form-urlencoded", "Accept": "text/plain", 'Accept-Charset': '
UTF-8'}
conn = httplib.HTTPSConnection("sandbox.simplepay.hu")
conn.request("POST", "/payment/order/ios.php", params, headers)
response = conn.getresponse()
result = response.read()
conn.close()
```

RUBY SOLUTION

```
require "net/http"
require "uri"
url = URI.parse("https://sandbox.simplepay.hu/payment/order/ios.php")
merchant = "PUBLICTESTHUF"
renoext = "101010514615913074586"
hash = "9607a566c832821b8447eea204e6da1e"
response = Net::HTTP.post_form(url, {"MERCHANT" => merchant, "REFNOEXT" => renoext, "HASH" =>
hash})
puts response.body
```

NODE.JS SOLUTION

```
var request = require('request');
url = 'https://sandbox.simplepay.hu/payment/order/ios.php?'
url += "MERCHANT" + "=" + "PUBLICTESTHUF" + "&"
url += "REFNOEXT" + "=" + "101010514615913074586" + "&"
url += "HASH" + "=" + "9607a566c832821b8447eea204e6da1e"
request.post({ url: url}, function (error, response, body) {
          console.log(body)
})
```

7. Query of IOS transactional data

By using IOS (Instant Order Status) a transaction's current status may be queried from the SimplePay system. An IOS call may any time be initiated from the merchant's page.

In the previous chapter, in the case of the sample code the IOS call was the example, thus its some elements will be repeated also in this chapter.

IOS call **URL**: https://sandbox.simplepay.hu/payment/order/**ios.php**

Necessary information
Merchant's ID
MERCHANT = PUBLICTESTHUF

Transaction's ID in the merchant's system
REFNOEXT = 101010514615913074586

Authenticating hash
HASH = 9607a566c832821b8447eea204e6da1e

In this case the value of HASH is generated based on the hash string calculated for the values of MERCHANT and REFNOEXT variables. The hast string must be composed and the hash value must be calculated in the way described in chapter 2.3.3.

If the IOS call is started with the above values, the SimplePay's server will provide the following XML response.

```
<order>
          <order_date>2016-04-25 13:35:07</order_date>
          <refno>99016764</refno>
          <refnoext>101010514615913074586</refnoext>
          <order_status>COMPLETE</order_status>
          <paymethod>Visa/MasterCard/Eurocard</paymethod>
          <hash>76621655113b7fd00d605075d28023f1</hash>
</order>
```

The following data are included in the response:

order_date: date of order
refno: SimplePay reference number belonging to the transaction
refnoext: merchant reference number belonging to the transaction
order_status: current status of the transaction
paymethod: payment method
hash: the hash which authenticates the response's data which is needed to be checked in the way described in chapter 2.3.3.

8. Providing IDN

IDN (Instant Delivery Notification) is a function used in case of a two-step payment method when the merchant finalizes all transactions with its own approval.

In case of a two-step payment the total payment is not carried out during the LiveUpdate but the reservation of the given amount. This means that the given amount is reserved on the customer's account but not debited. By sending IDN the debit may be initiated and the payment can be finalized.

In this case only those transactions will be accounted that have been confirmed with IDN message. The payment is made in two steps.

In the 1st step the transaction's total amount is reserved on the customer's card.

In the 2nd step the real debit takes place after sending the IDN.

NOTE: the merchant has 21 calendar days to send IDN in case of a two-step payment. If it does not give it until then, the reserved amount will automatically be unlocked.

If before the 21st day the merchant decides that it does not want to debit the amount (e.g. because it cannot fulfil the order), then instead of sending an IDN, it unlocks the reserved amount on the customer's account by using an IRN (Instant Refund Notification).

The transaction's status in this case will be PAYMENT_AUTHORIZED at the time when the amount is reserved. It will remain in this status until the following take place:

- The debit occurs by sending an IDN
- The amount on the customer's account is unlocked by sending an IRN
- after the expiry of 21 days the amount is automatically unlocked

After a successful IDN the transaction's status will be COMPLETE.

In order to use it, IDN function has to be configured on the SimplePay's side. For further information on the use, please contact our colleagues or customer service.

IDN call **URL**: https://sandbox.simplepay.hu/payment/order/idn.php

Necessary information:

Merchant's unique ID
MERCHANT = PUBLICTESTHUF

SimplePay ID of the transaction
ORDER_REF = 99017183

Total amount of order
ORDER_AMOUNT = 331

Currency of amount
ORDER_CURRENCY = HUF

Date of sending IDN
IDN_DATE = 2016-04-29 11:16:37

Hash value calculated from the above data
ORDER_HASH = 1a542b752502a620c243573f73bc8472

The value of ORDER_HASH in this case can be calculated based on those described in chapter 2.3.3.

By starting the above request, the SimplePay system will provide the following response if the call has been started with the appropriate data.

```
<epayment>99017183|1|OK|2016-04-29 11:16:37|39d5a164d54b10c707b5a0fd32088cf9</epayment>
```

The above output includes a string separated by | symbols between the <epayment> tags. The data are the following:

Transaction ID against which the IDN has been started
99017183

The call's returning value
1

The call's returning value in text
OK

The call's date
2016-04-29 12:16:55

The hash generated from the above data on the basis of which the merchant may authenticate the received reply.
39d5a164d54b10c707b5a0fd32088cf9

The IDN message may only be sent successfully once. If another IDN message is sent after a successful IDN you will receive an error message.

```
99017183|6|Error during the order's confirmation.|2016-04-29 11:29:03|d60f650b5994b4e063737c467365d4a6
```

9. Providing IRN

An IRN (Instant Refund Notification) may be sent against any transaction when the merchant intends to reimburse the transaction's total amount, or if it wants to unlock the amount reserved in the first step of a two-step transaction. In order to use this function there is no need for a configuration on the SimplePay's side.

An IRN may be sent regarding the transaction's total amount after each successful authorization (e.g. in case of a two-step payment) or an entire successful payment.

If the payment's status becomes COMPLETE, it can be also sent regarding a part amount.

The amount provided in the IRN has to be bigger than 0, and the aggregated amount of the one or more IRNs cannot be more than the transaction's total amount. The reimbursement is paid in the same currency as the original payment transaction.

NOTE: the SimplePay system initiates the reimbursement at the same time as the receipt of the IRN request. However, the amount is credited for the cardholder at the time in accordance with the unique accounting system of the bank which issued the card.

NOTE: The IRN may be sent manually from the merchant's admin interface. By selecting from the "Transaction list" it may be initiated by clicking on "Refund" button on the detailed data sheet of the given transaction.

IRN call **URL**: https://sandbox.simplepay.hu/payment/order/irn.php

Necessary information:

Merchant's unique ID
MERCHANT = PUBLICTESTHUF

SimplePay ID of the transaction
ORDER_REF = 99017212

Total amount of order
ORDER_AMOUNT = 331

Currency of amount
ORDER_CURRENCY = HUF

Date of IRN sending
IRN_DATE = 2016-04-29 12:59:57

Amount to be refunded
AMOUNT = 331

Hash value calculated from the above data
ORDER_HASH = 14aa6a4cefe5698cc5b45dc204da80e8

The value of ORDER_HASH in this case can be calculated based on those described in chapter 2.3.3.

By starting the above request, the SimplePay system will provide the following response if the call has been started with the appropriate data.

```
<epayment>99017212|1|OK|2016-04-29 12:59:57|2c071f3bc310ba6a2df2f93095ac2c91</epayment>
```

The above output includes a string separated by | symbols between the <epayment> tags. The data are the following:

Transaction ID against which the IDN has been started
99017212
The call's returning value
1

The call's textual returning value
OK

The call's date
2016-04-29 12:59:57

The hash generated from the above data on the basis of which the merchant may authenticate the received reply.
2c071f3bc310ba6a2df2f93095ac2c91