ICEPAY WEB SERVICE

Implementation Guide

Version 1.5.0
24-september-2013

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 WEBSERVICEA 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 www.icepay.eu 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

Date	Author	Version	Comment
24-sep-2013	Huy Hoang	1.5.0	Added CheckoutExtended
4-june-2013	Alessandro Tomeo	1.4.0	- added Recurring Payments
26-june-2012	Sander van Tilburg	1.3.6	- Fixed checksum for SearchPayments.
22-may-2012	Sander van Tilburg	1.3.5	- Updated document for new webservice URL's.
06-sep-2011	Sander van Tilburg	1.3.4	- Added comment about UTF-8 character encoding.
24-june-2011	Sander van Tilburg	1.3.3	- Added new certificate
06-may-2011	Sander van Tilburg	1.3.2	- New layout - Added UTF-8 postback description
03-feb-2011	Sander van Tilburg	1.3.1	- Updated document for Paysafecard - Added some extra explanations.
27-sep-2010	Huy Hoang	1.3	- Added a new chapter about the *Refund Web Service*. - Added new error codes: ERR_0016, ERR_2000 to ERR_2007 - Minor text changes throughout the entire document.
14-jul-2010	Huy Hoang	1.2.1	Fixed example in chapter 6.8
8-jun-2010	Huy Hoang	1.2	- Added License Conditions - Revised document structure - Inserted a new segment about session operations at chapter 6, thus moving the original chapter 6 to 7, etc. - Added new exceptions to chapter 7 - Minor text revisions throughout the entire document
09-mar-2010	Huy Hoang	1.1.2.1	Changed the postback interval (defined in chapter 3.1) from 10 to 30 seconds
15-dec-2009	Huy Hoang	1.1.2	Added text to paragraphs 4.5 and 4.6 regarding the validation code, added paragraph 5.2.1, and minor text revision throughout the entire document.
08-dec-2009	Huy Hoang	1.1.1	Corrected the flow diagram in paragraph 10.2, paragraph 4.2 revised. Added a link pointing to the Supported Parameters Sheet.
07-dec-2009	Huy Hoang	1.1	Added SMS payment flow, paragraph 3.1 revised, changed the flow chart in paragraph 5.1.3, added appendix D, and added new web methods: PhoneDirectCheckout, GetPremiumRateNumbers and GetPayment.
17-sep-2009	Huy Hoang	1.0.3	Minor text changes
7-sep-2009	Huy Hoang	1.0.2	Minor text changes
26-aug-2009	Huy Hoang	1.0.1	Added a new paragraph
18-aug-2009	Huy Hoang	1.0	Document revised
14-aug-2009	Huy Hoang	0.5	Appendices added
13-aug-2009	Huy Hoang	0.4	Document revised

6-aug-2009	Huy Hoang	0.3	Minor text changes
25-jun-2009	Huy Hoang	0.2	Minor text changes
16-jun-2009	Huy Hoang	0.1	Initial draft

TABLE OF CONTENTS

PREFACE

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.

Chapters 2 to 5 describe the possibilities of the *ICEPAY Checkout web service*.

Chapter 2 “Implementation Strategies” covers the *basic integration* and *seamless integration* approaches. The basic integration saves you time by using default payment screens, whereas the seamless integration allows you to create your own payment screens and use web method calls to finish the payment process (only available for SMS and phone payments).

Chapter 3 “Seamless Integration in Details” covers the flow of payment methods and giving tips on which web methods to use.

Chapter 4 “Web methods” covers the various methods which your system can invoke in order to initialize a payment, query information from ICEPAY, etc.

Chapter 5 “Postback Notifications” covers the *Postback notification*, which is how your system will be informed regarding the status of your transactions.

Chapter 6 “Reporting web service” solely covers the *ICEPAY Reporting Web Service*, which allows your client applications to query statistical data from your ICEPAY account.

Chapter 7 “Refund web service” solely covers the *ICEPAY Refund Web Service*, which allows your client applications to programmatically initiate refund requests.

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:

http://www.icepay.com/downloads/pdf/documentation/ICEPAY_Webservice.pdf

ICEPAY WEB SERVICE ESSENTIALS

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

1.1 WHAT IS THE ICEPAY WEB SERVICE?

The ICEPAY Web Service is an exciting new 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 chapters 2 to 5.

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 6 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:

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

The WSDL can be found here:

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

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 7 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!

2 IMPLEMENTATION STRATEGIES

There are several *implementation strategies* for you to choose from when you want to integrate ICEPAY with your own system using the *Checkout Web Service*. Both have their advantages and disadvantages.

2.1 BASIC INTEGRATION

The most basic integration strategy is to use the *Checkout* web method. The general idea is that you provide the ICEPAY web service the required parameters, and it will return you the results in XML. Inside the XML body, you will find a URL to the payment screen. Use that URL to redirect your end-users to the payment screen. From that moment on, everything will be handled by ICEPAY. Your server will receive a *Postback Notification* once the user has finished or cancelled the payment. Please look at chapter 5 for more information regarding the Postback Notification.

The basic integration is simple and offers ready-to-use ICEPAY payment screens. However, its disadvantage is that you cannot have full control over the payment screens. Also, your target implementation platform must be a website.

Please take a look at *Appendix A* for a visualization of the flow of the basic integration strategy.

2.2 SEAMLESS INTEGRATION

There are times that you want to have full control over your payment screen. How it looks, how it interacts with your user, and preferably as part of your website. For that purpose, we offer the *seamless integration strategy*. The idea is that your implementation platform provides the ICEPAY web service the required parameters. We will then return you the results back in XML. Inside the XML body, you will find information which you can use to display to the end-user in your own existing website or any other platform.

Please take a look at *Appendix B* for a visualization of the flow of a possible seamless integration.

THIS STRATEGY IS CURRENTLY ONLY AVAILABLE FOR 'SMS' AND 'PHONE' PAYMENTS.
THERE ARE CURRENTLY NO PLANS TO MAKE OTHER PAYMENT METHODS AVAILABLE FOR THIS TYPE OF INTEGRATION.

2.3 PICK YOUR IMPLEMENTATION STRATEGY

Now that you know the different strategies that are available, you should pick the strategy that suits your requirements.

3 SEAMLESS INTEGRATION IN DETAILS

This chapter will provide more details regarding seamless integration for phone and SMS payments.

3.1 PHONE PAYMENTS

If you want to implement phone payments using the seamless integration approach, then it is imperative that you understand how phone payments work. The ICEPAY web service supports two types of phone payment methods: *Pay Per Call* and *Pay Per Minute*.

Pay Per Call is suitable for payments in which you want to charge the end-user a fixed amount of money.

Pay Per Minute on the other hand, is meant to charge the end-user an unspecified amount of money. This type of phone payment is suitable for online services for which the end-user will be granted access as long as they don't hang up the phone.

The end-user flow for both payment methods can be like this:

1. Your website offers a service which can be paid using the phone payment
2. The end-user sees a premium-rate number on your website
3. The end-user dials the premium-rate number
4. The end-user will hear a six digit code
5. The end-user enters the six digit code on your website
6. The six digit code is verified and depending on the result it will go to step 6 or 7
7. The six digit code is invalid, and the end-user needs to repeat step 4
8. The six digit code is correct, and the end-user proceeds to the secured page

There are two solutions to achieve the above flow:

The first solution involves the use of the *PhoneCheckout* web method (please see paragraph 4.2) for initializing the payment and getting back data such as the premium-rate number, rate per call, and rate per minute. You must display this information to the end-user. Otherwise, how would the end-user know which premium-rate number to dial? The *ValidatePhoneCode* web method (please see paragraph 4.4) is needed for verifying the code that the end-user enters. After the six digit code is entered successfully, your system will automatically get a *Postback notification*. In case of *Pay Per Minute*, you will also receive them every 30 seconds, indicating that the user is still connected. We advise you to use this information to update your local database. In addition, you could use AJAX requests on your payment page to poll your local database and display some nice progress bar or to find out if the user has already hung up or not.

The second solution is involves the use of the *PhoneDirectCheckout* web method (please see paragraph 4.6) to initialize the payment and simultaneously also validating the code. So, every time the end-user enters the code, you will call this web method. The difference with the first solution is that initialization and code validation is done in one single call. You then might be wondering how the end-user knows to which premium-rate number he/she must dial? Otherwise, how would he/she know the code? There is a supplementary web method called *GetPremiumNumbers* which returns a list of premium-rate numbers that is assigned to your ICEPAY API key. You can then cache this so that you do not have to query ICEPAY upon every transaction. If you do cache this information, then we recommend you to call *GetPremiumNumbers* at least once every 24 hours to refresh your cache.

3.2 SMS PAYMENTS

If you want to implement SMS payments using the seamless integration approach, then it is imperative that you understand how SMS payments work. There are two end-user flows, both which must be supported by your payment screen.

3.2.1 DEFAULT END-USER FLOW

Let us take a look at how the default end-user flow looks like:

1. Your website offers a service that can be paid using a simple SMS.
2. Your website displays a telephone number, called a short code, to which the end-user must send an SMS. The SMS will be a keyword + activation code, e.g. *ICE 1234*.
3. The end-user sends an SMS to the short code.
4. The end-user receives a “Thank You” SMS from ICEPAY.
5. The end-user sees that the payment is completed.

3.2.2 END-USER VALIDATION FLOW

There are cases in which ICEPAY cannot verify that the SMS payment is completed. In that case, the end-user flow is somewhat different.

1. Your website offers a service that can be paid using a simple SMS.
2. Your website displays a telephone number, called a short code, to which the end-user must send an SMS. The SMS will be a keyword + activation code, e.g. *ICE 1234*.
3. The end-user sends an SMS to the short code.
4. The end-user receives an SMS message from ICEPAY. The SMS message contains a validation code.
5. On the website, the end-user enters the validation code and submits.
6. If the validation code is correct, then the end-user sees that the payment is completed.

For a visual diagram of the flow, please take a look at appendix D.

3.2.3 HOW TO KNOW WHICH FLOW TO APPLY?

Both flows have a beginning and end that are the same. If you have not done so already, please check appendix D. Initializing the payment, handling the Postback with the status OK is the same for both flows. The signal for switching to the end-user validation flow is when your system receives a Postback with the status VALIDATE. You should then display a text input where the user can enter the validation code. If the validation code is correct, you will receive a True in the response (please see paragraph 4.5.2 for more information). In addition, a Postback with the status OK is sent.

4 CHECKOUT WEB SERVICE WEB METHODS

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.

4.1 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.

4.2 CHECKOUT

The *Checkout* web method allows you to initialize a new payment in the ICEPAY system for **ALL** the payment methods that you have access to.

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*. It does not contain data that can be used for seamless integration. If that is what you try to accomplish, then please take a look at chapter 4.2 and 4.3 which covers the phone and SMS checkout respectively.

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:

http://www.icepay.com/downloads/pdf/documentation/ICEPAY_Supported_Parameters_Sheet.pdf

Compatible with
Basic integration

4.2.1 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: http://www.icepay.com/downloads/pdf/documentation/ICEPAY_Supported_Parameters_Sheet.pdf

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String

Amount	This is the amount (in cents) that will be charged	Integer
Country	This member will be interpreted differently depending on the chosen payment method. For phone and SMS payments, this will be the country in which the payment will take place. For all other payment methods, it is used to validate if the payment method is supported in a certain country.	Integer
Currency	This is the currency	String
Description	A description of the payment	String
EndUserIP	IP address of the end-user	String
Issuer	This is the issuer of the payment method	String
Language	Language of the payment screen	String
OrderID	A unique code up to 10 characters. *It is recommended to use the primary key field of your local transactions table*	String
PaymentMethod	This is the payment method that will be used for the transaction initialization	String
Reference	Custom information that you want to include, e.g. primary key of your local transactions table (up to 50 characters)	String
URLCompleted	The URL to which the end-user will be redirected *If you do not set this member then the URLCompleted that you set in your ICEPAY account will be used*	String
URLError	The URL to which the end-user will be redirected *If you do not set this member then the URLError that you set in your ICEPAY account will be used*	String

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. URLCompleted
15. URLError

Example

```
secret|12345|2009-06-09T01:30:00Z|100|NL|EUR|Test|127.0.0.1|AMEX|NL|1|CREDITCARD|MyReference||
```

4.2.2 RESPONSE

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key.	Integer
Checksum	This is the SHA1 digest of all the members. You can use this value to match your own checksum to see if the response is actually from ICEPAY.	String
Timestamp	This will be the current UTC time that has the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
Amount	The requested amount in cents	Integer
Country	This is the requested country code	String
Currency	This is the requested currency	String
Description	This is the specified description	String
EndUserIP	This is the provided end user IP	String
Issuer	This is the requested issuer	String
Language	This is the requested language code	String
OrderID	This is the provided Order ID	String
PaymentID	This is the generated ICEPAY transaction ID. **Tip:** if possible please store this in your local database as this ID may come in handy if you contact our support department.	Integer
PaymentMethod	This is the requested Payment Method	String
PaymentScreenURL	This is the URL of the payment screen (if available)	String
ProviderTransactionID	This is the transaction ID of the issuer (if available)	String
Reference	This is the specified reference information	String
TestMode	Indicates whether the transaction was initialized in test mode. This is true if your API key is still in test mode. To switch to production mode, please contact your account manager	Boolean
URLCompleted	This is the page to which the end-user will be redirected to after a successful transaction	String

URLError	This is the page to which the end-user will be redirected to after a failed or cancelled transaction	String

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|2009-06-09T01:30:00Z|100|NL|EUR|Test|127.0.0.1|AMEX|NL|1|100123456|CREDITCARD|https://live.icepay.eu|01234|MyReference|false||
```

4.3 PHONECHECKOUT

The *PhoneCheckout* web method allows you to create a phone payment in the ICEPAY system. The main difference with the *Checkout* web method is the response. The response is a *PhoneCheckoutResponse* object, which contains extra members such as the phone number etc., making seamless integration possible.

Compatible with
Basic integration, seamless integration

4.3.1 REQUEST

Requires a *CheckoutRequest* object. For more information, please take a look at chapter 4.2.1

4.3.2 RESPONSE

Returns the *PhoneCheckoutResponse* object, which is an extension of the *CheckoutResponse* object. In addition to the *CheckoutResponse* members (see paragraph 4.1.2), it includes the following extra members that are specific to phone payments:

Member	Description	Data type
PhoneNumber	This is the phone number to which the end-user must dial	String
RatePerCall	The rate per call in cents	Integer
RatePerMinute	The rate per minute in cents	Integer

The checksum for the *PhoneCheckoutResponse* can be calculated in the same manner as the checksum described in 4.1.2, but the following three members need to be added also:

20. PhoneNumber
21. RatePerCall
22. RatePerMinute

Example
secret|12345|2009-06-09T01:30:00Z|100|NL|EUR|Test|127.0.0.1|PPM|NL|1|100123456|PHONE|https://live.icepay.eu|01234|My Reference|false|||0909123456|100|0

4.3.3 SEE ALSO

Chapter 4.6 for the *PhoneDirectCheckout*

4.4 SMSCHECKOUT

The *SmsCheckout* web method allows you to create an SMS payment in the ICEPAY system. The main difference with the *Checkout* web method is the response. The response will contain extra members such as the premium-rate number, making seamless integration possible.

Compatible with
Basic integration, seamless integration

4.4.1 REQUEST

Requires a *CheckoutRequest* object. For more information, please take a look at paragraph 4.1.1

4.4.2 RESPONSE

Returns the *SMSCheckoutReponse* object, which is an extension of the *CheckoutResponse* object. In addition to the *CheckoutResponse* members (see paragraph 4.1.2), it includes the following extra members that are specific to SMS payments:

Member	Description	Data type
ActivationCode	This is the activation code that also needs to be sent to the premium-rate number	String
Keyword	This is the keyword that needs to be sent to the premium-rate number	String
PremiumNumber	This is the premium-rate number or short code to which the user must send a MO message to	String
Disclaimer	This is the disclaimer text that must be put on your payment screen. Failing to do so will result in your ICEPAY account being suspended. Remarks: although this member is always available, it is not always populated.	String

The checksum for the *SMSCheckoutResponse* can be calculated in the same manner as the checksum described in 4.1.2, but the following three members need to be added also:

20. ActivationCode
21. Keyword
22. PremiumNumber
23. Disclaimer

Example

secret|12345|2009-06-09T01:30:00Z|100|NL|EUR|Test|127.0.0.1|DEFAULT|NL|1|100123456|SMS|https://live.icepay.eu|01234|MyReference|false|||1234|ICE|2211||false

4.5 VAULTCHECKOUT

The method is called when is wished to vault a consumer information as credit card or bank account number in order to perform an automatic checkout in the future.

That information is stored after the above mentioned payment has been successfully completed.

4.5.1 REQUEST

Requires a *VaultCheckoutRequest* object.

4.5.1.1.1 PARAMETERS LIST (VAULTCHECKOUTREQUEST)

Member	Description	Data type
MerchantID	Is the ID of the merchant provided by Icepay when creating a new merchant.	integer
PaymentMethod	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”	String
Amount	Ids the amount of the transaction	integer
Language	Is the string corresponding to the iso code of the language ex. Dutch is ‘NL’	string
Currency	Is the String corresponding to the iso code of the currency ex. Euro is ‘EUR’	string
Country	Is the string corresponding to the iso code of the country ex. Netherlands is ‘NL’	string
Issuer	Is the issuer connected to the payment method Ex. For credit card can be ‘VISA’ or ‘MASTERCARD’	string
ConsumerID	Is the id which is wished to link to the consumer credit card or bank account to perform automatic checkouts. It can be alphanumeric.	string
Timestamp	It is the timestamp referring to the payment	string
OrderID	It is the Unique OrderId of the transaction	string
Description	It is the description which appears along the payment in the ICEPAY’s environment.	String
EndUserIP	Is the ip address of the customer’s machine	string
URLCompleted	Is the url where the user is redirected after successful payment.	string
URLError	Is the url where the user is redirected after erroneous payment.	string
Reference	This field can be empty	string

Checksum	Is the checksum of all the fields used in a regular checkout therefore excluding ConsumerID	string

4.5.2 RESPONSE

Returns the *CheckoutResponse* object (see paragraph 4.1.2).

4.5.3 USAGE

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

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

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

```
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# is not meant to work if copy/pasted.

```
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)
}
```

4.6 AUTOCHECKOUT

This method is called when is needed to perform an automatic checkout using the stored consumerID, is available for credit cards and the Ideal system.

4.6.1 REQUEST

Requires an *AutomaticCheckoutRequest* object.

4.6.1.1 PARAMETERS LIST (AUTOMATICCHECKOUTREQUEST)

Member	Description	Data type
MerchantID	is the Id of the ICEPAY merchant which is performing the automatic checkout	Integer
PaymentMethod	Can have two values: • 'ddebit' this is required to perform an automatic checkout prior storing a iDeal accout number • '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 bill.	Integer
Language	is the language codeof 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	this depends on the payment method insert before: • For 'creditCard' must be 'CCAUTOCHECKOUT' • For 'ddebit' must be 'IDEALINCASSO'	String
ConsumerID	this is the consumer id which was vaulted previously using the VaultCheckout method. When testing use 123456 to generate test success (it will return success only if everything else is correct).	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 compulsory description can be left empty	String

EndUserIP	is the ip address of the machine from which the action is performed.	Srting
URLCompleted	is the url where the user lands on a successful transaction.	String
Reference	can be and 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

4.6.2 RESPONSE

Returns the *AutomaticCheckoutResponse* object

4.6.2.1 PARAMETERS LIST (AUTOMATICCHECKOUTRESPONSE)

Member	Description	Data type
MerchantID	is the Id of the ICEPAY merchant which is performing the automatic checkout	Integer
Success	Indicates whether the operation had been successfully completed	Boolean
TimeStamp	Indicates the timestamp of the response.	String
PaymentID	Is the ICEPAY Id of the payment	Integer
ErrorDescription	If errors are present indicates the message of the error	String
Checksum	Is the checksum SHA1 crypted of the above fields	String

4.6.3 USAGE

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

Finally process Response.

4.7 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: AfterPay

4.7.1 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. Please consult "Appendix E: CheckoutExtended XML" for the XSD of the XML message, as well as an example.

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

XML node	Description
Order/Products	This node contains information about the ordered products. In case of AfterPay, they will appear as lines on the invoice.
Order/Products/Product/ProductID	This is the ID of your product.
Order/Products/Product/UnitPrice	The amount (in cents) of the product including VAT
Order/Products/Product/VATCategory	A valid VAT category needs to be populated. The VAT rate will be determined automatically. **standard** = This is the standard (high) tariff, e.g. 21% in The Netherlands, 21% in Belgium, etc. **reduced-middle** = This is the middle VAT tariff in case of three different tariffs. Not applicable in The Netherlands, 12% in Belgium, etc. **reduced-low** = This is the lowest VAT tariff, e.g. 6% in The Netherlands, 6% in Belgium, etc. **zero** = 0% VAT **exempt** = Use this when the product is exempted from VAT, e.g. second hand books (in The Netherlands).

Order/Products/Product/Description	A description of the product. AfterPay accepts a maximum of 45 characters. Diacritics, etc. will be filtered by AfterPay automatically.
Order/Addresses/Address/ZipCode	This value will be validated. Make sure the format is correct for the specified country: 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.

4.7.2 RESPONSE

Please see 4.2.2

4.8 VALIDATEPHONECODE

The *ValidatePhoneCode* web method verifies the code that the end-user must provide in order to start a phone payment. For more information on how this web method fits in the big picture, please take a look at chapter 3.1.

Test Mode
When your API Key is in *Test Mode*, you will not know the code because you will not be dialing to a premium-rate number. The standard code that you can use for this method while you are in Test Mode is **123456**.

Compatible with
Seamless integration

4.8.1 REQUEST

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format:	String

	yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	
PaymentID	This is the ICEPAY transaction ID that you will get once you have successfully initialized a transaction using *PhoneCheckout*.	Integer
PhoneCode	This is the phone code that the user enters in order to start the phone payment	String

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. PaymentID
5. PhoneCode

Example

```
secret|12345|2009-06-09T01:30:00Z|10012345|98765
```

4.8.2 RESPONSE

Returns the *ValidatePhoneCodeResponse* object, which contains the following members:

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
PaymentID	This is the transaction that needed validation	Integer
Success	Indicates whether the code is valid or not	Boolean

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. Success

Example

```
secret|12345|2009-06-09T01:30:00Z|10012345|true
```

4.9 VALIDATESMSCODE

The *ValidateSmsCode* web method validates the code that the end-user must provide when the SMS payment is following the flow as described in paragraph 3.2.2, rather than the default flow as described in paragraph 3.2.1.

When your API Key is in *Test Mode*, you will not receive an SMS with code. The standard code that you can use for this method while you are in Test Mode is **test123**.

Compatible with
Seamless integration

4.9.1 REQUEST

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
PaymentID	This is the ICEPAY transaction ID that you will get once you have successfully initialized a transaction using *SmsCheckout*.	Integer
SmsCode	This is the SMS code that the user enters in order to validate the SMS code.	String

4.9.2 RESPONSE

Returns the *ValidateSmsCodeResponse* object, which contains the following members:

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format:	String

	yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	
PaymentID	This is the transaction that needed validation	Integer
Success	Indicates whether the code is valid or not	Boolean

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.

6. Secret code of your API key
7. MerchantID
8. Timestamp
9. PaymentID
10. Success

Example

```
secret|12345|2009-06-09T01:30:00Z|10012345|true
```

4.10 PHONEDIRECTCHECKOUT

The *PhoneDirectCheckout* web method is similar to the web method described in chapter 4.2. The difference is that using this web method, you will be able to provide an extra parameter.

4.10.1 REQUEST

Requires a *CheckoutWithPINRequest* object. This object inherits every member from the *CheckoutRequest* object (please see paragraph 4.2.1). The only extra member is the following:

Member	Description	Data type
PINCode	This is the code that the end-user hears when they are on the phone.	String

The checksum for the request object is calculated according to the formula described in chapter 4.2.1. The PINCode value needs to be concatenated to the string before passing it to the SHA1 function to calculate the checksum.

4.10.2 RESPONSE

Returns the *PhoneDirectCheckoutResponse* object. The object inherits every member from the *PhoneCheckoutResponse* object (please see paragraph 4.3.2). The extra members are the following:

Member	Description	Data type
Success	Indicates whether the provided PIN code is correct or not.	Boolean
ErrorDescription	A description of why the PIN code is incorrect.	String

The checksum for the *PhoneCheckoutResponse* can be calculated in the same manner as the checksum described in 4.3.2, but the following two members need to be added also:

1. Success
2. ErrorDescription

4.11 GETPREMIUMRATENUMBERS

The *GetPremiumRateNumbers* web method is supplementary to the *PhoneDirectCheckout* web method. The idea is that you query the latest premium-rate number information (such as rate per minute, etc.) and cache it on your own system so that you can display the premium-rate number information to the end-user without having to start a new transaction.

4.11.1 REQUEST

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key.	Integer
Checksum	This is the SHA1 digest of all the members. You can use this value to match your own checksum to see if the response is actually from ICEPAY.	String
Timestamp	This will be the current UTC time that has the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String

4.11.2 RESPONSE

Returns the *GetPremiumRateNumbersResponse* object, which contains the following members:

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest which is calculated according to the formula in 4.7.3.1	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
PremiumRateNumbers	This is an object which contains a list of objects of the *PremiumRateNumberInformation* class (please see paragraph 4.7.2.2)	List of objects

4.11.2.1 CHECKSUM

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. API key secret code
2. MerchantID
3. Timestamp
4. PremiumRateNumber
5. RatePerCall
6. RatePerMinute

If you get back more than 1 premium-rate number, then you will have to concatenate this to the string.

Example with one premium-rate number:

```
secret|12345|2009-06-09T01:30:00Z|0900 123456|0|130
```

Example with two premium-rate number

```
secret|12345|2009-06-09T01:30:00Z|0900 123456|0|130|0900 654321|0|80
```

4.11.2.2 PREMIUMRATENUMBERINFORMATION CLASS

An object of the PremiumRateNumberInformation class contains the following members:

Member	Description	Data type
Country	The country of the premium-rate number	String
Currency	The currency used by the premium-rate number	String
Issuer	The issuer type. This can be one of the following values: PPM = Pay Per Minute PPC = Pay Per Call	String
PhoneNumber	The number that the end-user can dial	String
RatePerMinute	The cost charged to the end-user per minute (in cents)	Integer
RatePerCall	The cost charged to the end-user per call (in cents)	Integer

4.12 GETPAYMENT

Calling the *GetPayment* web method will return you 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 lost it.

4.12.1 REQUEST

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
PaymentID	This is the ICEPAY PaymentID. You will receive this when you initialize a payment or in your Postback.	Integer

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

```
Secret|12345|2009-06-09T01:30:00Z|1234567
```

4.12.2 RESPONSE

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest which is calculated according to	String

	the formula in 4.8.2.1	
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
PaymentID	The ICEPAY PaymentID	Integer
Amount	The requested amount of the payment	Integer
Currency	The requested currency of the payment	Integer
Description	The description specified by the merchant during the initialization	
Duration	The numbers of seconds dialed. Only applicable for phone payments	Integer
ConsumerName	Name of the consumer (if available)	String
ConsumerAccountNumber	Partial account number of the consumer (if available)	String
ConsumerAddress	Address of the consumer (if available)	String
ConsumerHouseNumber	House number of the address of the consumer (if available)	String
ConsumerCity	City of the consumer (if available)	String
ConsumerCountry	Country of the consumer (if available)	String
ConsumerEmail	E-mail address of the consumer (if available)	String
ConsumerPhoneNumber	Phone number of the consumer (if available)	String
ConsumerIPAddress	IP address of the consumer (if available)	String
Issuer	Requested payment method issuer	String
OrderID	The OrderID specified by the merchant during the initialization	String
OrderTime	The time when the payment got started. In GMT.	String
PaymentMethod	The requested payment method	String
PaymentTime	The time indicating when the payment got closed/completed (either successful or not successful). In GMT.	String
Reference	The reference specified by the merchant during the initialization	String
Status	The status of the payment (please see 5.1.1 for possible statuses)	String
StatusCode	A description giving you more information on the status	String

TestMode	Indicates whether the payment was created when the API key was still in test mode.	Boolean

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
22. PaymentTime
23. Reference
24. Status
25. StatusCode
26. TestMode

Example

```
secret|12345|2009-12-01T17:51:53.760Z|1234567|3500|||||||||EUR|Test|0|VISA|@100000007|2008-10-07T11:24:51.963Z|CREDITCARD|2009-08-21T17:13:22.270Z||ERR||false
```

5 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://www.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.

Parameter	Description	Data type	Sample	Can be empty
Status	Payment status as OK, OPEN, ERR, REFUND, CBACK	String(10)	OPEN	N
StatusCode	A short description of the status. We will use the codes as received from the payment method provider	String(100)	Completed	N
Merchant	Your MerchantID	Numeric		N
OrderID	The OrderID member that was provided in the request object	String(10)	1234567	N
PaymentID	A unique numeric value that identifies this payment in our system	Numeric	12345	N
Reference	The Reference member that was provided in the request object	String(50)	Z1234567	Y
TransactionID	This value is created by the payment method provider / bank and showed on the user's bank statement	String(50)		Y
ConsumerName	Name of the bank account owner	String(100)		Y
ConsumerAccountNumber	Last 4 digits of accountnumber from which the payment was done, if received from the bank	String(100)		Y
ConsumerAddress	Consumer address/street as	String(100)		Y

	filled in payment form			
ConsumerHouseNumber	Consumer house number as filled in payment form	String(10)		Y
ConsumerCity	Consumer city as filled in payment form	String(100)		Y
ConsumerCountry	Consumer country as filled in payment form	String(100)		Y
ConsumerEmail	Consumer email value as filled in payment form	String(200)		Y
ConsumerPhoneNumber	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}	String(50)		Y
ConsumerIPAddress	IP address from which the payment form was filled in	String(50)	1.2.3.4	Y
Amount	The final paid amount value in whole cents	Numeric	550	Y
Currency	The currency in which the amount is represented	String(3)	EUR	Y
Duration	The number of seconds dialed. Only available for phone payment methods	Numeric	0	Y
PaymentMethod	The payment method which was used	String(20)	CREDITCARD	N
Checksum	The checksum that is generated over the return parameters, so that you can verify the authenticity of the returned values	String(40)		N

5.1 HANDLING THE POSTBACK NOTIFICATION

This section provides you with guidelines on how your Postback Script should handle incoming *Postback Notifications*. Paragraph 5.2 describes how you can test your Postback script.

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

Status	Description
OPEN	The payment is not yet completed. 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.
OK	The payment has been completed.
ERR	The payment was not completed successfully or expired. It cannot change into anything else.
REFUND	A payment has been successfully refunded.
CBACK	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.
VALIDATE	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.

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

5.1.2 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

5.1.3 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*.

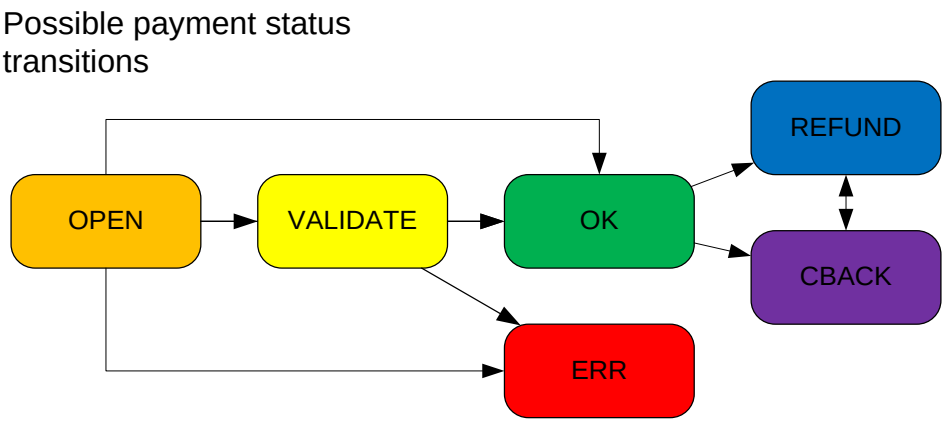


Figure 1. Possible payment status transitions

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

5.1.4 CHECKING THE CHECKSUM

You should always generate a checksum using the information that you get and compare this with the checksum that you received. If it does not match, then you should always ignore the incoming *Postback Notification*.

5.1.5 LOGGING

You should log **ALL** incoming *Postback Notifications* as it might come in handy for debugging purposes and/or requesting support from ICEPAY.

5.2 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!

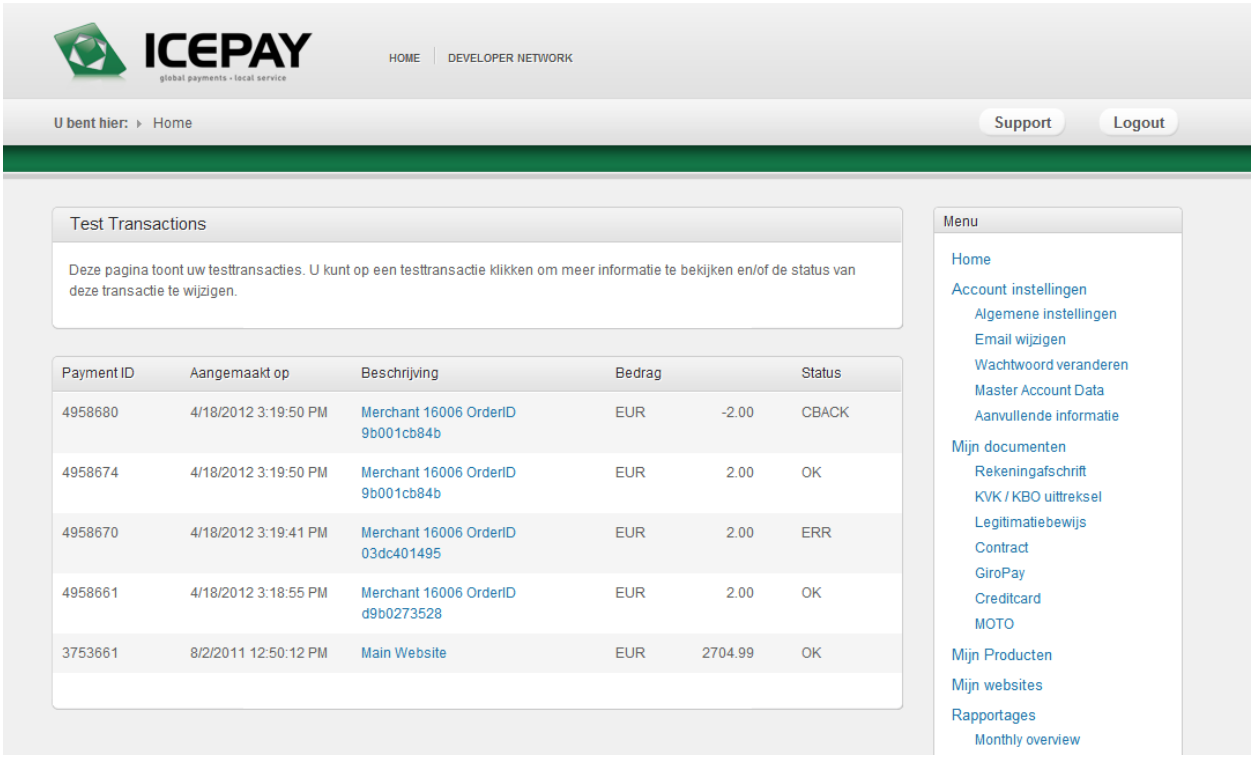


Figure 1

5.2.1 SMS PAYMENTS

You cannot use this tool to change SMS payments from VALIDATE to OK. You can change it from VALIDATE to OK only if you invoke the method *ValidateSmsCode* (please see paragraph 4.6).

The reason for this is that in *Production Mode*, the transition from VALIDATE to OK will only happen when your application invokes the *ValidateSmsCode* method with the correct code which means that the end-user has provided your application with the correct code.

6 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.

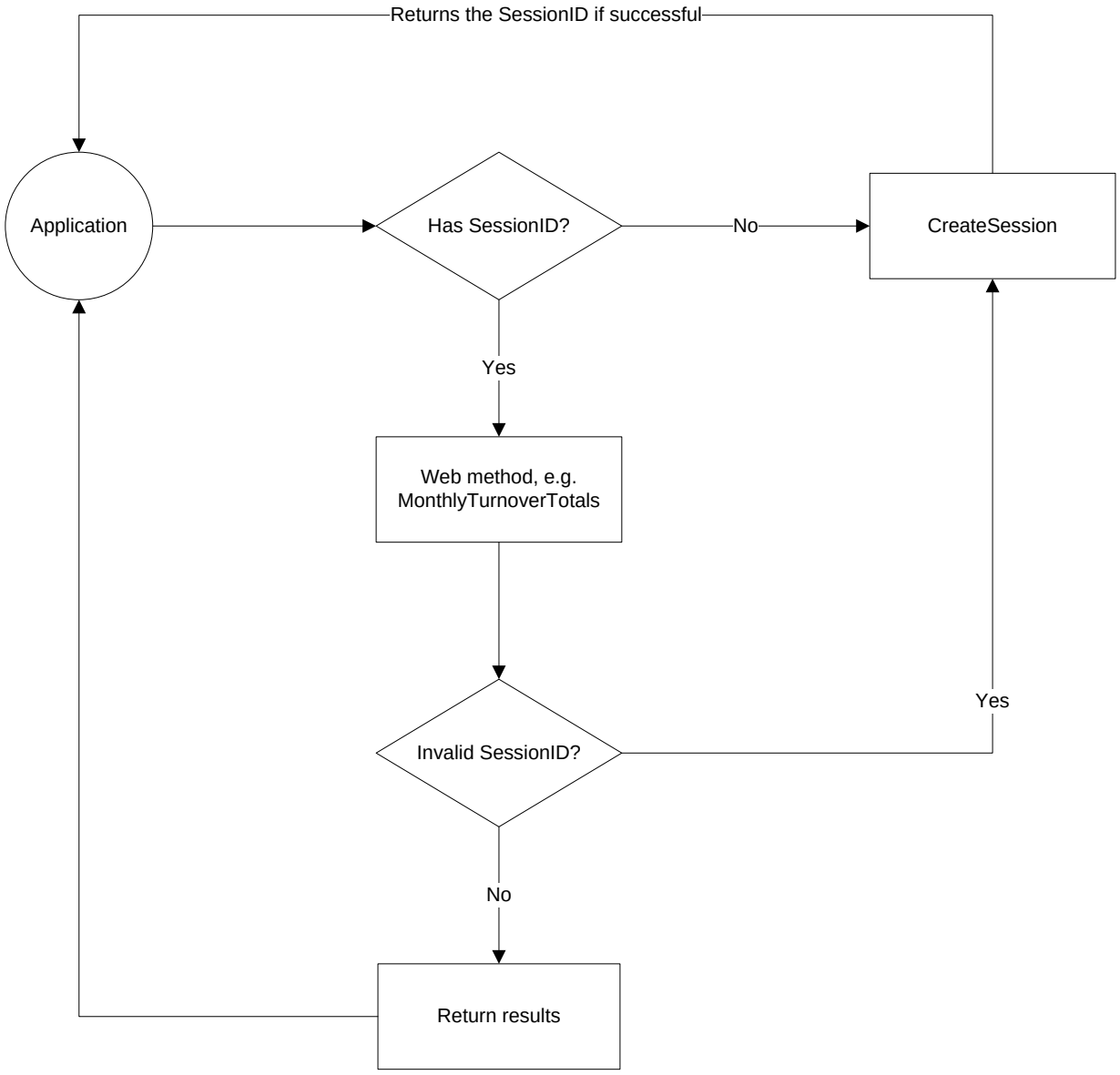
6.1 ALLOW REPORTING ACCESS

In order for your application to access the ICEPAY account information, the option "*Allow Reporting Access*" must be enabled in the ICEPAY account. 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.

6.2 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:



6.3 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.

6.3.1 INPUT

The following parameters are expected for the CreateSession method:

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
Username	This is your ICEPAY username	String
UserAgent	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.	String(15..50)
Checksum	This is the SHA1 message digest to verify the authenticity of the request.	String

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

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

Example

```
2009-06-09T01:30:00Z|icepay|12345678|ICEPAY Mobile App 1.0 +31612345678
```

6.3.2 RESPONSE

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

Member	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
Success	Indicates whether the session was created or not	Boolean
Description	A short description which tells you why it failed. Your script should never rely on this description as it might change.	String
SessionID	The session ID which you must use for all the other session ID.	String
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String

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

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

Example

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|true|LoginSuccess
```

6.4 KILLSESSION

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.

6.4.1 INPUT

The following parameters are expected for the KillSession method:

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	This is the session ID created by the CreateSession method	String
UserAgent	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.	String(15..50)
Checksum	This is the SHA1 message digest to verify the authenticity of the request.	String

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

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

Example

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0
```

6.4.2 RESPONSE

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

Member	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	The session ID which you must use for all the other session ID.	String
Success	Indicates whether the session was destroyed or not.	Boolean
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String

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

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

Example

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|true
```

6.5 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.

6.5.1 INPUT

The following input parameters are required for this method:

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

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

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|12345|EUR|2010|5
```

6.5.2 RESPONSE

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

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	The SessionID that initiated the request	String
CurrencyCode	Specifies in what currency the results are.	String
Year	Specifies the year of the results	Integer
Month	Specifies the month of the results	Integer
Days	A list of *DayStatistics* objects	DayStatistics[]
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String

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

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|EUR|2010|5|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.

6.6 GETMERCHANTS

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

6.6.1 REQUEST PARAMETERS

The following input parameters are required for this method:

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	This is the session ID created by the CreateSession method	String
UserAgent	The same user agent that you provided the CreateSession method with	String
Checksum	This is the SHA1 message digest to verify the authenticity of the request.	String

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

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

Example

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0
```

6.6.2 RESPONSE

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

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	The SessionID that initiated the request	String
Merchants	A list of Merchant objects. Please see 0 for the structure of a single Merchant object.	Merchant[]
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String

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

2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|10000|My test website|true|10001|My production website|false

6.6.2.1 MERCHANT

This object contains information about a merchant.

Parameter	Description	Data type
MerchantID	The merchant ID	Integer
Description	Name of the merchant (as entered by the merchant in the ICEPAY back end)	String
TestMode	Indicates whether the merchant record is in Test Mode.	Boolean

It is recommended that you cache your results

6.7 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.

6.7.1 INPUT

The following input parameters are required for this method:

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	This is the session ID created by the CreateSession method	String
UserAgent	The same user agent that you provided the CreateSession method with	String
Checksum	This is the SHA1 message digest to verify the authenticity of the request.	String

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

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

Example

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0
```

6.7.1.1 RESPONSE

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

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	The SessionID that initiated the request	String
PaymentMethods	A list of PaymentMethod objects. Please see 0 for the structure of a single PaymentMethod object.	PaymentMethod[]
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String

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

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|iDEAL|IDEAL|Credit card|CREDITCARD|Wire transfer|WIRE
```

6.7.1.2 PAYMENTMETHOD OBJECT

This object contains information about a payment method.

Parameter	Description	Data type
Description	A description of the payment method	String
PaymentMethodCode	This is the payment method code of a payment method	String

6.8 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.

6.8.1 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.

Parameter	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	This is the session ID created by the CreateSession method	String
UserAgent	The same user agent that you provided the CreateSession method with	String
MerchantID	Filter on the MerchantID	Integer (optional)
PaymentID	Filter on the PaymentID	Integer (optional)
Status	Filter on the status of payments, e.g. OK, OPEN, ERR.	String (optional)
OrderTime1 *	Filter on the creation time of the payment. This parameter will be used for the start range of the filter.	String (optional)
OrderTime2 *	Filter on the creation date of the payment. This parameter will be used for the end range of the filter.	String (optional)
PaymentTime1 *	Filter on the time when the payment was completed. This parameter will be used for the start range of the filter.	String (optional)
PaymentTime2 *	Filter on the time when the payment was completed. This parameter will be used for the end range of the filter.	String (optional)

Please check the following page for the remaining parameters

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

** 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:

Timestamp|SessionID|ReportingPinCode|UserAgent|MerchantID|PaymentID|OrderID|Reference|Description|Status|OrderTime1|OrderTime2|PaymentTime1|PaymentTime2|

CountryCode|CurrencyCode|Amount|PaymentMethod|ConsumerAccountNumber|ConsumerName|ConsumerAddress|ConsumerHouseNumber|ConsumerPostCode|ConsumerCity|ConsumerCountry|ConsumerEmail|ConsumerPhoneNumber|ConsumerIPAddress|Page

Example

2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|12345|||||OPEN|||||||30|IDEAL||||||||||1

6.8.2 RESPONSE

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

Member	Description	Data type
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
SessionID	The SessionID that initiated the request	String
Payments	A list of *Payment* objects	Payment[]
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String

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

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

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

```
Amount|ConsumerAccountNumber|ConsumerAddress|ConsumerHouseNumber|ConsumerName|ConsumerPostCode|CountryCode|CurrencyCode|Duration|MerchantID|OrderTime|PaymentID|PaymentMethod|PaymentTime|Status|StatusCode|TestMode
```

Example

```
2009-06-09T01:30:00Z|ABCDEFGHIJKLMNOPQRSTUVWXYZ|12345678|My Reporting App 1.0|100||||John Doe||NL|EUR|0|12345|2009-06-09T01:30:00Z|1000000|IDEAL|2009-06-09T01:30:00Z|OK|Success|false
```

7 AFTERPAY CAPTURE WEB SERVICE

Afterpay payments can be captured using the following webservice; when an afterpay payment is done to make it effective is necessary to capture it via thehis webservice is possible to do the capture.

To perform a capture is therefore necessary to call the method *CaptureFull* using the following parameters:

Member	Description	Data type
MerchantID	Is the merchantId provided by icepay	Integer
Checksum	Is the SHA1 checksum generated using MerchantID,SecretCode,Timestamp,PaymentID in the same order.	String
Timestamp	Is the timestamp at which is desired to make the capture appear.	String
PaymentID	Is the paymentID of an Afterpay payment.	String

The method returns an object type *RequestAPCaptureResponse* which contains the following information:

Member	Description	Data type
MerchantID	Is the merchantId provided by icepay	Integer
Checksum	Is the SHA1 checksum generated using MerchantID,SecretCode,Timestamp,PaymentID in the same order.	String
Timestamp	Is the timestamp of the response time.	String
PaymentID	Is the PaymentID of the captured payment.	String

7.1.1 CODE EXAMPLE

C#:

```
APCaptureClient capture = new APCaptureClient();
            int MerchantID = "MerchantID";
            try
            {
               capture.CaptureFull(MerchantID, this.CalculateChecksum(MerchantID,
"timestamp", PaymentID)**, "timestamp", paymentID);
                }
                catch (Exception e)
                {
                    HandleException(e);
                }
            }
```

Note: The code is just an example it will not work if copy/pasted.
**When making the checksum if is desired to do not include the amount and the currency of the payment use an empty string and a zero.

8 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 merchant 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.

8.1 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.

8.2 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.

8.3 SUPPORTED PAYMENT METHODS

Not all payment methods support refunding. Refunding is limited to iDEAL, Credit Card and Direct Debit.

8.4 TEST AND LIVE TRANSACTIONS

Currently, all refund web methods accept live transactions **ONLY**. This means that when writing your client application, you must work and test with live transactions. So please do not forget to cancel the refund requests once you are done with it. We recommend you to do this not around 8am UTC+0 because our refunding queue will be processed at that time.

The status of the merchant itself does not matter as test merchants can have live transactions, and vice versa.

8.5 REQUESTREFUND

The *RequestRefund* web method allows you to initiate a refund request for a payment. You can request the entire amount to be refunded or just a part of it. If you request only a partial amount to be refunded then you are allowed to perform refund requests for the same payment until you have reached its full amount. After that you cannot request refunds anymore for that payment.

8.5.1 INPUT

The required input parameters are displayed below.

Parameter	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
PaymentID	The ID of the payment.	Integer
RefundAmount	The amount to be refunded (in cents)	Integer
RefundCurrency	The currency of the refund. **Remarks:** This currency must match the currency of the payment.	String

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|2009-06-09T01:30:00Z|1234567|1000|EUR
```

8.5.2 RESPONSE

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
RefundID	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.	Integer
PaymentID	This is the payment for which you requested a refund.	Integer
RefundAmount	The requested refund amount specified in the request	Integer
RemainingRefundAmount	The remaining amount that you can still request a refund for.	Integer
RefundCurrency	The requested refund currency specified in the request	String

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
```

Example

```
secret|12345|2009-06-09T01:30:00Z|12345|1234567|1000|0|EUR
```

8.6 CANCELREFUND

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

8.6.1 INPUT

The required input parameters are displayed below.

Parameter	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
RefundID	This is the RefundID that is returned by the RequestRefund web method upon a successful invocation.	Integer
PaymentID	This is the PaymentID of the transaction for which you requested the refund in the first place	Integer

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|2009-06-09T01:30:00Z|12345|1234567
```

8.6.2 RESPONSE

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

Member	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 message digest to verify the authenticity of the response.	String

Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
Success	This field will contain the value 'Y' if the refund request was cancelled successfully.	String

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|2009-06-09T01:30:00Z|false
```

8.7 GETPAYMENTREFUNDS

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

8.7.1 INPUT

The required input parameters are displayed below.

Parameter	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
PaymentID	This is the PaymentID of the transaction for which you requested the refund in the first place	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|2009-06-09T01:30:00Z|12345|1234567
```

8.7.2 RESPONSE

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

Parameter	Description	Data type
MerchantID	This is the merchant ID that is part of your API key. You can create the API key in your ICEPAY account.	Integer
Checksum	This is the SHA1 digest of all the members.	String
Timestamp	This is the current UTC time that must have the following format: yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	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|2009-06-09T01:30:00Z|1234567|12345|2009-06-09T01:30:00Z|1000|EUR|PENDING|12350|2009-06-09T01:31:00Z|1200|EUR|PROCESSING
```

8.7.2.1 REFUND OBJECT

This object contains information about a payment method.

Parameter	Description	Data type
RefundID	This is a unique ID that is assigned to your refund request	Integer
RefundAmount	This is the amount of the refund	Integer
RefundCurrency	This is the currency of the refund	String
DateCreated	This value indicates when the refund request was created (in UTC+0): yyyy-mm-ddThh:mm:ssZ Example: 2009-06-09T01:30:00Z	String
Status	A refund can have one of the following status: PENDING Refund is placed in the queue PROCESSING Refund sent to financial institution for refund COMPLETED Refund processed by financial institution REFUSED Refund cannot be processed	String

9 EXCEPTIONS

9.1 ERROR CODES 0000 – 0016

The following error codes are generic error codes:

ERR_0000	**Internal server error: XXX** An unexpected error occurred. Please contact support@icepay.eu to resolve the issue.
ERR_0001	**Request is missing** You did not provide a request object with the web method
ERR_0002	**Please provide a valid 'MerchantID' member** You must provide a valid numeric 'MerchantID' member
ERR_0003	**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.
ERR_0004	**Please provide the IP address of your end-user** You forgot to include the IP address of the end-user.
ERR_0005	**Merchant 'XXXXX' is disabled** Your API key is disabled. Please contact your account manager regarding this issue.
ERR_0006	**Merchant 'XXXXX' was not found** Unknown merchant ID
ERR_0007	**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.
ERR_0008	**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.
ERR_0009	**Payment with ID 'XXX' not found**
ERR_0010	**'XXX' parameter must be at least YY characters** The length of the provided parameter must be at least YY characters long.
ERR_0011	**'XXX' parameter may not exceed YY characters** The length of the parameter has exceeded the maximum allowed length YY.
ERR_0012	**'XXX' is an invalid payment method**
ERR_0013	**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
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 (accidently) specified a payment ID that does not belong to the specified merchant.

9.2 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**

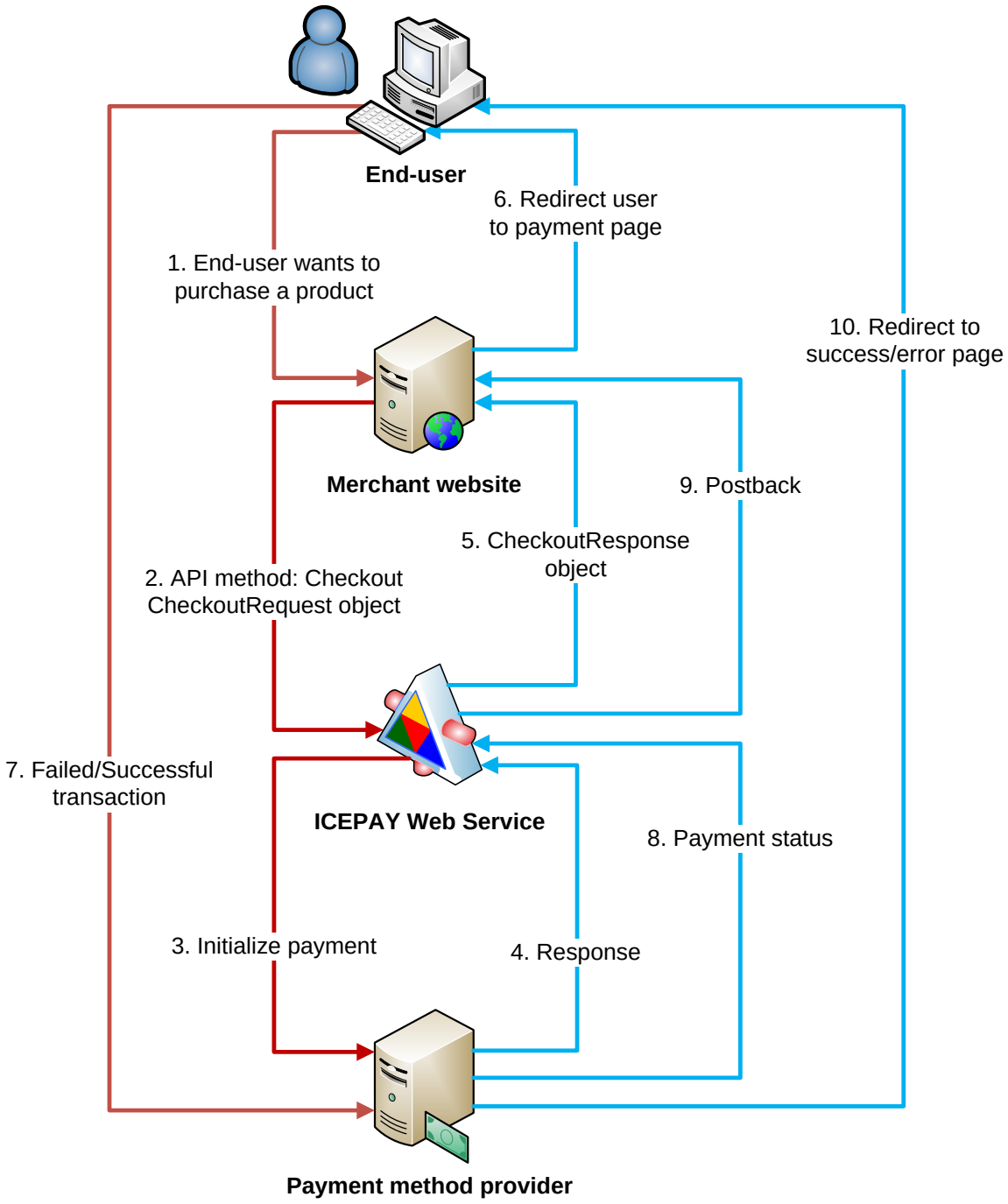
9.3 ERROR CODES 2000 – 2007

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

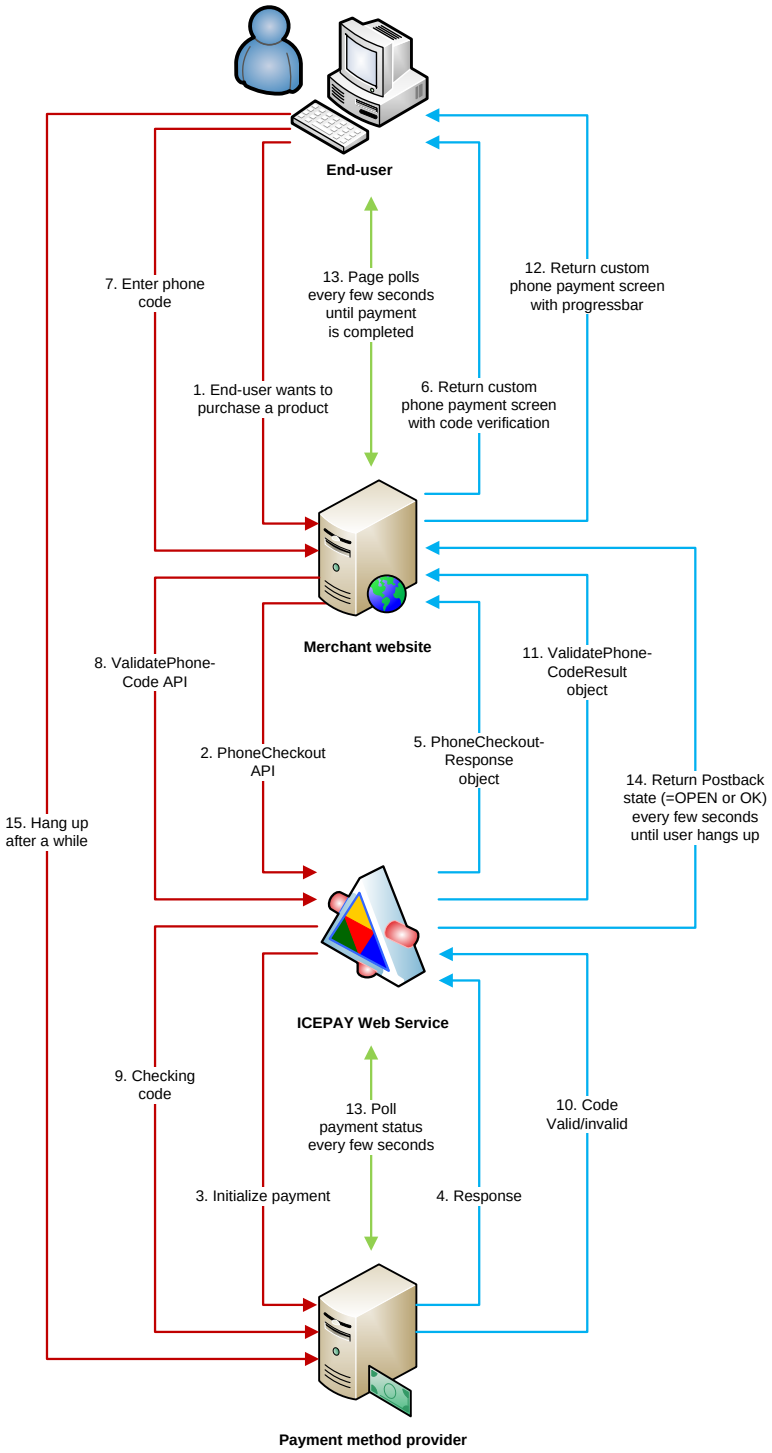
ERR_2000	**Merchant not granted to use Refund Web Service** The merchant is not granted access to use the Refund Web Service. In order to enable the refund web service for your merchant, please log into your ICEPAY account to configure your merchants.
ERR_2001	**Invalid RefundID 'XXX'** The RefundID 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**
ERR_2006	**You can only refund payments with the status 'OK'**

ERR_2007	RefundID does not belong to PaymentID

10 APPENDIX A: FLOW OF THE BASIC INTEGRATION APPROACH USING CHECKOUT WEB METHOD



APPENDIX B: Flow of seamless integration using PhoneCheckout web method



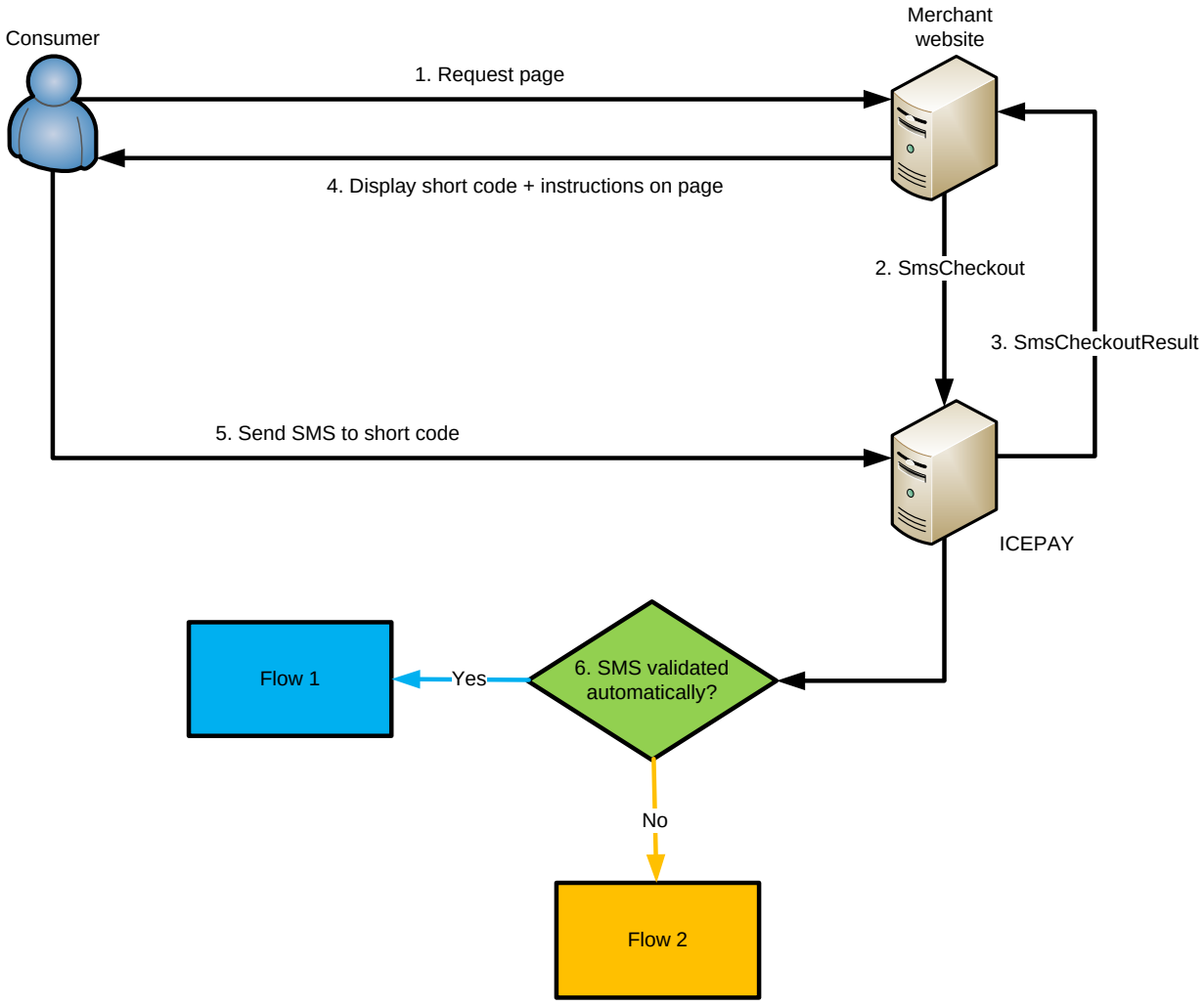
11 APPENDIX C: CERTIFICATE

You *can* improve the connection speed/performance between your server and the ICEPAY server by importing our public SSL certificate into your local storage. You can do this by copying the text below and save it into a plain text file with the extension **.cer**.

This public SSL certificate is valid until **2014/04/26** (YYYY/MM/DD)

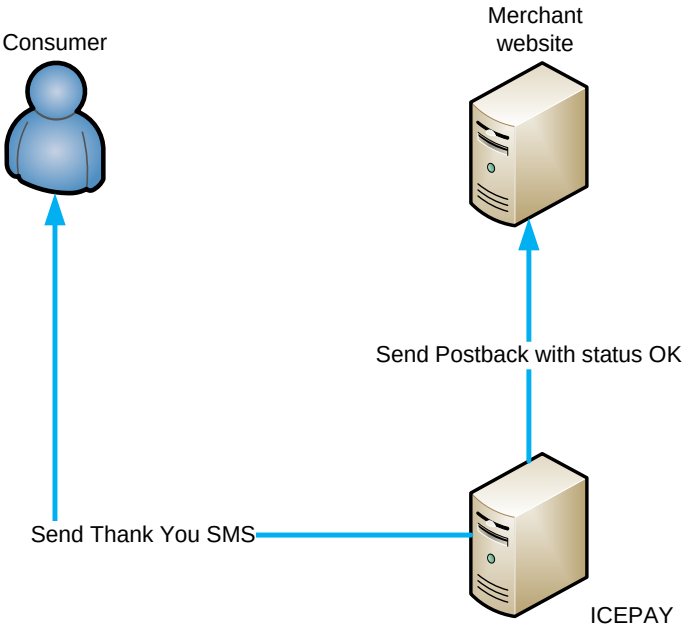
-----BEGIN CERTIFICATE-----
MIIEQDCCAyigAwIBAgIQUQkwyYF+fB0ARLEEDBUfbjANBgkqhkiG9w0BAQUFADBe
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMVGhhd3RlLCBJbmMuMR0wGwYDVQQLExRE
b21haW4gVmFsaWRhdGVkIFNTTDEZMBcGA1UEAxMQVGhhd3RlIERWIFNTTCBDQTAe
Fw0xMjA0MjUwMDAwMDBaFw0xNDA0MjUyMzU5NTlaMIG2MRswGQYDVQQKExJjb25u
ZWN0LmljZXBheS5jb20xOzA5BgNVBAsTMkdvIHRvIGh0dHBzOi8vd3d3LnRoYXd0
ZS5jb20vcmVwb3NpdG9yeS9pbmRleC5odG1sMSIwIAYDVQQLExlUaGF3dGUgU1NM
MTIzIGNlcnRpZmljYXRlMRkwFwYDVQQLExBEb21haW4gVmFsaWRhdGVkMRswGQYD
VQQDExJjb25uZWN0LmljZXBheS5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
ggEKAoIBAQDdEW3lwWJlqQZYes8+uK7rjmHz+fVtZ4TkMz473UCCkDywsptmTDzj
yQ3fLI13lfBR3hEVCPrfmiVDAi68ZPTKtcziSB5Wz8UKuHysyetDS8vFx8daG3CM
d3lCzjG8jEnHkp1vX0QlzTH8/leGP6rYmFZVtSWsi1VA37oZLI5kYnpcdpSQ7MYS
fusFbMIUCNAF7/dCBxuJioDs/l3rePW5/jtJaKbawyJphp7rbN3nnD6uv5CZne3/
yfxgCmkyqWMqU8V2uAvsdvm9YR3tvvQySH13K/u9FQ7OJeJiFk4guMdX57+VqTNA
zm83n8/fa4/JMXaqQly7YAdKWVro9FCVAgMBAAGjgaAwgZ0wDAYDVR0TAQH/BAIw
ADA6BgNVHR8EMzAxMC+gLaArhilodHRwOi8vc3ZyLWR2LWNybC50aGF3dGUuY29t
L1RoYXd0ZURWLmNybDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwMgYI
KwYBBQUHAQEEJjAkMCIGCCsGAQUFBzABhhZodHRwOi8vb2NzcC50aGF3dGUuY29t
MA0GCSqGSIb3DQEBBQUAA4IBAQCO1rkN1EoV7D1b8d862nACTMqynAHFphzDJ5KX
2LIQipKMmja5RF5J8dITNTjncvyTt1omgklysjJNktlMYc48fxndYTJSZ+dY0cd7
oC8+UnGH2aJ2f5uk8qm/OZsbD6Bi2jjxKpp2NQKOwGB1oKeqEpmiKAdSVZvPk1H1
glqL9XQR+3c6rrB79rDBJmm6JU/Q3EtRpltmBprcwh2BQZqOa0+LruKhJRQWuExl
YoWKQRmbprzgrmHQWKoTyn5boq+Ud8sbZ1K32oPP5PrKOwaCFGjFAvvbyvaUiCUt
Ys/DvlTzrleTyzLGnNl7MQRJTWEfuMWMJwmZoAf3+PNys3Xt
-----END CERTIFICATE-----

12 APPENDIX D: SMS FLOW (SEAMLESS INTEGRATION)

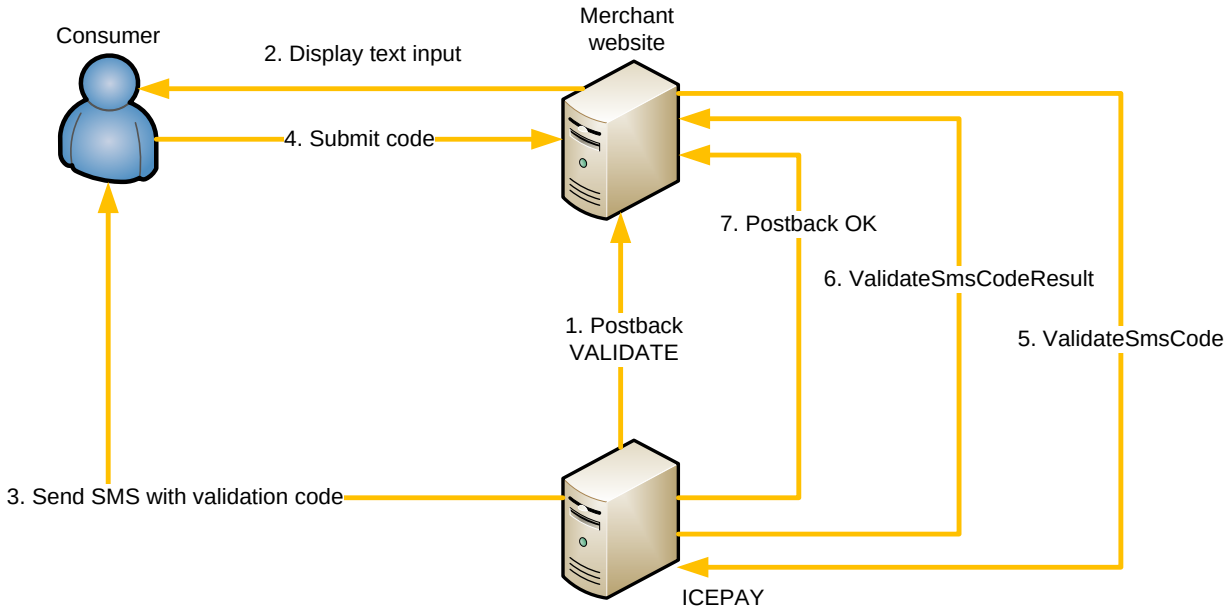


Please take a look at the next page for a more detailed flow on Flow 1 and Flow 2.

12.1 FLOW 1: DEFAULT END-USER FLOW



12.2 FLOW 2: END-USER VALIDATION FLOW



13 APPENDIX E: CHECKOUTEXTENDED XML

13.1 XSD

```
<?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>
        <xs:element name="Consumer">
          <xs:complexType>
            <xs:sequence>
              <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:simpleType>
              </xs:element>
              <xs:element name="Phone">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:minLength value="1" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>

        <xs:element name="Addresses">
          <xs:complexType>
            <xs:sequence minOccurs="2" maxOccurs="2">
              <xs:element name="Address">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Initials">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:minLength value="1" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="Prefix" type="xs:string" />
                    <xs:element name="LastName">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:minLength value="1" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="Street" type="xs:string" />
                    <xs:element name="HouseNumber">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:minLength value="1" />
                        </xs:restriction>
                      </xs:simpleType>
                    </xs:element>
                    <xs:element name="HouseNumberAddition" type="xs:string" />
                    <xs:element name="ZipCode">
                      <xs:simpleType>
                        <xs:restriction base="xs:string">
                          <xs:minLength value="1" />
                        </xs:restriction>
```

```
                </xs:simpleType>
              </xs:element>
              <xs:element name="City">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:minLength value="1" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <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:sequence>
    </xs:complexType>
    <xs:unique name="id">
      <xs:selector xpath="Address"/>
      <xs:field xpath="@id"/>
    </xs:unique>
  </xs:element>

  <xs:element name="Products">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element name="Product">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ProductID" type="xs:string" />
              <xs:element name="ProductName" type="xs:string" />
              <xs:element name="Description" type="xs:string" />
              <xs:element name="Quantity" type="xs:int" />
              <xs:element name="UnitPrice" type="xs:int" />
              <xs:element name="VATCategory">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="standard|reduced-low|reduced-middle|zero|exempt" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

13.2 EXAMPLE

```
<?xml version="1.0" encoding="utf-8" ?>
<Order>
  <Consumer>
    <Email>john.doe@email.com</Email>
    <Phone>06-12345678</Phone>
  </Consumer>

  <Addresses>
```

```
    <Address id="billing">
      <Initials>J.</Initials>
      <Prefix></Prefix>
      <LastName>Doe</LastName>
      <Street>Nachtwachtlaan</Street>
      <HouseNumber>20</HouseNumber>
      <HouseNumberAddition></HouseNumberAddition>
      <ZipCode>1058 EA</ZipCode>
      <City>Amsterdam</City>
      <Country>NL</Country>
    </Address>
    <Address id="shipping">
      <Initials>R.</Initials>
      <Prefix></Prefix>
      <LastName>Roe</LastName>
      <Street>Wassenaarseweg</Street>
      <HouseNumber>7</HouseNumber>
      <HouseNumberAddition></HouseNumberAddition>
      <ZipCode>2596 CD</ZipCode>
      <City>The Hague</City>
      <Country>NL</Country>
    </Address>
  </Addresses>

  <Products>
    <Product>
      <ProductID>123456789</ProductID>
      <ProductName>Smartphone</ProductName>
      <Description></Description>
      <Quantity>1</Quantity>
      <UnitPrice>59995</UnitPrice>
      <VATCategory>standard</VATCategory>
    </Product>
  </Products>
</Order>
```