1 LICENSE CONDITIONS WEB SERVICE

1. DEFINITIONS

1.1 ICEPAY Web Service:
The software product provided by ICEPAY B.V. on an ‘as is’ basis without any warranty of any kind.

1.2 License:
A written public act of the Dutch central bank or other governmental body which provides ICEPAY B.V. with these rights.

2. USER LICENSE CONDITIONS WEB SERVICE

2.1 This User License Agreement applies to the use of this ICEPAY Web Service, as supplied by ICEPAY B.V. (further referred to as ICEPAY B.V.).

2.2 BY USING ICEPAY WEBSERVICE YOU FULLY AGREE TO THE CONDITIONS OF THIS USER LICENSE AGREEMENT. IF YOU DO NOT AGREE TO THIS LICENSE AGREEMENT, YOU SHOULD REFRAIN FROM USING THE ICEPAY WEBSERVICE.

2.3 You may only use the ICEPAY Web Service if such is directly obtained from ICEPAY B.V. and provided from icepay.com and if you or the organization where you work has entered into an official contract with ICEPAY B.V. and therefore is a Customer in accordance with these conditions.

2.4 This User License Agreement and the use of the ICEPAY Web Service are governed by the laws of The Netherlands. Any disagreement will be placed before a qualified court in The Hague, The Netherlands. The United Nations Convention on Contracts for the International Sale of Goods (CISG) is not applicable.

3. USER LICENSE ICEPAY WEBSERVICE

3.1 ICEPAY B.V. grants Customer the non exclusive right to use this ICEPAY Web Service and corresponding documentation. The license shall go into effect after Customer has fulfilled all its obligations.

4. WARRANTY DISCLAIMER

4.1 The ICEPAY Web Service is made available on an “as is” basis only and without any warranty or indemnity of any kind.

4.2 ICEPAY B.V. makes no warranties, conditions, indemnities, representations or terms, express or implied, whether by statute, custom, or otherwise as to any other matters, including but not limited to non-infringement of third party rights, integration, accuracy, security, availability, satisfactory quality, merchantability or fitness for any particular purpose.
5. **LIMITATIONS TO INDEMNIFICATION & LIABILITY**

5.1 Customer agrees to indemnify ICEPAY B.V. from all liability, losses, actions, damages or claims (including all reasonable costs and attorney costs) which flow forth or are regarding the use or dependency upon the ICEPAY Web Service.

5.2 Under no circumstances will ICEPAY B.V. be liable to Customer, or any other person or entity, for any loss of use, revenue or profit, lost or damaged data, or other commercial or economic loss or for any direct, indirect, special, statutory, or consequential damages whatsoever related to the use or reliance upon ICEPAY Web Service, even if advised of the possibility of such damages or if such damages are foreseeable. This limitation shall apply to each breach of this User License Agreement by ICEPAY B.V.

6. **ADDITIONAL WORK & SUPPORT**

6.1 All activities that ICEPAY B.V. must perform upon request of Customer related to the use of the ICEPAY Web Service, which has been made available at no charge, shall be invoiced as additional work (or support) on the basis of actual costs according to the applicable rates of ICEPAY B.V.

6.2 (Future) incompatibility problems (products are unable to interoperate with each other) can be resolved on the basis of additional work.

6.3 It will be assumed that Customer has agreed with the performance of additional work and the connected costs, if Customer has allowed additional work to take place without raising objections in writing prior to the commencement of additional work.

7. **DURATION**

7.1 This agreement is effective as of the moment of acceptance and may be terminated at any time by ICEPAY B.V. whereby a notice period of one week shall apply.

8. **GENERAL CONDITIONS/APPLICABILITY**

8.1 The General Conditions ICEPAY apply to the agreement. The General Conditions ICEPAY are filed at the Chamber of Commerce in The Hague under number 27348492. The applicability of purchase conditions or any other conditions from Customer or third parties is, then, expressly rejected by ICEPAY B.V. Customer explicitly declares to have received the General Conditions ICEPAY and to agree with the General Conditions ICEPAY.
## REVISION SHEET

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Version</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-Feb-2018</td>
<td>IT</td>
<td>1.7.0</td>
<td>Added FundsTransfer,Attributes in GetPayment response &amp; other small improvements</td>
</tr>
<tr>
<td>11-Jul-2017</td>
<td>IT</td>
<td>1.6.1</td>
<td>Removed Phone &amp; SMS payment methods. Expanded postback documentation, expanded CheckoutExtended.</td>
</tr>
<tr>
<td>6-Feb-2017</td>
<td>IT</td>
<td>1.6.0</td>
<td>Added implementation of AchterafBetalen</td>
</tr>
<tr>
<td>11-Nov-2015</td>
<td>Marketing</td>
<td>1.5.5</td>
<td>- Checked all links to latest documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Replaced 2 links to Parameter Sheet – pages 16 &amp; 17</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

1 License Conditions Web Service ............................................................................................................ 2

Revision sheet .................................................................................................................................................. 4

Table of contents.............................................................................................................................................. 5

Preface .............................................................................................................................................................. 7

*How this document is organized* .................................................................................................................. 7

*Some assumptions about the reader* ........................................................................................................... 8

*Latest version* .............................................................................................................................................. 8

1 ICEPAY web service essentials .................................................................................................................... 9

1.1 *What is the ICEPAY Web Service?* ....................................................................................................... 9

1.2 *Where are the web services located?* ..................................................................................................... 9

2 Checkout Web Service Web methods ......................................................................................................... 12

2.1 *PHP* ......................................................................................................................................................... 12

2.2 *Checkout* ................................................................................................................................................ 12

2.3 *VaultCheckout* ....................................................................................................................................... 17

2.4 *AutoCheckout* ....................................................................................................................................... 19

2.5 *FundsTransfer* ....................................................................................................................................... 21

2.6 *CheckoutExtended* ................................................................................................................................. 23

2.7 *GetPayment* .......................................................................................................................................... 24

3 Postback notification ................................................................................................................................... 27

3.1 *Calculating the Checksum* ................................................................................................................... 29

3.2 *Handling the Postback Notification* ..................................................................................................... 30

3.3 *Testing your Postback script* ................................................................................................................ 32

4 Reporting Web Service ................................................................................................................................ 33

4.1 *Allow reporting access* ........................................................................................................................ 33

4.2 *How does it work?* ................................................................................................................................... 33

4.3 *CreateSession* ...................................................................................................................................... 35

February 6, 2018 – Version 1.7.0 ICEPAY WEBSERVICE – Implementation Guide
4.4 KillSession ................................................................................................................. 37
4.5 MonthlyTurnoverTotals ............................................................................................... 39
4.6 GetMerchants ............................................................................................................. 41
4.7 GetPaymentMethods .................................................................................................. 43
4.8 SearchPayments ......................................................................................................... 45

5 Refund Web Service ....................................................................................................... 48
  5.1 Refunding queue ....................................................................................................... 48
  5.2 Notification ............................................................................................................... 48
  5.3 Supported payment methods ...................................................................................... 48
  5.4 Test and live transactions .......................................................................................... 49
  5.5 RequestRefund .......................................................................................................... 49
  5.6 CancelRefund ............................................................................................................. 51
  5.7 GetPaymentRefunds ................................................................................................. 52

6 Exceptions ...................................................................................................................... 55
  6.1 Error codes 0000 – 0016 .......................................................................................... 55
  6.2 Error codes 1000 – 1003 ......................................................................................... 56
  6.3 Error codes 2000 – 2007 ......................................................................................... 56

7 Appendix A: flow of the basic integration approach using Checkout web method ............ 58

8 Appendix B: Generic ExtendedCheckout Xml ................................................................. 59
  8.1 XSD ......................................................................................................................... 59
  8.2 Sample XML ............................................................................................................. 62

9 Appendix C: AchterAfbetalen CheckoutExtended XML .................................................. 63
  9.1 AchterAfbetalen.xsd ................................................................................................. 64
  9.2 SimpleTypes.xsd ...................................................................................................... 66
  9.3 ComplexTypes.xsd ................................................................................................. 69
  9.4 Sample XML ............................................................................................................. 69

10 Appendix D: Supported Payment Attributes ............................................................... 71
  10.1 Payvision Authorisation Code .................................................................................. 71
This document is written for software developers who want to integrate online payment methods offered by ICEPAY into their website or computer system using a web service provided by ICEPAY.

HOW THIS DOCUMENT IS ORGANIZED

Here is a brief summary of the chapters in this document:

Chapter 1 “ICEPAY Web Service Essentials” covers everything you need to know about the ICEPAY Web Service such as the location of the web service and its WSDL.

Chapter 2 describes the possibilities of the ICEPAY Checkout web service.

Chapter 3 “Postback Notifications” covers the Postback notification, which is how your system will be informed regarding the status of your transactions.

Chapter 4 “Reporting web service” solely covers the ICEPAY Reporting Web Service, which allows your client applications to query statistical data from your ICEPAY account.

Chapter 5 “Refund web service” solely covers the ICEPAY Refund Web Service, which allows your client applications to programmatically initiate refund requests.

Chapter 6 “Exceptions” described the different exceptions you might expect when interacting with the webservice.
SOME ASSUMPTIONS ABOUT THE READER

We assume that you, the reader, are an experienced developer and that you are comfortable with object-oriented concepts and web services. Some knowledge about the payment method that you want to implement can be useful, but is not necessary.

LATEST VERSION

Always check if you have the latest version of this document. The latest version of this document can be found here:

1 ICEPAY WEB SERVICE ESSENTIALS

This chapter describes the essential concepts of the ICEPAY Web Service and its architecture, enabling you to integrate with ICEPAY.

1.1 WHAT IS THE ICEPAY WEB SERVICE?

The ICEPAY Web Service is a technology offered by ICEPAY. It allows you to do server-to-server communication with ICEPAY. The ICEPAY Web Service allows you to initiate a transaction, query transaction details, etc. You can access the web service with SOAP.

1.2 WHERE ARE THE WEB SERVICES LOCATED?

The ICEPAY Web Service consists of three web services, each containing a set of operations or web methods that are specifically designed for a specific purpose.

1.2.1 CHECKOUT WEB SERVICE

The Checkout Web Service contains a set of web methods that are designed for checkout purposes such as creating payments, querying payment information, etc. You can read more about the possibilities of this web service in chapter 2.

Use this web service if you are building an online game, auction website, web shop, etc. and you need the possibility for your end-users to pay for your platform’s service or products.

Note: Please make sure the default encoding is set to UTF-8

The URL for the Checkout Web Service is located here:

https://connect.icepay.com/webservice/ICEPAY.svc

The WSDL can be found here:

https://connect.icepay.com/webservice/ICEPAY.svc?wsdl

⚠️ ONLY HTTPS IS SUPPORTED!
1.2.2 REPORTING WEB SERVICE

The Reporting Web Service contains a set of web methods that are designed for querying information from your ICEPAY account such as statistical information.

Use this web service if you want to query statistical information from your ICEPAY account using your own scripts, or if you want to write a mobile application that is able to view ICEPAY information to the end-user.

Please read chapter 4 for more detailed information about this web service.

**Note: Please make sure the default encoding is set to UTF-8**

The URL for the Reporting Web Service is located here:


The WSDL can be found here:


⚠️ ONLY HTTPS IS SUPPORTED!

1.2.3 REFUND WEB SERVICE

The Refund Web Service contains a set of web methods that are designed for you to programmatically perform refund requests, query refund requests of a payment, and cancel refund requests.

Use this web service if you want to have refund capability in your own backend system.

Please read chapter 5 for more detailed information about this web service.

**Note: Please make sure the default encoding is set to UTF-8**

The URL for the Refund Web Service is located here:

https://connect.icepay.com/webservice/Refund.svc

The WSDL can be found here:

https://connect.icepay.com/webservice/Refund.svc?wsdl
ONLY HTTPS IS SUPPORTED!
This chapter covers the various web methods that are available in the Checkout Web Service. Using the web methods you can e.g. initialize transactions or query transaction details.

Please keep in mind that web methods can raise an exception if the input is incorrect. Therefore, it is recommended that you ALWAYS enclose your calls in a Try ... Catch block.

2.1 PHP

When you write a client in PHP for the ICEPAY Web Service using the SOAP extension, responses returned are differently than is described in this document.

First of all, responses returned have a slightly different naming convention than what is described in this document. For instance, if you invoke the Checkout web method, the response object is called CheckoutResult instead of CheckoutResponse. Other than this, there are no known naming differences.

When a list of objects is returned, which the GetPremiumRateNumbers web method does (please see paragraph 4.8), PHP will format this differently depending on the number of elements. If there is only one element, it will immediately return it as an object. If there are two elements, it will be returned as an array which contains two objects.

2.2 CHECKOUT

The Checkout web method allows you to initialize a new payment in the ICEPAY system for most of the payment methods that you have access to. For AchterafBetalen you should use CheckoutExtended instead.

You will have to include a CheckoutRequest object which contains the information that is necessary for the initialization of the transaction, such as the currency, the amount that you want to charge, payment method you want to use, etc.

Calling the Checkout web method will return you the CheckoutResponse object, which contains data such as the Payment Screen URL.

Providing the correct parameters

Every payment method has its own set of rules that you must follow. For instance, the payment method iDEAL accepts payments in Euro’s only. In order to provide the correct parameters to the CheckoutRequest object, please consult the following document:

2.2.1 REQUEST

Please keep in mind that when you populate the CheckoutRequest object with payment initialization values, that the combination of values are allowed. For instance, some payment methods do not support certain currencies, or have a strict limit of the amount, etc. For an up-to-date list of possible combinations, please go to: https://icepay.com/downloads/tech-docs/ICEPAY_Supported_Parameters_Sheet.pdf

If any member is invalid or if the combination is invalid, then the web service will raise an exception.

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest of all the members.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>yyyy-mm-ddTh:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>Amount</td>
<td>This is the amount (in cents) that will be charged</td>
<td>Integer</td>
</tr>
<tr>
<td>Country</td>
<td>This member will be interpreted differently depending on the chosen payment method. It is used to validate if the payment method is supported in a certain country.</td>
<td>Integer</td>
</tr>
<tr>
<td>Currency</td>
<td>This is the currency</td>
<td>String</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the payment</td>
<td>String</td>
</tr>
<tr>
<td>EndUserIP</td>
<td>IP address of the end-user</td>
<td>String</td>
</tr>
<tr>
<td>Issuer</td>
<td>This is the issuer of the payment method</td>
<td>String</td>
</tr>
<tr>
<td>Language</td>
<td>Language of the payment screen</td>
<td>String</td>
</tr>
<tr>
<td>OrderID</td>
<td>A unique code up to 10 characters. It is recommended to use the primary key field of your local transactions table</td>
<td></td>
</tr>
<tr>
<td>PaymentMethod</td>
<td>This is the payment method that will be used for the transaction initialization</td>
<td>String</td>
</tr>
<tr>
<td>Reference</td>
<td>Custom information that you want to include, e.g. primary key of your local transactions table (up to 50 characters)</td>
<td>String</td>
</tr>
<tr>
<td>URLCompleted</td>
<td>The URL to which the end-user will be redirected</td>
<td>String</td>
</tr>
</tbody>
</table>
If you do not set this member then the URLEndUser will be used

<table>
<thead>
<tr>
<th>URLError</th>
<th>The URL to which the end-user will be redirected</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If you do not set this member then the URLErr that you set in your ICEPAY account will be used</td>
<td></td>
</tr>
</tbody>
</table>

The `checksum` member is a SHA1 message digest, which can be calculated by concatenating the values that are specified below. The values need to be separated by the pipe (|) character.

1. Secret code of your API key
2. MerchantID
3. Timestamp
4. Amount
5. Country
6. Currency
7. Description
8. EndUserIP
9. Issuer
10. Language
11. OrderID
12. PaymentMethod
13. Reference
14. URLEndUser
15. URLErr

Example

```
secret|12345|2017-06-09T01:30:00Z|100|NL|EUR|Test|127.0.0.1|AMEX|NL|1|CREDITCARD|MyReference|
```
### 2.2.2 RESPONSE

Returns the `CheckoutResponse` object, which contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest of all the members. You can use this value to match</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>your own checksum to see if the response is actually from ICEPAY.</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>This will be the current UTC time that has the following format: yyyy-mm-ddThh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>Amount</td>
<td>The requested amount in cents</td>
<td>Integer</td>
</tr>
<tr>
<td>Country</td>
<td>This is the requested country code</td>
<td>String</td>
</tr>
<tr>
<td>Currency</td>
<td>This is the requested currency</td>
<td>String</td>
</tr>
<tr>
<td>Description</td>
<td>This is the specified description</td>
<td>String</td>
</tr>
<tr>
<td>EndUserIP</td>
<td>This is the provided end user IP</td>
<td>String</td>
</tr>
<tr>
<td>Issuer</td>
<td>This is the requested issuer</td>
<td>String</td>
</tr>
<tr>
<td>Language</td>
<td>This is the requested language code</td>
<td>String</td>
</tr>
<tr>
<td>OrderID</td>
<td>This is your provided Order ID</td>
<td>String</td>
</tr>
<tr>
<td>PaymentID</td>
<td>This is the generated ICEPAY transaction ID.</td>
<td>Integer</td>
</tr>
<tr>
<td>PaymentMethod</td>
<td>This is the requested Payment Method</td>
<td>String</td>
</tr>
<tr>
<td>PaymentScreenURL</td>
<td>This is the URL of the payment screen (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ProviderTransactionID</td>
<td>This is the transaction ID of the issuer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>Reference</td>
<td>This is the specified reference information</td>
<td>String</td>
</tr>
<tr>
<td>TestMode</td>
<td>Indicates whether the transaction was initialized in test mode. This is</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td>true if your API key is still in test mode. To switch to production</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mode, please contact your account manager</td>
<td></td>
</tr>
<tr>
<td>URLCompleted</td>
<td>This is the page to which the end-user will be redirected to after a</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>successful transaction</td>
<td></td>
</tr>
<tr>
<td><strong>Snackbar</strong></td>
<td>This is the page to which the end-user will be redirected to after a failed or cancelled transaction</td>
<td><strong>String</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

The checksum is a SHA1 message digest which can be calculated by concatenating the values below in the specified order. The values also need to be separated by the pipe (|) character:

1. Secret code of your API key
2. MerchantID
3. Timestamp
4. Amount
5. Country
6. Currency
7. Description
8. EndUserIP
9. Issuer
10. Language
11. OrderID
12. PaymentID
13. Payment Method
14. PaymentScreenURL
15. ProviderTransactionID
16. Reference
17. TestMode
18. URLCompleted
19. URLError

**Example**

```
secret|12345|2017-06-09T01:30:00Z|100|NL|EUR|Test|127.0.0.1|AMEX|NL|1|100123456|CREDITCARD|https://live.icepay.eu|01234|MyReference|false||
```
2.3 VAULTCHECKOUT

This method is called when you need to vault consumer information like a credit card or bank account number in order to perform an automatic checkout or Funds Transfer in the future. This method is only available for payment methods iDeal and CreditCard.

That information is stored after the above mentioned payment has been successfully completed. This method MUST be called if a recurring payment or Funds Transfer in the future is desired.

2.3.1 REQUEST

Requires a VaultCheckoutRequest object.

2.3.1.1 PARAMETERS LIST (VAULTCHECKOUTREQUEST)

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>Is the ID of the merchant provided by ICEPAY when creating a new merchant.</td>
<td>integer</td>
</tr>
<tr>
<td>PaymentMethod</td>
<td>It is a description of the payment method which will be used in the payment at the moment vaulting is supported for “credit card” and “ideal”</td>
<td>String</td>
</tr>
<tr>
<td>Amount</td>
<td>Ids the amount of the transaction</td>
<td>integer</td>
</tr>
<tr>
<td>Language</td>
<td>Is the string corresponding to the iso code of the language ex. Dutch is ‘NL’</td>
<td>string</td>
</tr>
<tr>
<td>Currency</td>
<td>Is the String corresponding to the iso code of the currency ex. Euro is ‘EUR’</td>
<td>string</td>
</tr>
<tr>
<td>Country</td>
<td>Is the string corresponding to the iso code of the country ex. Netherlands is ‘NL’</td>
<td>string</td>
</tr>
<tr>
<td>Issuer</td>
<td>Is the issuer connected to the payment method Ex. For credit card can be ‘VISA’ or ‘MASTERCARD’</td>
<td>string</td>
</tr>
<tr>
<td>ConsumerID</td>
<td>Is the id which is wished to link to the consumer credit card or bank account to perform automatic checkouts. It can be alphanumeric.</td>
<td>string</td>
</tr>
<tr>
<td>Timestamp</td>
<td>It is the timestamp referring to the payment</td>
<td>string</td>
</tr>
<tr>
<td>OrderID</td>
<td>It is the Unique OrderId of the transaction</td>
<td>string</td>
</tr>
<tr>
<td>Description</td>
<td>It is the description which appears along the payment in the ICEPAY’s environment.</td>
<td>String</td>
</tr>
<tr>
<td>EndUserIP</td>
<td>Is the ip address of the customer's machine</td>
<td>string</td>
</tr>
<tr>
<td><strong>URLCompleted</strong></td>
<td>Is the url where the user is redirected after successful payment.</td>
<td>string</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>URLError</strong></td>
<td>Is the url where the user is redirected after erroneous payment.</td>
<td>string</td>
</tr>
<tr>
<td><strong>Reference</strong></td>
<td>This field can be empty</td>
<td>string</td>
</tr>
<tr>
<td><strong>Checksum</strong></td>
<td>Is the checksum of all the fields used in a regular checkout therefore excluding ConsumerID</td>
<td>string</td>
</tr>
</tbody>
</table>

### 2.3.2 RESPONSE

Returns the `CheckoutResponse` object (see paragraph 2.2.2).

### 2.3.3 USAGE

Create an object type `VaultCheckoutRequest` and feed the fields described above with relevant data.

```csharp
ICEPAYService.VaultCheckoutRequest cr = new VaultCheckoutRequest();
    cr.MerchantID = MerchantID";
    cr.PaymentMethod = pmethodId";
    ................
```

Pass to the method `VaultCheckout` the created object and handle response.

```csharp
CheckoutResponse response = client.VaultCheckout(cr);
handleResponse(response)
```

The response can be handled as a standard ICEPAY's response.

The code below is an example in C# but is not meant to work if copy/pasted:

```csharp
public void VaultCheckout()
{
    ICEPAYClient client = new ICEPAYClient();
    ICEPAYService.VaultCheckoutRequest cr = new VaultCheckoutRequest();
    cr.MerchantID = MerchantID";
    cr.PaymentMethod = pmethodId";
```
cr.Amount = Amount;
cr.Language = “Country”;
cr.Currency = “Currency”;
cr.Country = ”Country”;
cr.Issuer = ”Issuer”;
cr.ConsumerID = ”ConsumerID”;
cr.Timestamp = DateTime.Now.ToShortDateString();
cr.OrderID = this.newOrderId().ToString();
cr.Description = “description”
cr.EndUserIP = ”REMOTE_ADDR”;
cr.URLCompleted = urlCompleted

cr.URLError = urlError

cr.Reference = ””;
cr.Checksum = this.CalculateChecksum(cr);

CheckoutResponse response = client.VaultCheckout(cr);

handleResponse(response)

}  

2.4 AUTOCHECKOUT

This method is called when is needed to perform an automatic checkout using the consumerID that is stored by a VaultCheckout. It is available for payment methods CreditCard and DirectDebit only. Again, an AutoCheckout object must first be created before it can be used. It must always be preceded by a succesfull VaultCheckout for the given ConsumerID.

2.4.1 REQUEST

Requires an AutomaticCheckoutRequest object.

2.4.1.1 PARAMETERS LIST (AUTOMATICCHECKOUTREQUEST)

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>is the Id of the ICEPAY merchant which is performing the automatic checkout</td>
<td>Integer</td>
</tr>
</tbody>
</table>
### PaymentMethod
Can have two values:
- 'ddebit' this is required to perform an automatic checkout prior storing a iDeal account number
- 'creditcard' this is required to perform an automatic checkout prior storing a credit card number.

The keywords are not case sensitive.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>is the amount in cents which is desired to bill.</td>
<td>Integer</td>
</tr>
<tr>
<td>Language</td>
<td>is the language code of the billing ex for the Netherlands 'NL'</td>
<td>String</td>
</tr>
<tr>
<td>Currency</td>
<td>is the currency code ex. for euro ‘EUR’</td>
<td>String</td>
</tr>
<tr>
<td>Country</td>
<td>is the country code ex for the Netherlands ‘NL’</td>
<td>String</td>
</tr>
<tr>
<td>Issuer</td>
<td>this depends on the payment method insert before:</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerID</td>
<td>this is the consumer id which was vaulted previously using the VaultCheckout method.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>represent a moment in time which the operation was performed.</td>
<td>String</td>
</tr>
<tr>
<td>OrderID</td>
<td>is the order id corresponding to the transaction</td>
<td>String</td>
</tr>
<tr>
<td>Description</td>
<td>a compulsory description can be left empty</td>
<td>String</td>
</tr>
<tr>
<td>EndUserIP</td>
<td>is the ip address of the machine from which the action is performed.</td>
<td>String</td>
</tr>
<tr>
<td>URLCompleted</td>
<td>is the url where the user lands on a successful transaction.</td>
<td>String</td>
</tr>
<tr>
<td>Reference</td>
<td>can be and additional remark, can be left empty.</td>
<td>String</td>
</tr>
<tr>
<td>Checksum</td>
<td>Is the checksum of all the fields used in a regular checkout therefore excluding ConsumerID</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>String, represent a moment in time which the operation was performed.</td>
<td>String</td>
</tr>
</tbody>
</table>

### 2.4.2 RESPONSE

Returns the `AutomaticCheckoutResponse` object

### 2.4.2.1 PARAMETERS LIST (AUTOMATICCHECKOUTRESPONSE)
<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>is the Id of the ICEPAY merchant which is performing the automatic checkout</td>
<td>Integer</td>
</tr>
<tr>
<td>Success</td>
<td>Indicates whether the operation had been successfully completed</td>
<td>Boolean</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>Indicates the timestamp of the response.</td>
<td>String</td>
</tr>
<tr>
<td>PaymentID</td>
<td>Is the ICEPAY Id of the payment</td>
<td>Integer</td>
</tr>
<tr>
<td>ErrorDescription</td>
<td>If errors are present indicates the message of the error</td>
<td>String</td>
</tr>
<tr>
<td>Checksum</td>
<td>Is the checksum SHA1 crypted of the above fields</td>
<td>String</td>
</tr>
</tbody>
</table>

2.4.3 USAGE

As for the VaultCheckout create an object AutomaticCheckoutRequest and pass it to the method AutoCheckout.

Finally process Response.

2.5 FUNDTRANSFER

This method is called to perform a Funds Transfer using the consumerID that is stored by a VaultCheckout. A Funds Transfer is a way of transferring funds (back) to a Consumer on their payment method. It differs from a refund in that the transfer is not limited to the amount of the original order. It is available for payment method CreditCard only. It must always be preceded by a successful VaultCheckout for the given ConsumerID.

2.5.1 REQUEST

Requires an FundsTransferRequest object.

2.5.1.1 PARAMETERS LIST (FUNDTRANSFERREQUEST)

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>is the Id of the ICEPAY merchant which is performing the funds transfer</td>
<td>Integer</td>
</tr>
</tbody>
</table>
### 2.5.2 RESPONSE

Returns the `FundsTransferResponse` object

#### 2.5.2.1 PARAMETERS LIST (FUNDTRANSFERRESPONSE)

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
</table>
| **PaymentMethod** | Must have value:  
  - ‘creditcard’ this is required to perform an automatic checkout prior storing a credit card number.  
  The keywords are not case sensitive. | String    |
| **Amount**      | is the amount in cents which is desired to transfer. Note that is a positive amount in the request but as it is an outgoing transfer it will be reflected on your statement as a negative amount. | Integer   |
| **Language**    | is the language code of the billing ex for the Netherlands ‘NL’              | String    |
| **Currency**    | is the currency code ex. for euro ‘EUR’                                     | String    |
| **Country**     | is the country code ex for the Netherlands ‘NL’                             | String    |
| **Issuer**      | For ‘creditCard’ must be ‘FUNDTRANSFER’                                     | String    |
| **ConsumerID**  | this is the consumer id which was vaulted previously using the VaultCheckout method. | String    |
| **Timestamp**   | represent a moment in time which the operation was performed.               | String    |
| **OrderID**     | is the order id corresponding to the transaction                            | String    |
| **Description** | a description can be left empty                                             | String    |
| **EndUserIP**   | is the ip address of the machine from which the action is performed.        | String    |
| **URLCompleted**| is the url where the user lands on a successful transaction.                | String    |
| **Reference**   | can be an additional remark, can be left empty.                            | String    |
| **Checksum**    | Is the checksum of all the fields used in a regular checkout therefore excluding ConsumerID | String    |
| **Timestamp**   | String, represent a moment in time which the operation was performed.      | String    |
MerchantID is the Id of the ICEPAY merchant which is performing the automatic checkout

Success Indicates whether the operation had been successfully completed

TimeStamp Indicates the timestamp of the response.

PaymentID Is the ICEPAY Id of the payment

ErrorDescription If errors are present indicates the message of the error

Checksum Is the checksum SHA1 crypted of the above fields

2.5.3 USAGE

As for the VaultCheckout create an object FundsTransferRequest and pass it to the method FundsTransfer.

Finally process Response.

2.6 CHECKOUTEXTENDED

The CheckoutExtended web method is almost identical to Checkout with the difference that it includes an extra XML field. This XML field must be populated with information about the order such as customer and product information.

⚠️ Payment methods that require this method: AchterafBetalen

2.6.1 REQUEST

You need to include the CheckoutExtendedRequest object. It INHERITS ALL THE MEMBERS OF THE CHECKOUTREQUEST OBJECT. In addition it has a member called ‘XML’ of the data type ‘string’.

The XML member needs to be populated with a proper XML message. The exact format of the XML message is documented in "Appendix B: Generic CheckoutExtended XML". It contains the XSD of the XML message, as well as an example.
For PaymentMethod AchterafBetalen a different XML schema should be used. This is documented in “Appendix C: AchterafBetalen CheckoutExtended XML”.

The Checkum in the request must be calculated similar to the standard Checkout with the addition of the XML field. The XML field should be the last field in the Checksum source string (immediately following URLError).

2.6.2 RESPONSE

Please see 2.2.2

2.7 GETPAYMENT

Calling the GetPayment web method will return more information about a payment. Most of the information that is returned is already being sent back via the regular Postback notification (please see chapter 5). You can use this web method to requery the information in case you have lost it.

2.7.1 REQUEST

You need to include the GetPaymentRequest object, which consists of the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest of all the members.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>yyyy-mm-ddThh:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>PaymentID</td>
<td>This is the ICEPAY PaymentID. You will receive this when you initialize a payment or in your Postback.</td>
<td>Integer</td>
</tr>
</tbody>
</table>

The checksum member is a SHA1 message digest, which can be calculated by concatenating the values that are specified below. The values need to be separated by the pipe (|) character.

1. Secret
2. MerchantID
3. Timestamp
4. PaymentID

Example
### 2.7.2 RESPONSE

Returns the `GetPaymentResponse` object, which contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest which is calculated according to the formula in 4.8.2.1</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td>PaymentID</td>
<td>The ICEPAY PaymentID</td>
<td>Integer</td>
</tr>
<tr>
<td>Amount</td>
<td>The requested amount of the payment</td>
<td>Integer</td>
</tr>
<tr>
<td>Currency</td>
<td>The requested currency of the payment</td>
<td>Integer</td>
</tr>
<tr>
<td>Description</td>
<td>The description specified by the merchant during the initialization</td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>The numbers of seconds dialed. Only applicable for phone payments</td>
<td>Integer</td>
</tr>
<tr>
<td>ConsumerName</td>
<td>Name of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerAccountNumber</td>
<td>IBAN of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerBIC</td>
<td>BIC number of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerAddress</td>
<td>Address of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerHouseNumber</td>
<td>House number of the address of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerCity</td>
<td>City of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerCountry</td>
<td>Country of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerEmail</td>
<td>E-mail address of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerPhoneNumber</td>
<td>Phone number of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>ConsumerIPAddress</td>
<td>IP address of the consumer (if available)</td>
<td>String</td>
</tr>
<tr>
<td>Issuer</td>
<td>Requested payment method issuer</td>
<td>String</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>OrderID</td>
<td>The OrderID specified by the merchant during the initialization</td>
<td>String</td>
</tr>
<tr>
<td>OrderTime</td>
<td>The time when the payment got started. In GMT.</td>
<td>String</td>
</tr>
<tr>
<td>PaymentMethod</td>
<td>The requested payment method</td>
<td>String</td>
</tr>
<tr>
<td>PaymentTime</td>
<td>The time indicating when the payment got closed/completed (either successful or not successful). In GMT.</td>
<td>String</td>
</tr>
<tr>
<td>Reference</td>
<td>The reference specified by the merchant during the initialization</td>
<td>String</td>
</tr>
<tr>
<td>Status</td>
<td>The status of the payment (please see 5.1.1 for possible statuses)</td>
<td>String</td>
</tr>
<tr>
<td>StatusCode</td>
<td>A description giving you more information on the status</td>
<td>String</td>
</tr>
<tr>
<td>TestMode</td>
<td>Indicates whether the payment was created when the API key was still in test mode.</td>
<td>Boolean</td>
</tr>
<tr>
<td>Attributes</td>
<td>Optional list of name-value pairs. See Appendix D for possible attributes that may be returned.</td>
<td>List of string / string</td>
</tr>
</tbody>
</table>

The *checksum* member is a SHA1 message digest, which can be calculated by concatenating the values below. The values need to be separated by the pipe (|) character.

1. Secret code of your API key
2. MerchantID
3. Timestamp
4. PaymentID
5. Amount
6. ConsumerAccountNumber
7. ConsumerAddress
8. ConsumerCity
9. ConsumerCountry
10. ConsumerEmail
11. ConsumerHouseNumber
12. ConsumerIPAddress
13. ConsumerName
14. ConsumerPhoneNumber
15. Currency
16. Description
17. Duration
18. Issuer
19. OrderID
20. OrderTime
21. PaymentMethod
3 POSTBACK NOTIFICATION

While the consumer is doing his payment, ICEPAY will report all status changes back to the Merchant with a server-to-server POST to the URL. You can set this URL in your ICEPAY account at https://portal.icepay.com/

Note: The script in URLPostback should not generate any output or errors. It is very important that this script works well, otherwise payments will be aborted! We would advise you to always return HTTP 200 (The default “OK” response of a web server. An empty page does exactly that.), if you have processed the request to prevent our server to repeat the request. Only return HTTP error if you have (good) reason for it (like db connection error or update error).

Note: Our postback uses the UTF-8 character encoding format.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
<th>Sample</th>
<th>Can be empty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Payment status as OK, OPEN, ERR, REFUND, CBA</td>
<td>String(10)</td>
<td>OPEN</td>
<td>N</td>
</tr>
<tr>
<td>StatusCode</td>
<td>A short description of the status. We will use the codes as received from the payment method provider</td>
<td>String(100)</td>
<td>Completed</td>
<td>N</td>
</tr>
<tr>
<td>Merchant</td>
<td>Your MerchantID</td>
<td>Numeric</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>OrderID</td>
<td>The OrderID member that was provided in the request object</td>
<td>String(10)</td>
<td>1234567</td>
<td>N</td>
</tr>
<tr>
<td>PaymentID</td>
<td>A unique numeric value that identifies this payment in our system</td>
<td>Numeric</td>
<td>12345</td>
<td>N</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
<td>Value</td>
<td>Required</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>Reference</td>
<td>The Reference member that was provided in the request object</td>
<td>String(50)</td>
<td>Z1234567</td>
<td>Y</td>
</tr>
<tr>
<td>TransactionID</td>
<td>This value is created by the payment method provider / bank and showed on the user’s bank statement</td>
<td>String(50)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerName</td>
<td>Name of the bank account owner</td>
<td>String(100)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerAccountNumber</td>
<td>Last 4 digits of accountnumber from which the payment was done, if received from the bank</td>
<td>String(100)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerAddress</td>
<td>Consumer address/street as filled in payment form</td>
<td>String(100)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerHouseNumber</td>
<td>Consumer house number as filled in payment form</td>
<td>String(10)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerCity</td>
<td>Consumer city as filled in payment form</td>
<td>String(100)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerCountry</td>
<td>Consumer country as filled in payment form</td>
<td>String(100)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerEmail</td>
<td>Consumer email value as filled in payment form</td>
<td>String(200)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerPhoneNumber</td>
<td>Phone number from which payment was made or used in payment form (if available). In international format as: 31703242323. If CID is hidden you will get {PRIVE}</td>
<td>String(50)</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>ConsumerIPAddress</td>
<td>IP address from which the payment form was filled in</td>
<td>String(50)</td>
<td>1.2.3.4</td>
<td>Y</td>
</tr>
<tr>
<td>Amount</td>
<td>The final paid amount value in whole cents</td>
<td>Numeric</td>
<td>550</td>
<td>Y</td>
</tr>
<tr>
<td>Currency</td>
<td>The currency in which the amount is represented</td>
<td>String(3)</td>
<td>EUR</td>
<td>Y</td>
</tr>
<tr>
<td>Duration</td>
<td>The number of seconds dialed. Only available for phone payment methods</td>
<td>Numeric</td>
<td>0</td>
<td>Y</td>
</tr>
<tr>
<td>PaymentMethod</td>
<td>The payment method which was used</td>
<td>String(20)</td>
<td>CREDITCARD</td>
<td>N</td>
</tr>
</tbody>
</table>
### 3.1 CALCULATING THE CHECKSUM

When you receive a Postback Notification, it is imperative that you verify the checksum before processing the postback. If it does not match, then you should always ignore the incoming *Postback Notification*. The checksum is a SHA1 message digest, which can be calculated by concatenating the values below. The values need to be separated by the pipe (|) character.

1. Secret code of your API key
2. MerchantID
3. Status
4. StatusCode
5. OrderID
6. PaymentID
7. Reference
8. TransactionID
9. Amount
10. Currency
11. Duration
12. ConsumerIP Address

**Example**

```
secret|12345|OK|Success|100000007|1234567|My Payment 100000007||10000|EUR|0|143.45.127.31
```
3.2 HANDLING THE POSTBACK NOTIFICATION

This section provides you with guidelines on how your Postback Script should handle incoming Postback Notifications. Paragraph 3.3 describes how you can test your Postback script.

3.2.1 POSSIBLE STATUSES

The Postback Notification contains a parameter called Status. You will most likely want to use this parameter to update the status of your payment in your local database. If you do NOT choose to update the status of your submitted transactions in your own database, this will negatively affect your support options in the instance of refunds, chargebacks or open/not yet validated transaction requests. This must be configured in your account PER WEBSITE at icepay.com. These settings can be found by clicking on the “Advanced” button.

The Status that is returned by ICEPAY can only be one of the following codes:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>The payment is not yet completed.</td>
</tr>
<tr>
<td></td>
<td>After some time you will receive a Postback Notification which contains the OK or ERR status. The time varies depending on the payment method that was used.</td>
</tr>
<tr>
<td>OK</td>
<td>The payment has been completed.</td>
</tr>
<tr>
<td>ERR</td>
<td>The payment was not completed successfully or expired. It cannot change into anything else.</td>
</tr>
<tr>
<td>REFUND</td>
<td>A payment has been successfully refunded.</td>
</tr>
<tr>
<td>CBACK</td>
<td>The consumer has filed a chargeback via their issuing bank. You will receive a different PaymentID parameter but all the other parameters remain the same.</td>
</tr>
<tr>
<td>VALIDATE</td>
<td>The payment is awaiting validation by the consumer by means of a validation code returned by ICEPAY. Currently, this status is only used by SMS payments. You can safely ignore postbacks with this status if you have integrated ICEPAY using the Checkout.aspx method.</td>
</tr>
</tbody>
</table>

You should ignore all other statuses. If a new status is introduced, you will be notified by your account manager.
3.2.2 DETAILED STATUS DESCRIPTION

The Postback Notification also contains a parameter called StatusCode. This is an additional parameter which gives you a more detailed description regarding the status of a payment. Your Postback Script should NOT rely on the content of this parameter to decide what to do as it may change from time to time. It is purely informational. Instead, you should always use the status parameter as described in paragraph 5.1.1.

Examples of StatusCode content:
- Acquirer Error
- Completed with user hangup
- Money received. Bank statement ID: 12345
- Payment aborted by user
- Success

3.2.3 POSSIBLE STATUS TRANSITIONS

If your Postback Script synchronizes the information from Postback Notifications with your local data storage, then you must only do that according to figure 1.

![Possible payment status transitions](image)

Figure 1. Possible payment status transitions

If your transaction is already flagged as OK, it will NEVER transition into ERR. Should you get an incoming Postback Notification, then you must simply ignore it.

3.2.4 LOGGING

You should log ALL incoming Postback Notifications. Support tickets regarding bugs or the status of transactions cannot be handled without this information from the backend of the website.
3.3 TESTING YOUR POSTBACK SCRIPT

You can easily test your Postback script when you log into your ICEPAY account.

Please navigate to the Tools page where you will see a list of test transactions (please see figure 1). These test transactions are created using any version of the Checkout web methods while your Merchant API key is still in Test Mode. Click on a test transaction to change the status. A Postback notification will be sent to your Postback script.

⚠️ You cannot use the tool for live transactions!

![Figure 1](image-url)
4 REPORTING WEB SERVICE

This chapter explains the Reporting Web Service. This web service consists of a set of web methods which allows you to develop applications that can query reporting data from your ICEPAY account. Say bye-bye to your screen scraping applications.

4.1 ALLOW REPORTING ACCESS

In order for your application to access the ICEPAY account information, the option "Allow Reporting Access" must be enabled PER WEBSITE in the ICEPAY account. This is only possible for merchant IDs that are NOT in test mode. A separate Reporting Pin Code consisting of 8-digits will then be created. The Reporting Pin Code is needed to sign all the Reporting Web Service web method requests.

4.2 HOW DOES IT WORK?

Of course, you might be wondering how your application should communicate with this web service. Well, first you will need to invoke the CreateSession web method. If you have successfully created a session, you will receive a SessionID. The SessionID is required for all the other web methods that your application might use. The SessionID is valid for one hour only. If it is expired, your application should automatically create a new session.

Please check the following visualization of the procedure:
Application → Has SessionID? → CreateSession

- Returns the SessionID if successful -

Has SessionID?

No → CreateSession

Yes → Web method, e.g. MonthlyTurnoverTotals

Invalid SessionID?

Yes → Return results

No
4.3 CREATESESSION

The CreateSession method is very important. You will need to create a new session before you are able to use the other methods. If you fail to create a session for 5 times, your ICEPAY account will be blocked. If you fail for 20 times, your IP will be banned and you will not be able to use this web service, or log into the ICEPAY website using the regular credentials at all.

If you get the exception ERR_1000: Session already created for user then it means that you have already started a session. You must always reuse the SessionID that you have created earlier using this method.

When calling other web methods which rely on the SessionID as an input parameter, ICEPAY will check the SessionID-IP-UserAgent combination to see if your application has access to the session.

4.3.1 INPUT

The following parameters are expected for the CreateSession method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>yyyy-mm-ddThh:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>Username</td>
<td>This is your ICEPAY username</td>
<td>String</td>
</tr>
<tr>
<td>UserAgent</td>
<td>This is a unique string that identifies your application. We strongly</td>
<td>String(15..50)</td>
</tr>
<tr>
<td></td>
<td>recommend you to include the name of your application, version, as well as</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a unique string from the machine that is running the application.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The minimum character length is 15. The maximum is 50.</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the request.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|Username|ReportingPinCode|UserAgent
```

Example

```
2017-06-09T01:30:00Z|icepay|12345678|ICEPAY Mobile App 1.0 +31612345678
```
### 4.3.2 RESPONSE

You will get a `CreateSessionResponse` object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timestamp</strong></td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td><strong>Success</strong></td>
<td>Indicates whether the session was created or not</td>
<td>Boolean</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A short description which tells you why it failed. Your script should never rely on this description as it might change.</td>
<td>String</td>
</tr>
<tr>
<td><strong>SessionID</strong></td>
<td>The session ID which you must use for all the other session ID.</td>
<td>String</td>
</tr>
<tr>
<td><strong>Checksum</strong></td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|Success|Description
```

**Example**

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|true|LoginSuccess
```
4.4 KILLSSESSION

The *KillSession* web method destroys your session. It is recommended that you invoke this method once your script or application has finished processing and does not need the session anymore.

### 4.4.1 INPUT

The following parameters are expected for the *KillSession* method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timestamp</strong></td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td><strong>SessionID</strong></td>
<td>This is the session ID created by the <em>CreateSession</em> method</td>
<td>String</td>
</tr>
<tr>
<td><strong>UserAgent</strong></td>
<td>This is a unique string that identifies your application. We strongly recommend you to include the name of your application, version, as well as a unique string from the machine that is running the application. The minimum character length is 15. The maximum is 50.</td>
<td>String(15..50)</td>
</tr>
<tr>
<td><strong>Checksum</strong></td>
<td>This is the SHA1 message digest to verify the authenticity of the request.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent
```

**Example**

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0
```
### 4.4.2 RESPONSE

You will get a `KillSessionResponse` object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td><code>yyyy-mm-ddThh:mm:ssZ</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: <code>2017-06-09T01:30:00Z</code></td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>The session ID which you must use for all the other session ID.</td>
<td>String</td>
</tr>
<tr>
<td>Success</td>
<td>Indicates whether the session was destroyed or not.</td>
<td>Boolean</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent|Success
```

**Example**

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|true
```
4.5 MONTHLYTURNOVERTOTALS

The *MonthlyTurnoverTotals* web method returns the sum of the turnover of all the transactions according to the provided criteria: month, year and currency.

Specifying the currency does not mean that the Web Service will convert the turnover to that currency. You are actually saying that you want to aggregate transactions that were done in that currency.

4.5.1 INPUT

The following input parameters are required for this method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SessionID</td>
<td>This is the session ID created by the CreateSession method</td>
<td>String</td>
</tr>
<tr>
<td>UserAgent</td>
<td>The same user agent that you provided the CreateSession method with</td>
<td>String</td>
</tr>
<tr>
<td>MerchantID</td>
<td>The MerchantID for which you want to see the totals</td>
<td>Integer</td>
</tr>
<tr>
<td>CurrencyCode</td>
<td>Specifies what transactions should be included in the results</td>
<td>String</td>
</tr>
<tr>
<td>Year</td>
<td>The year that you want to query</td>
<td>Integer</td>
</tr>
<tr>
<td>Month</td>
<td>The month that you want to query</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the request.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent|MerchantID|CurrencyCode|Year|Month
```

**Example**

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|12345|EUR|2010|5
```
4.5.2 RESPONSE

You will get a *MonthlyTurnoverTotals* object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>yyyy-mm-ddTh:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>The SessionID that initiated the request</td>
<td>String</td>
</tr>
<tr>
<td>CurrencyCode</td>
<td>Specifies in what currency the results are.</td>
<td>String</td>
</tr>
<tr>
<td>Year</td>
<td>Specifies the year of the results</td>
<td>Integer</td>
</tr>
<tr>
<td>Month</td>
<td>Specifies the month of the results</td>
<td>Integer</td>
</tr>
<tr>
<td>Days</td>
<td>A list of <em>DayStatistics</em> objects</td>
<td>DayStatistics[]</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent|CurrencyCode|Year|Month|Days
```

Where the component Days consists of a series of input messages with the following syntax:

```
Year|Month|Day|Duration|TransactionsCount|Turnover
```

**Example**

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|EUR|2010|5|1|0|100|20000|2010|5|2|0|200|40000|2010|5|3|0|0
```

The web service will never give you back the results of 3 days like in the above example. It will always give you back the exact number of days of that particular month. So for the month May you will get back 31 days. The reason we left it out in the above example is because otherwise the example would become too large.
4.6 GETMERCHANTS

This method allows you to retrieve a list of merchants that belong to your ICEPAY account.

4.6.1 REQUEST PARAMETERS

The following input parameters are required for this method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>This is the session ID created by the CreateSession method</td>
<td>String</td>
</tr>
<tr>
<td>UserAgent</td>
<td>The same user agent that you provided the CreateSession method with</td>
<td>String</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the request.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent
```

**Example**

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0
```
4.6.2 RESPONSE

You will get a GetMerchantsResponse object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>yyyy-mm-ddThh:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>The SessionID that initiated the request</td>
<td>String</td>
</tr>
<tr>
<td>Merchants</td>
<td>A list of Merchant objects. Please see 0 for the structure of a single Merchant object.</td>
<td>Merchant[]</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent(Merchants)
```

Where the component Merchants consists of a series of input messages with the following syntax:

```
MerchantID|Description|TestMode
```

Example

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|10000|My test website|true|10001|My production website|false
```

4.6.2.1 MERCHANT

This object contains information about a merchant.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>The merchant ID</td>
<td>Integer</td>
</tr>
<tr>
<td>Description</td>
<td>Name of the merchant (as entered by the merchant in the ICEPAY back end)</td>
<td>String</td>
</tr>
<tr>
<td>TestMode</td>
<td>Indicates whether the merchant record is in Test Mode.</td>
<td>Boolean</td>
</tr>
</tbody>
</table>
It is recommended that you cache your results

4.7 GETPAYMENTMETHODS

Use the GetPaymentMethods web method to retrieve a list of all supported payment methods by ICEPAY.

Since this list will DEFINITELY not be updated on a daily basis, we recommend that your application caches the response that you get from this method for as long as possible in order to avoid unnecessary communication with the web service.

4.7.1 INPUT

The following input parameters are required for this method:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>yyyy-mm-ddThh:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>This is the session ID created by the CreateSession method</td>
<td>String</td>
</tr>
<tr>
<td>UserAgent</td>
<td>The same user agent that you provided the CreateSession method with</td>
<td>String</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the request.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent
```

Example

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0
```
4.7.1.1 RESPONSE

You will get a `GetPaymentMethodsResponse` object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddTh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>The SessionID that initiated the request</td>
<td>String</td>
</tr>
<tr>
<td>PaymentMethods</td>
<td>A list of PaymentMethod objects. Please see 0 for the structure of a single PaymentMethod object.</td>
<td>PaymentMethod[]</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Timestamp|SessionID|ReportingPinCode|UserAgent(PaymentMethods)
```

Where the component `PaymentMethods` consists of a series of input messages with the following syntax:

```
Description|PaymentMethodCode
```

Example:

```
2017-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|iDEAL|iDEAL|Credit card|CREDITCARD|Wire transfer|WIRE
```

4.7.1.2 PAYMENTMETHOD OBJECT

This object contains information about a payment method.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A description of the payment method</td>
<td>String</td>
</tr>
<tr>
<td>PaymentMethodCode</td>
<td>This is the payment method code of a payment method</td>
<td>String</td>
</tr>
</tbody>
</table>
4.8 SEARCHPAYMENTS

Use the SearchPayments web method to search for payments linked to your ICEPAY account. There are several filters which you can employ for a more detailed search.

ICEPAY will return a maximum of 25 records for each search request. You can use the parameter Page to browse through the results.

4.8.1 INPUT

The required input parameters are displayed below. If you do not want to include the optional parameters, then please set the parameter to the value NULL (or the equivalent of this value in your programming language).

Please do not use an empty value as the optional value.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddTh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>This is the session ID created by the CreateSession method</td>
<td>String</td>
</tr>
<tr>
<td>UserAgent</td>
<td>The same user agent that you provided the CreateSession method with</td>
<td>String</td>
</tr>
<tr>
<td>MerchantID</td>
<td>Filter on the MerchantID</td>
<td>Integer</td>
</tr>
<tr>
<td>PaymentID</td>
<td>Filter on the PaymentID</td>
<td>Integer</td>
</tr>
<tr>
<td>Status</td>
<td>Filter on the status of payments, e.g. OK, OPEN, ERR.</td>
<td>String</td>
</tr>
<tr>
<td>OrderTime1</td>
<td>Filter on the creation time of the payment. This parameter will be used for the start range of the filter.</td>
<td>String</td>
</tr>
<tr>
<td>OrderTime2</td>
<td>Filter on the creation date of the payment. This parameter will be used for the end range of the filter.</td>
<td>String</td>
</tr>
<tr>
<td>PaymentTime1</td>
<td>Filter on the time when the payment was completed. This parameter will be used for the start range of the filter.</td>
<td>String</td>
</tr>
<tr>
<td>PaymentTime2</td>
<td>Filter on the time when the payment was completed. This parameter will be used for the end range of the filter.</td>
<td>String</td>
</tr>
</tbody>
</table>
Please check the following page for the remaining parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountryCode</td>
<td>The requested country code</td>
<td>String (optional)</td>
</tr>
<tr>
<td>CurrencyCode</td>
<td>The requested currency code</td>
<td>String (optional)</td>
</tr>
<tr>
<td>Amount</td>
<td>The amount in cents</td>
<td>Integer (optional)</td>
</tr>
<tr>
<td>PaymentMethod</td>
<td>Filter on the payment method, e.g. IDEAL, WIRE</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerAccountNumber</td>
<td>Search for account number</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerName</td>
<td>Search for consumer name</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerAddress</td>
<td>Search for the address</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerHouseNumber</td>
<td>Search for the house number</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerPostCode</td>
<td>Search for the postal code</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerCity</td>
<td>Search for the city</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerCountry</td>
<td>Search for the country</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerEmail</td>
<td>Search for the e-mail</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerPhoneNumber</td>
<td>Search for the phone number</td>
<td>String (optional)</td>
</tr>
<tr>
<td>ConsumerIPAddress</td>
<td>Search for the IP address</td>
<td>String (optional)</td>
</tr>
<tr>
<td>Page</td>
<td>The page index of the results</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the request.</td>
<td>String</td>
</tr>
</tbody>
</table>

* The date time will be using the time zone settings of the user account.

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:
4.8.2 RESPONSE

You will get a `SearchPaymentsResponse` object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>SessionID</td>
<td>The SessionID that initiated the request</td>
<td>String</td>
</tr>
<tr>
<td>Payments</td>
<td>A list of <code>Payment</code> objects</td>
<td>Payment[]</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```plaintext
Timestamp|SessionID|ReportingPinCode|UserAgent(Payments)
```

Where the component `Payments` consists of a series of input messages with the following syntax:

```plaintext
```
Example

| 2179-06-09T01:30:00Z|ABCDEFGH|JKLMNOPQRSTUVWXYZ|12345678|My Reporting App
| 1.0|100||John Doe||NL|EUR|0|12345|2017-06-09T01:30:00Z|10000000000|IDEAL|2017-06-09T01:30:00Z|OK|Success|false |

5 REFUND WEB SERVICE

This chapter covers the Refund Web Service, which allows you to perform various web methods related to payment refunding.

In order to use the Refund Web Service, you will need to explicitly grant your website access to this web service. You can do this by logging into your ICEPAY account.

Before you actually start implementing the Refund Web Service, there are certain things that you should know.

5.1 REFUNDING QUEUE

When you initiate a refund request, it will be put in the refunding queue. This queue will be processed once per day at 8am UTC+0. Once processed, it does not mean that the consumer will receive its money immediately. It only means that we have instructed the bank or payment method provider to perform the refund. It may take up to a few workdays before it is actually processed.

5.2 NOTIFICATION

You will be notified of successful refunds via either e-mail or the ICEPAY postback system. The type of feedback depends on the settings you set for your merchant record. You can configure these settings by logging into your ICEPAY account. If you receive postbacks, there is absolutely no need for you to implement a polling system.

5.3 SUPPORTED PAYMENT METHODS

Not all payment methods support refunding. Refunding is limited to:

- iDEAL
- Credit Card
- SEPA Direct Debit
- PayPal
- Giropay
- SOFORT
- Bancontact
Wire transfer  
AchterafBetalen

The below payment methods require IBAN and BIC to be processed:

- iDEAL
- SEPA Direct Debit
- Giropay
- SOFORT
- Bancontact
- Wire transfer

It takes ICEPAY a few working days to retrieve the IBAN and BIC (due to bank processing times).

### 5.4 TEST AND LIVE TRANSACTIONS

Currently, all refund web methods accept live transactions **ONLY**.

### 5.5 REQUESTREFUND

The *RequestRefund* web method allows you to initiate a refund request for a payment. Refunds for payment methods that are executed by ICEPAY, like Direct Debit, may only be initiated for payments that have successfully been collected from the shopper. Refund requests made before successful collection might result in a deduction from outpayments.

**Input**

The required input parameters are displayed below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest of all the members.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>yyyymm-ddThh:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: 2017-06-09T01:30:00Z</td>
<td></td>
</tr>
<tr>
<td>PaymentID</td>
<td>The ID of the payment.</td>
<td>Integer</td>
</tr>
<tr>
<td>RefundAmount</td>
<td>The amount to be refunded (in cents)</td>
<td>Integer</td>
</tr>
<tr>
<td>RefundCurrency</td>
<td>The currency of the refund. <strong>Remarks:</strong> This currency must match the currency of the payment.</td>
<td>String</td>
</tr>
</tbody>
</table>
The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Secret|MerchantID|Timestamp|PaymentID|RefundAmount|RefundCurrency
```

Example

```
secret|12345|2017-06-09T01:30:00Z|1234567|1000|EUR
```

5.5.1 RESPONSE

You will get a `RequestRefundResponse` object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ</td>
<td>String</td>
</tr>
<tr>
<td>RefundID</td>
<td>This is the ID of the requested refund. If possible, it is recommended to store this value in your system as you may decide (at a later stage), to cancel the refund request.</td>
<td>Integer</td>
</tr>
<tr>
<td>PaymentID</td>
<td>This is the payment for which you requested a refund.</td>
<td>Integer</td>
</tr>
<tr>
<td>RefundAmount</td>
<td>The requested refund amount specified in the request</td>
<td>Integer</td>
</tr>
<tr>
<td>RemainingRefundAmount</td>
<td>The remaining amount that you can still request a refund for.</td>
<td>Integer</td>
</tr>
<tr>
<td>RefundCurrency</td>
<td>The requested refund currency specified in the request</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Secret|MerchantID|Timestamp|RefundID|PaymentID|RefundAmount|RemainingRefundAmount|RefundCurrency
```
5.6 CANCELREFUND

The *CancelRefund* web method allows you to cancel a refund request if it has not already been processed.

### 5.6.1 INPUT

The required input parameters are displayed below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest of all the members.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td><code>yyyy-mm-ddThh:mm:ssZ</code></td>
<td></td>
</tr>
<tr>
<td>RefundID</td>
<td>This is the RefundID that is returned by the RequestRefund web method upon a successful invocation.</td>
<td>Integer</td>
</tr>
<tr>
<td>PaymentID</td>
<td>This is the PaymentID of the transaction for which you requested the refund in the first place</td>
<td>Integer</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```plaintext
secret|MerchantID|Timestamp|RefundID|PaymentID
```

**Example**

```
secret|12345|2017-06-09T01:30:00Z|12345|1234567|1000|0|EUR
```

**5.6.1 INPUT**

The required input parameters are displayed below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest of all the members.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td><code>yyyy-mm-ddThh:mm:ssZ</code></td>
<td></td>
</tr>
<tr>
<td>RefundID</td>
<td>This is the RefundID that is returned by the RequestRefund web method upon a successful invocation.</td>
<td>Integer</td>
</tr>
<tr>
<td>PaymentID</td>
<td>This is the PaymentID of the transaction for which you requested the refund in the first place</td>
<td>Integer</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
secret|MerchantID|Timestamp|RefundID|PaymentID
```

**Example**

```
secret|12345|2017-06-09T01:30:00Z|12345|1234567
```
5.6.2 RESPONSE

You will get a `CancelRefundResponse` object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 message digest to verify the authenticity of the response.</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>This is the current UTC time that must have the following format:</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td><code>yyyy-mm-ddThh:mm:ssZ</code></td>
<td></td>
</tr>
<tr>
<td>Success</td>
<td>This field will contain the value ‘Y’ if the refund request was cancelled successfully.</td>
<td>String</td>
</tr>
</tbody>
</table>

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Secret|MerchantID|Timestamp|Success
```

**Example**

```
secret|12345|2017-06-09T01:30:00Z|false
```

5.7 GETPAYMENTREFUNDS

The `GetPaymentRefunds` web method allows you to query refund request information that belongs to the payment.

5.7.1 INPUT

The required input parameters are displayed below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
</table>


### MerchantID
This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.  
**Data type:** Integer

### Checksum
This is the SHA1 digest of all the members.  
**Data type:** String

### Timestamp
This is the current UTC time that must have the following format:  
`yyyy-mm-ddThh:mm:ssZ`  
Example: 2017-06-09T01:30:00Z  
**Data type:** String

### PaymentID
This is the PaymentID of the transaction for which you requested the refund in the first place  
**Data type:** Integer

The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Secret|MerchantID|Timestamp|PaymentID
```

#### Example

```
secret|12345|2017-06-09T01:30:00Z|12345|1234567
```

## 5.7.2 RESPONSE

You will get a `GetPaymentRefundsResponse` object as the response, which consists of the following members:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MerchantID</td>
<td>This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.</td>
<td>Integer</td>
</tr>
<tr>
<td>Checksum</td>
<td>This is the SHA1 digest of all the members.</td>
<td>String</td>
</tr>
</tbody>
</table>
| Timestamp | This is the current UTC time that must have the following format:  
`yyyy-mm-ddThh:mm:ssZ`  
Example: 2017-06-09T01:30:00Z | String |
| Refunds | A collection of Refund objects. If you have done several partial refunds, then this collection contains several objects. | Refund[] |
The checksum is calculated by creating a SHA1 message digest using an input message with the following syntax:

```
Secret|MerchantID|Timestamp|PaymentID(Refunds)
```

Where the component **Refunds** consists of a series of input messages with the following syntax:

```
RefundID|DateCreated|RefundAmount|RefundCurrency|Status
```

**Example**

```
secret|12345|2017-06-09T01:30:00Z|1234567|12345|2009-06-09T01:30:00Z|12345|2017-06-09T01:30:00Z|1200|EUR|PENDING|12350|2017-06-09T01:31:00Z|1200|EUR|PROCESSING
```

### 5.7.2.1 REFUND OBJECT

This object contains information about a payment method.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefundID</td>
<td>This is a unique ID that is assigned to your refund request</td>
<td>Integer</td>
</tr>
<tr>
<td>RefundAmount</td>
<td>This is the amount of the refund</td>
<td>Integer</td>
</tr>
<tr>
<td>RefundCurrency</td>
<td>This is the currency of the refund</td>
<td>String</td>
</tr>
<tr>
<td>DateCreated</td>
<td>This value indicates when the refund request was created (in UTC+0):</td>
<td>String</td>
</tr>
<tr>
<td>Status</td>
<td>A refund can have one of the following status:</td>
<td>String</td>
</tr>
<tr>
<td>PENDING</td>
<td>Refund is placed in the queue</td>
<td></td>
</tr>
<tr>
<td>PROCESSING</td>
<td>Refund sent to financial institution for refund</td>
<td></td>
</tr>
<tr>
<td>COMPLETED</td>
<td>Refund processed by financial institution</td>
<td></td>
</tr>
<tr>
<td>REFUSED</td>
<td>Refund cannot be processed</td>
<td></td>
</tr>
</tbody>
</table>

Example: 2017-06-09T01:30:00Z
6 EXCEPTIONS

6.1 ERROR CODES 0000 – 0016

The following error codes are generic error codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR_0000</td>
<td>Internal server error: XXX An unexpected error occurred. Please contact <a href="mailto:support@icepay.com">support@icepay.com</a> with the full merchant web shop database log to resolve.</td>
</tr>
<tr>
<td>ERR_0001</td>
<td>Request is missing You did not provide a request object with the web method</td>
</tr>
<tr>
<td>ERR_0002</td>
<td>Please provide a valid 'MerchantID' member You must provide a valid numeric 'MerchantID' member</td>
</tr>
<tr>
<td>ERR_0003</td>
<td>Please provide the 'XXX' member You forgot to include a member in your web method request object. Where XXX is the name of the member that you forgot to include.</td>
</tr>
<tr>
<td>ERR_0004</td>
<td>Please provide the IP address of your end-user You forgot to include the IP address of the end-user.</td>
</tr>
<tr>
<td>ERR_0005</td>
<td>Merchant 'XXXXX' is disabled Your API key is disabled. Please contact your account manager regarding this issue.</td>
</tr>
<tr>
<td>ERR_0006</td>
<td>Merchant 'XXXXX' was not found Unknown merchant ID</td>
</tr>
<tr>
<td>ERR_0007</td>
<td>Checksum for 'XXX' is invalid The provided checksum did not match. Where XXX is the name of the web method for which the checksum failed. Please make sure the hash of the checksum is in lowercase.</td>
</tr>
<tr>
<td>ERR_0008</td>
<td>You can only invoke this method using the 'XXX' payment method This exception means that the web method can only be used in conjunction with the XXX payment method.</td>
</tr>
<tr>
<td>ERR_0009</td>
<td>Payment with ID 'XXX' not found</td>
</tr>
<tr>
<td>ERR_0010</td>
<td>'XXX' parameter must be at least YY characters The length of the provided parameter must be at least YY characters long.</td>
</tr>
<tr>
<td>ERR_0011</td>
<td>'XXX' parameter may not exceed YY characters The length of the parameter has exceeded the maximum allowed length YY.</td>
</tr>
<tr>
<td>ERR_0012</td>
<td>'XXX' is an invalid payment method</td>
</tr>
<tr>
<td>ERR_0013</td>
<td>Invalid date: X The provided date is in an invalid format. Please provide a string in the following format: YYYY-MM-DD or YYYY-MM-DD HH:MM</td>
</tr>
</tbody>
</table>
ERR_0014  Invalid date period
Please provide a valid month and year combination.

ERR_0015  The provided country code ‘XX’ is invalid

ERR_0016  Payment does not belong to specified merchant
You (accidentally) specified a payment ID that does not belong to the specified merchant.

6.2 ERROR CODES 1000 – 1003

The following error codes are specific for the Reporting Web Service:

ERR_1000  Session already created for user
This means that you have already invoked the CreateSession web method from that IP address.

ERR_1001  Account does not allow API access
The ICEPAY account does not allow API access

ERR_1002  Invalid SessionID
The SessionID that you provided is invalid or expired

ERR_1003  Invalid username


The following error codes are specific for the Refund Web Service:

ERR_2000  Merchant not granted to use Refund Web Service
The merchant is not granted access to use the Refund Web Service. In order to enable the refund web service for your merchant, please log into your ICEPAY account to configure your merchants.

ERR_2001  Invalid RefundID ‘XXX’
The RefundID that you provided is invalid. Please check the refund ID.

ERR_2002  Amount to refund exceeds remaining balance
You were trying to initiate a refund request with an amount that is larger than the requested amount

ERR_2003  Payment method is not supported by the Refund Web Service
The payment method used for the provided payment does not support refund possibilities

ERR_2004  Invalid RefundAmount
Please make sure that:
- The amount is in cents
- The amount is positive

ERR_2005  Refund currency does not match payment currency
<table>
<thead>
<tr>
<th>ERR_2006</th>
<th>You can only refund payments with the status 'OK'</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR_2007</td>
<td>RefundID does not belong to PaymentID</td>
</tr>
</tbody>
</table>
APPENDIX A: FLOW OF THE BASIC INTEGRATION APPROACH USING CHECKOUT WEB METHOD

1. End-user wants to purchase a product

2. API method: Checkout
   CheckoutRequest object

3. Initialize payment

4. Response

5. CheckoutResponse object

6. Redirect user to payment page

7. Failed/Successful transaction

8. Payment status

9. Postback

10. Redirect to success/error page

End-user
APPENDIX B: GENERIC EXTENDEDCHECKOUT XML

Note that all fields in the XSD that do not have the minOccurs="0" attribute are mandatory and must be provided on checkout.

The following table explains XML nodes and attributes that are not fully self-explanatory from the XSD.

<table>
<thead>
<tr>
<th>XML node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order/Products</td>
<td>This node contains information about the ordered products. In case of AfterPay, they will appear as lines on the invoice.</td>
</tr>
<tr>
<td>Order/Products/Product/ProductID</td>
<td>This is the ID of your product.</td>
</tr>
<tr>
<td>Order/Products/Product/UnitPrice</td>
<td>The amount (in cents) of the product including VAT.</td>
</tr>
<tr>
<td>Order/Products/Product/VATCategory</td>
<td>A valid VAT category needs to be populated. The VAT rate will be determined automatically.</td>
</tr>
<tr>
<td></td>
<td><strong>standard</strong> = This is the standard (high) tariff, e.g. 21% in The Netherlands, 21% in Belgium, etc.</td>
</tr>
<tr>
<td></td>
<td><strong>reduced-middle</strong> = This is the middle VAT tariff in case of three different tariffs. Not applicable in The Netherlands, 12% in Belgium, etc.</td>
</tr>
<tr>
<td></td>
<td><strong>reduced-low</strong> = This is the lowest VAT tariff, e.g. 6% in The Netherlands, 6% in Belgium, etc.</td>
</tr>
<tr>
<td></td>
<td><strong>zero</strong> = 0% VAT</td>
</tr>
<tr>
<td></td>
<td><strong>exempt</strong> = Use this when the product is exempted from VAT, e.g. second hand books (in The Netherlands).</td>
</tr>
<tr>
<td>Order/Products/Product/Description</td>
<td>A description of the product. AfterPay accepts a maximum of 45 characters. Diacritics, etc. will be filtered by AfterPay automatically.</td>
</tr>
<tr>
<td>Order/Addresses/Address/ZipCode</td>
<td>This value will be validated. Make sure the format is correct for the specified country:</td>
</tr>
</tbody>
</table>
|                                        | Belgium: NNNN  
Germany: NNNNN  
The Netherlands: NNNNLL                                                                                                                               |
|                                        | Where N is a number, and L is a letter.                                                                                                                                                                |
| Order/Addresses/Address[id=billing]/CountryCode | The country code of the consumer receiving the billing. This must be the same as the country code provided in the CheckoutExtendedRequest object.                                      |

8.1 XSD

```xml
<?xml version="1.0" encoding="utf-8"?>
```
<xs:schema id="Order" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Order">
    <xs:complexType>
      <xs:sequence>
        <!-- Consumer information -->
        <xs:element name="Consumer">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ConsumerID" type="xs:string" minOccurs="0" maxOccurs="1"/>
              <xs:element name="Email">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="[A-Za-z0-9_-]+([-.'][A-Za-z0-9_-]+)*@[A-Za-z0-9_-]+([-.][A-Za-z0-9_-]+)*\.[A-Za-z0-9_-]+([-.][A-Za-z0-9_-]+)*"/>
                </xs:restriction>
              </xs:element>
              <xs:element name="Phone">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:minLength value="1"/>
                  </xs:restriction>
                </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <!-- Addresses -->
        <xs:element name="Addresses">
          <xs:complexType>
            <xs:sequence minOccurs="2" maxOccurs="2">
              <xs:element name="Address">
                <xs:complexType>
                  <xs:sequence>
                    <!-- APB2C: referencePerson.initials -->
                    <xs:element name="Initials">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:minLength value="1"/>
                        </xs:restriction>
                      </xs:complexType>
                    </xs:element>
                    <!-- APB2C: referencePerson.tussenvoegsel -->
                    <xs:element name="Prefix" type="xs:string"/>
                    <!-- APB2C: referencePerson.lastname -->
                    <xs:element name="LastName">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:minLength value="1"/>
                        </xs:restriction>
                      </xs:complexType>
                    </xs:element>
                </xs:sequence>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
</xs:simpleType>
</xs:element>
</!-- APB2C: B2CAddress.Streetname -->
<xs:element name="Street" type="xs:string" />
</!-- APB2C: B2CAddress.Housenumber -->
<xs:element name="HouseNumber">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
</!-- APB2C: B2CAddress.Housenumberaddition -->
<xs:element name="HouseNumberAddition" type="xs:string" />
</!-- APB2C: BC2Address.Postalcode -->
<xs:element name="ZipCode">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
</!-- APB2C: BC2Address.city -->
<xs:element name="City">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
</!-- APB2C: BC2Address.isoCountryCode -->
<xs:element name="Country" type="xs:string" />
</xs:sequence>
<xs:attribute name="id" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="billing|shipping" />
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:complexType>
<xs:unique name="id">
  <xs:selector xpath="Address"/>
  <xs:field xpath="@id"/>
</xs:unique>
</xs:element>

</!-- Products -->
<xs:element name="Products">
  <xs:complexType>
8.2 SAMPLE XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Order>
  <Consumer>
    <ConsumerID>98680</ConsumerID>
    <Email>mrsomebody@me.com</Email>
    <Phone>+4912345678</Phone>
  </Consumer>
  <Addresses>
    <Address id="shipping">
      <Initials>Dhr</Initials>
      <LastName>Somebody</LastName>
      <Street>Mystreet</Street>
      <HouseNumber>20</HouseNumber>
      <HouseNumberAddition/>
      <ZipCode>12345</ZipCode>
      <City>MyCity</City>
      <Country>DE</Country>
    </Address>
    <Address id="billing">
      <Initials>Dhr</Initials>
      <LastName>Somebody</LastName>
    </Address>
  </Addresses>
</Order>
```
<Street>Mystreet</Street>
<HouseNumber>28</HouseNumber>
<HouseNumberAddition/>
<ZipCode>12345</ZipCode>
<City>MyCity</City>
<Country>DE</Country>
</Address>
</Addresses>
<Products>
<Product>
<ProductID>1</ProductID>
<ProductName>myshop.de</ProductName>
<Description>myProduct</Description>
<Quantity>1</Quantity>
<UnitPrice>22940</UnitPrice>
<VATCategory>standard</VATCategory>
</Product>
<Product>
<ProductID>01</ProductID>
<ProductName>Shipping Costs</ProductName>
<Description/>
<Quantity>1</Quantity>
<UnitPrice>0</UnitPrice>
<VATCategory>exempt</VATCategory>
</Product>
</Products>
</Order>

9 APPENDIX C: ACHTERAFBETALEN CHECKOUT EXTENDED XML

Note that all fields in the XSD that do not have the minOccurs="0" attribute are mandatory and must be provided on checkout.

The following table explains XML nodes and attributes that are not fully self-explanatory from the XSD.

<table>
<thead>
<tr>
<th>XML node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer/prefix</td>
<td>The prefix, tussenvoegsel in dutch, must be used for names like 'De Jong' and 'Van der Wiel'. The lastName field must not contain the prefix.</td>
</tr>
<tr>
<td>customer/birthDate</td>
<td>Dateformat is YYYY-MM-DD</td>
</tr>
<tr>
<td>customer/gender</td>
<td>Gender must be 'M' (male), 'F' (female) or 'unknown'</td>
</tr>
<tr>
<td>invoiceAddress/country</td>
<td>Supported countrycodes are NL, BE, FR and DE</td>
</tr>
<tr>
<td>invoiceAddress/zipCode</td>
<td>This value will be validated. Make sure the format is correct for the specified country:</td>
</tr>
</tbody>
</table>
Belgium: NNNN  
Germany: NNNNN  
The Netherlands: NNNNLL  

Where N is a number, and L is a letter.

| order/amount | The total amount of the order, in cents. Must be the same as the amount in the CheckoutRequest object. The amount must include VAT and must be equal to shippingCost+Total for all orderDetails, where the amount for an orderDetail is quantity * unitPrice |
| order/amountVAT | The VAT part of amount |
| order/shippingCost | The shipping cost for the order (incl VAT) |
| order/shippingCostVAT | The VAT part of the shippingCost |
| order/orderDetail | You must provide an orderDetail record for each article that will be delivered |
| order/orderDetail/productCode | Code of the article delivered |
| order/orderDetail/productName | Name of the article delivered |
| order/orderDetail/productDescription | Optional additional description of the article delivered |
| order/orderDetail/quantity | The quantity of the article. The total amount of the orderDetail will be calculated by doing quantity * unitPrice |
| order/orderDetail/unitPrice | The price per unit of the article, in whole cents (incl VAT) |
| order/orderDetail/unitPriceVAT | The VAT part of the unitPrice |
| order/orderDetail/vatCode | Code for the VAT that applies to the article. Must be: 1000 (High), 1001 (Low), 1002 (None), 1003 (VAT 0%), 1004 (Reverse charged) or 1005 (Margin scheme) |

The XSD consists of three separate XSD’s: AchterafBetalen.xsd, which references ComplexTypes.xsd and SimpleTypes.xsd.

### 9.1 ACHTERAFBETALEN.XSD

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>
      Copyright (C) 2017 ICEPAY B.V.
    </xs:documentation>
  </xs:annotation>

  <!-- includes -->
  <xs:include schemaLocation="SimpleTypes.xsd" />
  <xs:include schemaLocation="ComplexTypes.xsd" />
</xs:schema>
```
<!-- define the actual Xsd -->
<xs:element name="achterafBetalen">
  <xs:complexType>
    <xs:sequence>
      <!-- Customer data -->
      <xs:element name="customer" type="Personal" />

      <!-- Customer bank data -->
      <xs:element name="iban" type="IBAN" />

      <!-- Invoice address -->
      <xs:element name="invoiceAddress" type="Address" />

      <!-- Shipping address -->
      <xs:element name="shippingAddress" type="Address" />

      <!-- Order -->
      <xs:element name="order" type="Order" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<!-- Complex types -->
<xs:complexType name="Personal">
  <xs:sequence>
    <xs:element name="initials" type="xs:string" />
    <xs:element name="prefix" type="xs:string" minOccurs="0" />
    <xs:element name="lastName" type="xs:string" />
    <xs:element name="birthDate" type="xs:date" />
    <xs:element name="gender" type="Gender" />
    <xs:element name="phoneNumber" type="xs:string" />
    <xs:element name="email" type="EmailAddress" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="Order">
  <xs:sequence>
    <xs:element name="amount" type="xs:int" />
    <xs:element name="amountVAT" type="xs:int" />
    <xs:element name="shippingCost" type="xs:int" />
    <xs:element name="shippingCostVAT" type="xs:int" />
    <xs:element name="orderDetail" type="OrderDetail" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="OrderDetail">
  <xs:sequence>
    <xs:element name="productCode" type="xs:string" />
    <xs:element name="productName" type="xs:string" />
    <xs:element name="productDescription" type="xs:string" minOccurs="0" />
    <xs:element name="quantity" type="NullOrInteger" />
  </xs:sequence>
</xs:complexType>
<xs:element name="vatCode" type="VATCode"/>
<xs:element name="unitPrice" type="xs:int"/>
<xs:element name="unitPriceVAT" type="xs:int"/>
</xs:sequence>
</xs:complexType>

<xs:simpleType name="VATCode">
<xs:restriction base="xs:string">
<!-- High -->
<xs:enumeration value="1000"/>
<!-- Low -->
<xs:enumeration value="1001"/>
<!-- None -->
<xs:enumeration value="1002"/>
<!-- VAT 0% -->
<xs:enumeration value="1003"/>
<!-- Reverse charged -->
<xs:enumeration value="1004"/>
<!-- Margin scheme -->
<xs:enumeration value="1005"/>
</xs:restriction>
</xs:simpleType>
</xs:schema>

9.2 SIMPLETYPES.XSD

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>
      Copyright (C) 2017 ICEPAY B.V.
    </xs:documentation>
  </xs:annotation>

  <!-- define the simple types -->
  <xs:simpleType name="Gender">
    <xs:annotation>
      <xs:documentation>
        Gender. Possible values: M(ale), F(eemale), unknown
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="M" />
      <xs:enumeration value="F" />
      <xs:enumeration value="unknown" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="EmailAddress">
    <xs:annotation>

February 6, 2018 – Version 1.7.0  ICEPAY WEBSERVICE – Implementation Guide
<xs:maxLength value="100"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max200Text">
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="200"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max250Text">
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="250"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max300Text">
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="300"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max4NumericText">
<xs:restriction base="xs:string">
<xs:pattern value="[0-9]{1,4}"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max5NumericText">
<xs:restriction base="xs:string">
<xs:pattern value="[0-9]{1,5}"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max10NumericText">
<xs:restriction base="xs:string">
<xs:pattern value="[0-9]{1,18}"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max18NumericText">
<xs:restriction base="xs:string">
<xs:pattern value="[0-9]{1,18}"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="Max50NumericText">
<xs:restriction base="xs:string">
<xs:pattern value="[0-9]{1,4}"/>
</xs:restriction>
</xs:simpleType>
9.3 COMPLEXTYPES.XSD

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>
      Copyright (C) 2017 ICEPAY B.V.
    </xs:documentation>
  </xs:annotation>
  <!-- includes -->
  <xs:include schemaLocation="SimpleTypes.xsd" />

  <xs:complexType name="Address">
    <xs:sequence>
      <xs:element name="street" type="xs:string" />
      <xs:element name="houseNumber" type="xs:int" />
      <xs:element name="extension" type="xs:string" minOccurs="0" />
      <xs:element name="zipCode" type="xs:string" />
      <xs:element name="city" type="xs:string" />
      <xs:element name="province" type="xs:string" minOccurs="0" />
      <xs:element name="country" type="CountryCode" default="NL" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>

9.4 SAMPLE XML

<?xml version="1.0" encoding="utf-8"?>
  <customer>
    <initials>T</initials>
    <prefix />
  </customer>
<lastName>Testpersoon</lastName>
<birthDate>1981-02-01</birthDate>
<gender>M</gender>
<phoneNumber>0201234567</phoneNumber>
<email>emailadres@bedrijf.nl</email>
</customer>
<iban>NL41RABO0181593308</iban>
<invoiceAddress>
.street>Straatweg</street>
.houseNumber>123</houseNumber>
.extension>a</extension>
.zipCode>1234AB</zipCode>
<city>Amsterdam</city>
<country>NL</country>
</invoiceAddress>
<shippingAddress>
.street>Straatweg</street>
.houseNumber>123</houseNumber>
.extension>a</extension>
.zipCode>1234AB</zipCode>
<city>Amsterdam</city>
<country>NL</country>
</shippingAddress>
<order>
.amount>18500</amount>
.amountVAT>3885</amountVAT>
.shippingCost>1000</shippingCost>
.shippingCostVAT>210</shippingCostVAT>
<orderDetail>
.productCode>Artikel01</productCode>
.productName>Artikelomschrijving 1</productName>
.productDescription>Zelfmaakmode stap voor stap met alle basiskennis. Deel 1 van 4 nummers om te verzamelen. *** dit artikel kan licht beschadigd zijn! ***</productDescription>
.quantity>2</quantity>
.vatCode>1000</vatCode>
.unitPrice>2500</unitPrice>
.unitPriceVAT>525</unitPriceVAT>
</orderDetail>
<orderDetail>
.productCode>Artikel02</productCode>
.productName>Artikelomschrijving 2</productName>
.productDescription>Zelfmaakmode stap voor stap met alle basiskennis. Deel 2 van 4 nummers om te verzamelen.</productDescription>
.quantity>1</quantity>
.vatCode>1000</vatCode>
.unitPrice>5000</unitPrice>
.unitPriceVAT>1050</unitPriceVAT>
</orderDetail>
<orderDetail>
.productCode>Artikel03</productCode>
.productName>Artikelomschrijving 3</productName>
Zelfmaakmode stap voor stap met alle basiskennis. Deel 3 van 4 nummers om te verzamelen.

```xml
<productDescription>Zelfmaakmode stap voor stap met alle basiskennis. Deel 3 van 4 nummers om te verzamelen.</productDescription>
<quantity>1</quantity>
<vatCode>1000</vatCode>
<unitPrice>7500</unitPrice>
<unitPriceVAT>1575</unitPriceVAT>
</orderDetail>
</order>
</achterafBetalen>
```

10 APPENDIX D: SUPPORTED PAYMENT ATTRIBUTES

The GetPayment call may return one or more Attributes. These are name-value pairs that can give additional information regarding the payment. The following attributes are supported:

- FinanceInvoiceID – the ID of the invoice this payment was invoiced on
- FinanceEstateID – the ID of the Statement this payment was included on
- AuthorisationCode – for Payvision Creditcard transactions only: An XML structure with the codes returned by Payvision indicating the transactions was accepted or not, see below for an example
- RedshieldResponse – only for Creditcard transactions where Fraud Check is enabled: a JSON structure with information from ACI Redshield regarding the transaction, including BIN and Expiry Date, see below for an example.

10.1 PAYVISION AUTHORISATION CODE

See example below:

```xml
<?xml version="1.0"?>
<TransactionResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<ExtensionData />
<Cdc>
  <CdcEntry>
    <ExtensionData />
  </CdcEntry>
  <Items>
    <CdcEntryItem>
      <ExtensionData />
      <Key>BankCode</Key>
      <Value>00</Value>
    </CdcEntryItem>
    <CdcEntryItem>
      <ExtensionData />
      <Key>BankMessage</Key>
      <Value>Approved</Value>
    </CdcEntryItem>
  </Items>
</Cdc>
```
10.2 REDSHIELD RESPONSE

See example below:
The following is a list of possible fraud statuses:

<table>
<thead>
<tr>
<th>STAT_CD</th>
<th>FRAUD_STAT_CD</th>
<th>FRAUD_RSP_CD</th>
<th>Description</th>
<th>Full Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT/DECLINE</td>
<td>NOSCORE</td>
<td>0000</td>
<td>No decision provided</td>
<td>Transaction not screened. No fraud screening decision is given.</td>
</tr>
<tr>
<td>ACCEPT/DECLINE</td>
<td>ACCEPT</td>
<td>0100</td>
<td>Transaction accepted</td>
<td>Accept</td>
</tr>
</tbody>
</table>

The following is a list of possible fraud statuses:
<table>
<thead>
<tr>
<th>ACCEPT/DECLINE</th>
<th>ACCEPT</th>
<th>0150</th>
<th>Always accept rule</th>
<th>An attribute associated with an Order matched a pre-configured &quot;Always Accept&quot; rule.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>0200</td>
<td>Transaction took place on a lost or stolen card</td>
<td>The card number appeared in a bank or card association negative file database.</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>0250</td>
<td>Always deny rule</td>
<td>An attribute associated with an Order matched a pre-configured &quot;Always Deny&quot; rule.</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>CHALLENGE</td>
<td>0300</td>
<td>Transaction challenged due to custom rules</td>
<td>A combination of customised rules and neural-based fraud assessments has determined the card usage is suspicious and possibly fraudulent.</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>CHALLENGE</td>
<td>0330</td>
<td>Transaction hit a velocity or rule threshold</td>
<td>A customised rule in the ACI ReD Shield Velocity Rules Engine returned a CHALLENGE response.</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>0400</td>
<td>Denied transaction due to a hit on a suspicious database and a high neural score</td>
<td>A combination of customised rules and neural-based fraud assessments has determined the card usage is suspicious and possibly fraudulent and the card number appeared in a ACI Retail Decisions card database.</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>CHALLENGE</td>
<td>0500</td>
<td>Challenged transaction due to high neural score</td>
<td>A combination of customised rules and neural-based fraud assessments has determined the card usage is questionable and possibly fraudulent. The overall ACI ReD Shield assessment has fallen into a...</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>0600</td>
<td>&quot;grey area&quot;, as defined by ACI and the Client.</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>0700</td>
<td>The card number associated with the Order was found in a ACI Retail Decisions card database.</td>
<td></td>
</tr>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>0800</td>
<td>Velocity or Rules Threshold Violation – An attribute associated with an Order has exceeded a preconfigured rules threshold.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUSPEND</th>
<th>DENY</th>
<th>0700</th>
<th>Velocity or Rules Threshold Violation – An attribute associated with an Order has exceeded a preconfigured rules threshold.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>0800</td>
<td>Tumbling and/or Swapping Pattern Detected – The ACI ReD Shield Tumbling and Swapping engine detected an unusual usage pattern in the card number, expiration date, or customer email address associated with a transaction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACCEPT/DECLINE</th>
<th>ACCEPT</th>
<th>1000</th>
<th>An attribute associated with an Order matched a pre-configured “Always Accept” screening entry in the ACI ReD Shield proprietary screening database service.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSPEND</td>
<td>CHALLENGE</td>
<td>1300</td>
<td>The transaction has been flagged in a screening database.</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>DENY</td>
<td>2000</td>
<td>Fraudulent transactions have already been seen with details in your request</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>ERROR</td>
<td>901</td>
<td>System Error or missing fields</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>ERROR</td>
<td>902</td>
<td>Information missing/bad data</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>ERROR</td>
<td>903</td>
<td>Request exceeds limit</td>
</tr>
</tbody>
</table>
