API 2.x

Documentation & Manual

Index

1 Introduction

ICEPAY offers different API's to use in your projects. The API 2.x is the latest addition. Compared to API 1.0, the ICEPAY API 2.x is easier to integrate in your projects and it offers more flexibility in development. The API 2.x offers a wide range of possibilities to manage the payment methods in your webshop even better than with API 1.0.

In the end, this will lead to a better user experience for your customer and a higher conversion in your webshop.

1.1 Important changes

IMPORTANT!

Regarding version 1: The 1.x versions of the PHP API are not compatible with the 2.x version of the PHP API.

Regarding version 2.0.x: The 2.1 version has been refactored due to the common libraries for both the basicmode and webservices implementation.

Starting a transaction and configuring the logger has been altered. It is required to change your implementation accordingly. The manual covers these changes and the sample scripts have been updated.

1.2 Document revisions

Date	Author	Document version	API version	Comment
6-8-2013	Wouter van Tilburg	1.0.7	2.3.0	Added VaultCheckout and AutoCheckout
8-1-2013	Wouter van Tilburg	1.0.6	2.2.0	Added Extended Checkout
16-10-2012	Wouter van Tilburg	1.0.5	2.1.2	Added whitelist IP functions documentation
23-07-2012	Olaf Abbenhuis	1.0.4	2.1.0	Added payment object documentation
23-07-2012	Olaf Abbenhuis	1.0.3	2.1.0	Added webservices documentation
27-02-2012	Olaf Abbenhuis	1.0.2	2 – 1.0.1	Added dynamic success and error URL Added getReadableName function Corrected some document errors.
28-12-2011	Olaf Abbenhuis	1.0.1	2 – 1.0.0	Added Postback method information and the Postback object chapter.
9-9-2011	Olaf Abbenhuis	1.0.0	2 – 1.0.0	Official document release

2 Sample Scripts

Please read the sample scripts provided with the API 2.x. Within these samples we have put comments to further understand the integration process. These sample scripts are used for training purposes only and should not be used in a live environment. Ensure these files are not uploaded and used in your project.

The software is provided “as is”, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use of other dealings in the software.

2.1 Basicmode sample scripts

Listed in chronologic sensible order

Name	Description	Classes used
sample_basicmode.php	Start a basicmode transaction	Icepay_Basicmode
sample_filter.php	Filter all the payment methods by certain payment details, such as amount and currency and start a transaction with the selected payment method.	Icepay_Api_Basic
sample_ideal.php	Start an iDeal transaction	Icepay_Paymentmethod_Ideal
sample_postback.php	Catch the ICEPAY transaction data and do some order handling, such as sending an e-mail.	Icepay_Postback
sample_result.php	The return script for the enduser, some data is caught and used to present a message to the enduser.	Icepay_Result

2.2 Webservice sample scripts

Listed in chronologic sensible order. Note that the sample scripts “**sample_postback.php**” and “**sample_postback.php**” also apply to the webservices.

Name	Description	Class used
sample_webservice_startpayment.php	Start an iDeal transaction using the webservices and redirect the end user to the payment screen.	Icepay_Api_Webservice
sample_webservice_start_extended_payment	Start an Afterpay transaction using the webservices and redirect the end user to the payment screen.	Icepay_Api_Webservice
sample_webservice_start_sms	Start an SMS transaction	Icepay_Api_Webservice
sample_webservice_getpaymentmethods	Retrieve all the payment method data of a merchant account and saves it to a local file.	Icepay_Api_Webservice
sample_webservice_filterpaymentmethod	Read the local file mentioned above and filters the payment methods based on payment data such as amount and currency.	Icepay_Api_Webservice
sample_webservice_getpayment	Get information on a payment.	Icepay_Api_Webservice
sample_webservice_reporting	Retrieve transaction statistics, a special pincode is required.	Icepay_Api_Webservice
sample_webservice_refund	Request a refund of a transaction through the webservices.	Icepay_Api_Webservice

3 Basic mode or Web services

3.1 Overview

Basic mode (Icepay_Api_Basic/Icepay_Basicmode) and the web services (Icepay_Api_Webservice) are regarded as two separate ways of implementation. Some functionality of these two implementation options will seem similar, but there are some substantial differences. However, the API 2.x combines these two gracefully to offer the best of both worlds.

Consider that you have more freedom in the design of your application when using the Web services Implementation.

3.2 Starting a payment

Within the API there are 4 ways to start a payment at ICEPAY. Here is a clear overview (From easy to hard integration):

Name	Description	Class used
Basicmode	Start a payment without a specific payment method in mind. This will display a payment method selection screen known as "Basicmode".	Icepay_Basicmode
API Basic Payment method	Start a specific payment method transaction and forward the customer to the page of the payment method directly.	Icepay_Paymentmethod extended classes, for example: Icepay_Paymentmethod_Ideal
API Webservice Payment method	Start a specific payment method transaction and forward the customer to the page of the payment method directly.	Icepay_Api_Webservice
API Webservice Extended Checkout (Not advised if you do not plan on using Afterpay)	Start a specific payment method transaction with extra order information (Required by some paymentmethods such as Afterpay) and forward the customer to the page of the payment method directly	Icepay_Api_Webservice

As you can see, the last 3 options “Api Basic Payment method”, “Api web service payment method” and “API Web service Extended Checkout” are similar. What are the differences?

The web services were added in 2.1 of the API and are actually the preferred method of starting a transaction. The web services platform is behind our load balanced server setup, meaning the payment request will always be handled by the least busy server.

The web services also offer more data at the start of a transaction.

Currently we offer two ways to start a payment in web services, normal checkout and extended checkout. The main difference is that the extended checkout requires a lot more order information, such as product information, vat categories and consumer information. Some payment methods like Afterpay require this extra information and therefore you can only start an Afterpay payment with the extended checkout method. Note however that every payment method works with the extended checkout.

Important: If you plan to use web services but not Afterpay, we do not advise using the extended checkout; instead use the normal web service checkout (See 17.2).

Below you can check what is returned on the payment request for each class:

Action	API Basic Payment method	API Webservice Payment method
Starting a payment	PaymentScreenURL	EndUserIP
		PaymentID
		PaymentScreenURL
		ProviderTransactionID
		TestMode

A lot more data is returned using the webservices, as you can see in the overview. Especially the PaymentID and TestMode values are useful for your application.

For more detailed information about the returned values by the webservices, please read our Webservices documentation:

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

3.3 Updating the status of your order (Postback script)

Use the Icepay_Postback class to handle the transactional postbacks. This is the best and fastest way of updating the payment status of the order in your application.

However, the webservices offer an additional call to ICEPAY to retrieve the real-time transaction information. Read the Payment Service for more information.

3.4 Handling the returning customer (Result script)

Use the Icepay_Result class to handle the returning customer on the Success (URLCompleted) and Error (URLError) URLs.

4 Static class: Icepay_Api_Logger

The API contains built-in loggers, but these are switched off by default. It is recommended to do some logging, especially during the development phase of your application.

To use the logging, the class must be configured before calling any other classes of the API. This will enable the API classes to read the logger configuration.

Using the class:

```
require_once('icepay_api_basic.php');
/* also available from: require_once('icepay_api_webservice.php'); */
$logger = Icepay_Api_Logger::getInstance();
```

4.1 Examples

Using the log class on its own:

```
$logger = Icepay_Api_Logger::getInstance();

/* Basic configuration */
$logger->enableLogging()
      ->setLoggingLevel(Icepay_Api_Logger::LEVEL_ALL)
      ->logToScreen();

/* Enable logging to a file */
$logger->logToFile()
      ->setLoggingDirectory(realpath("../logs"))
      ->setLoggingFile("log.txt");

/* log something */
$logger->log("hello world");
```

Using the log class within ICEPAY classes.

```
/* Configure the Log class first */
$logger = Icepay_Api_Logger::getInstance();

$logger->enableLogging()
       ->setLoggingLevel(Icepay_Api_Logger::LEVEL_ALL)
       ->logToFile();

/* Start using the ICEPAY classes. */
$payment = Icepay_Basicmode::getInstance();

$payment->setMerchantID(MERCHANTID)
         ->setSecretCode(SECRETCODE);
/* etc… */
```

4.2 Methods

Method	Params	Default value	Description
getInstance()			Create an Icepay_Api_Logger instance
enableLogging	True/False (Boolean)	True	Enable logging
setLoggingLevel	Level (int)	Icepay_Api_Logger:: LEVEL_ERRORS_AND_TRANSACTION	Configure the log level
setLoggingDirectory	Directory (string)	“logs”	Set the directory where log files are placed.
setLoggingFile	File name (string)	“log.txt”	Set the filename where the logger adds its lines.
logToFile	True/False (Boolean)	True	Enable logging to a file on the server
logToScreen	True/False (Boolean)	True	Enables printing log output to screen (echo)
logToFunction	Class name (String)		Use a logger class already existing in your PHP project
	Function name (String)		Use a logger function already existing in your PHP project
	True/False (Boolean)	True	Enables logging to your own logger function. Note that a class requires to be hooked first using the hookLogClass method.
log	Line of text (String)		Logs the string to all of the configured options.
	Log level type	Icepay_Api_Logger::NOTICE	Set the type of level

4.3 Logging levels

Several loggers are built within the API. To prevent log cluttering, it is possible to select which logs should be executed.

Use the setLoggingLevel method to set the desired level.

By default, only **Errors and Transactions** are logged.

```
$logger = new Icepay_Api_Logger ();
$logger->enableLogging()
            ->logToScreen()
            ->setLoggingLevel(Icepay_Api_Logger::LEVEL_ALL);
```

Level	API will only log
Icepay_Api_Logger::LEVEL_ALL	NOTICE,ERROR,TRANSACTION
Icepay_Api_Logger:: LEVEL_TRANSACTION	TRANSACTION
Icepay_Api_Logger:: LEVEL_ERRORS	ERROR
Icepay_Api_Logger:: LEVEL_ERRORS_AND_TRANSACTION	ERROR,TRANSACTION

An example of using the logger with a log level type:

```
$logger = Icepay_Api_Logger::getInstance();
$logger->logToScreen()
            ->setLoggingLevel(Icepay_Api_Logger::LEVEL_ALL);

$logger->log(“test”,Icepay_Api_Logger::NOTICE);
```

5 Class: Icepay_PaymentObject

This class contains all the information regarding the payment which will be made using either the BasicMode or Webservices.

Using the class:

```
require_once('icepay_api_basic.php');
/* also available from: require_once('icepay_api_webservice.php'); */

$paymentObj = new Icepay_PaymentObject();
```

5.1 Class setters

Use the following methods to set data, these are chainable.

Method	Required	Param	Default value	Description
setCountry	No	Country (String)		Set ISO 3166-1-alpha-2 Country Code. If omitted, the “00” country code is used.
setCurrency	Yes	Currency (String)		Set ISO 4217 Currency Code
setLanguage	Yes	Language (String)		Set ISO 639-1 Language Code
setAmount	Yes	Amount (uint)		Set amount in cents
setOrderID	Yes*	Order ID (String)		The Order ID is the **unique** ID linked to your order and ICEPAY transaction. Although this field is not required by Basicmode, it is recommended to make use of this ID to handle postbacks and link them to the correct order. If omitted the Order ID will be auto generated.
setReference	No	Reference (String)		The Reference field will be visible in your ICEPAY transactions overview.
setDescription	No	Description (String)		The description field can be used to display information on the customer

5.2 Example

```
$paymentObj = new Icepay_PaymentObject();

$paymentObj->setAmount(1000)
        ->setCountry("NL")
        ->setLanguage("NL")
        ->setReference("My Sample Website")
        ->setDescription("My Sample Payment")
        ->setCurrency("EUR")
        ->setOrderID(101);
```

5.3 Payment method specific class methods

The following methods are discussed in 7.6 and are required to use a specific payment method. These are also used by the webservices.

Method	Required for Basic API	Required for Webservices
setPaymentMethod	Yes	Yes
setIssuer	Optional	Optional

6 Class: Icepay_Basicmode

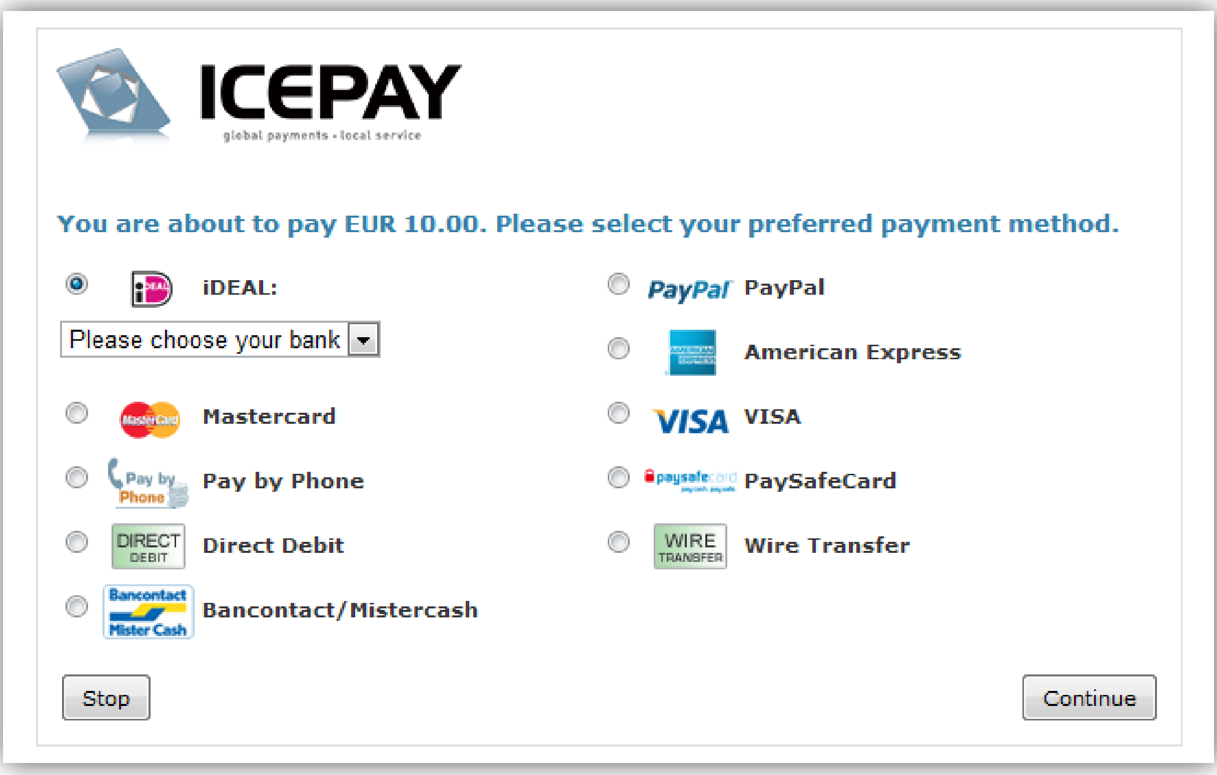
This class functions to start a transaction and forward the end user to the Basicmode payment screen.

Basicmode is an ICEPAY payment screen with a collection of payment methods based on the submitted parameters.

Using the class:

```
require_once('icepay_api_basic.php');
$basicmode = Icepay_Basicmode::getInstance();
```

Example of the Basicmode page:



You are about to pay EUR 10.00. Please select your preferred payment method.

[✓ iDEAL:] Please choose your bank
PayPal
American Express
Mastercard
VISA
Pay by Phone
PaySafeCard
Direct Debit
Wire Transfer
Bancontact/Mistercash

Note that it is possible to upload your own logo. (Log in to the ICEPAY website and go to your merchant settings.)

6.1 Example

```
$basicmode = Icepay_Basicmode::getInstance();

$basicmode->setMerchantID(10000)
   ->setSecretCode("MySecretCode")
   ->validatePayment($paymentObj);

$url = $basicmode->getURL();

echo(sprintf("<a href=\"%s\">%s</a>",$url,$url));
```

6.2 Class setters

Use the following methods to set data, these are chainable.

Method	Required	Param	Default value	Description
setMerchantID	Yes	MerchantID (uint)		
setSecretCode	Yes	SecretCode (String)		
setProtocol	No	Protocol (“https”/”http”)	“https”	It is recommended to use the https protocol. For development purposes it is optional to use the http protocol.
validatePayment	Yes	Icepay_PaymentObject		Pass the payment you want to process via this method.

6.3 Dynamic URLs

It is possible to set the URLs for success and error through the API and override the URLs set in the merchant settings. These are not required, but if one is used, both need to be set. It is not possible to set the Postback URL.

Method	Required	Param	Description
setSuccessURL	No	URL (String)	The URL used to redirect the customer to when the transaction is completed
setErrorURL	No	URL (String)	The URL used to redirect the customer to when there's an error during the transaction

6.4 Class getters

Use the following methods to get data.

Method	Returns	Description
getURL	Returns the Payment URL (String)	Uses all the set data to generate the URL. Uses PHP CURL to connect to ICEPAY
getVersion	Returns the version of the basicmode class (String)	

7 Basic API: Icepay_Paymentmethod classes

The payment method classes are located within the payment methods subfolder, all of them extend the Icepay_Paymentmethod class.

Using a payment method class the end-user can be redirected directly to the payment screen of the selected payment method.

Using the class:

```
require_once('icepay_api_basic.php');
```

7.1 Loading the classes

Create an instance of the Icepay_Api_Basic class and read all the payment method classes for later use.

Method	Parameter	Description
getInstance		Create an instance of the Icepay_Api_Basic Class
setPaymentMethodsFolder	Directory name (String)	Set the folder where the payment method classes are stored
readFolder		Read all the paymentmethod classes in the payment methods folder

Example:

```
Icepay_Api_Basic::getInstance()->readFolder();
```

7.2 Retrieving the payment method data

Retrieve the payment methods list using the following method:

Method	Parameter	Returns
getArray		The paymentmethods classes
getClassByPaymentMethodCode	ICEPAY payment method code	The Icepay_Paymentmethod class

Example:

```
$paymentMethods = Icepay_Api_Basic::getInstance()->readFolder()->getArray();
```

7.3 Filtering the Payment Method data

If some order data is known, such as the currency, it is possible to filter the list of payment methods to just the applicable ones.

Method	Param	Description
prepareFiltering		Load the classes, needs to be called before any filtering can be done
filterByCurrency	Set ISO 4217 Currency Code	Filters the paymentMethods array on currency
filterByCountry	Set ISO 3166-1-alpha-2 Country Code	Filters the paymentMethods array on country
filterByLanguage	Set ISO 639-1 Language Code	Filters the paymentMethods array on language
filterByAmount	Amount (int)	Filters the paymentMethods array on your order total

Example:

```
$paymentMethods = Icepay_Api_Basic::getInstance()->readFolder()
            ->prepareFiltering()
            ->filterByCurrency("EUR")
            ->filterByAmount(10000)
            ->getArray();
```

7.4 Loading and showing a single payment method

It is possible to load the payment method class specifically, or to load it programmatically, using the payment method array mentioned in the previous chapter.

Using the class:

```
require_once('icepay_api_basic.php');
$ideal = new Icepay_Paymentmethod_Ideal();

/* Or if the $paymentMethods array is filled:
 * $ideal = new $paymentMethods["ideal"]();
*/
```

7.5 Payment method getters

Use these methods to get information regarding the loaded payment method.

Method	Returns	Note
getCode	The Payment method code (String)	
getReadableName	Payment method name (String)	The name of the payment method in English
getSupportedIssuers	Array of issuers	Displays the issuer codes
getSupportedCountries	Array of countries	Note that 00 means all countries
getSupportedCurrency	Array of currencies	Note that 00 means all currencies
getSupportedLanguages	Array of languages	Note that 00 means all languages
getSupportedAmountRange	Array of minimum and maximum amount values	

Example:

```
$version = $ideal->getCode();
```

7.6 Required additions to the Icepay_PaymentObject method

In order to start a payment using a selected payment method, the payment method code must be passed to the Icepay_PaymentObject class. If a payment method has multiple issuers, the selected issued must also be set. Issuers can be read from the Icepay_Paymentmethod class using the getSupportedIssuers() function. If just one issuer exists, you do not have to use the setIssuer method.

Method	Required	Param	Description
setPaymentMethod	Yes	ICEPAY Payment method code	Loads the paymentmethod within the API.
setIssuer	No	ICEPAY Issuer code	Set the Issuer

Sample:

```
$ideal = new Icepay_Paymentmethod_Ideal();
$issuers = $ideal->getSupportedIssuers();

$paymentObj = new Icepay_PaymentObject();

$paymentObj
        ->setPaymentMethod($ideal->getCode())
        ->setIssuer($issuers[0])
        ->setAmount(1000)
        ->setCountry("NL")
        ->setLanguage("NL")
        ->setReference("My Sample Website")
        ->setDescription("My Sample Payment")
        ->setCurrency("EUR")
        ->setOrderID(101);
```

During validatePayment of the Icepay_Basicmode class the set parameters will be checked against the supported parameters of the payment method. Incompatible values will result in catchable exceptions.

7.7 Additional configuration

The Icepay_Basicmode class offers an easy way to use the webservices for checkout. This is only possible when using a specific payment method.

Use the following method to further configure the API:

Method	Required	Param	Description
useWebservices	No	Boolean	Use the webservices to start the payment

Example:

```
$basicmode = Icepay_Basicmode::getInstance();

$basicmode->setMerchantID(10000)
   ->setSecretCode("MySecretCode")
   ->validatePayment($paymentObj)
   ->useWebservices();

$url = $basicmode->getURL();

echo(sprintf("<a href=\"%s\">%s</a>",$url,$url));
```

8 Static class: Icepay_Parameter_Validation – Optional validation

The setters make use of the Icepay_Parameter_Validation class in order to check the basic requirements, such as character length and contents.

It’s possible to use this static class in your project to do some basic field validation. A useful example would be if you’re writing a module where the merchant requires to enter their merchant ID.

Method	Parameter	Returns
Icepay_Parameter_Validation::amount	Integer	Boolean
Icepay_Parameter_Validation::country	ISO 3166-1-alpha-2 Country Code	Boolean
Icepay_Parameter_Validation::currency	ISO 4217 Currency Code	Boolean
Icepay_Parameter_Validation::language	ISO 639-1 Language Code	Boolean
Icepay_Parameter_Validation::merchantID	5 character numeric value	Boolean
Icepay_Parameter_Validation::pinCode	? character numeric value	Boolean
Icepay_Parameter_Validation::secretCode	40 character string	Boolean

Using the class:

```
require_once('icepay_api_basic.php');
/* also available from: require_once('icepay_api_webservice.php'); */

$country = “ESP”; // Spain
$check = Icepay_Parameter_Validation::country($country); // Returns false

$country = “ES”; // Spain
$check = Icepay_Parameter_Validation::country($country); // Returns true
```

9 Static class: Icepay_StatusCode – Status code handling

It is recommended to save and retrieve the status of the payment with ICEPAY status codes. To enable easy access to these status codes we have created global variables, accessible from any script, as long as the API file is included in your project.

It is recommended to create a database table parallel to your order database table where the status code handling is being done. When you do this, you can add extra status code verification. This increases protection against malicious attacks.

ATTENTION: Do not update any status from the result (success/error) pages. Instead, use the status codes solely to inform the customer about the progress of their payment.

Use the postback script to update your order statuses.

Statuscode constants
Icepay_StatusCode::OPEN
Icepay_StatusCode::ERROR
Icepay_StatusCode::SUCCESS
Icepay_StatusCode::REFUND
Icepay_StatusCode::CHARGEBACK

See the postback and result documentation for example usage.

10 Class: Icepay_Result - Handling the error and success page

The Icepay_Result class handles the returning customer, intended for the error and success URL. Note that no order logic should be handled by this class, use it to notify the customer and close/open the cart in your application.

Using the class:

```
require_once('icepay_api_basic.php');
/* also available from: require_once('icepay_api_webservice.php'); */
$api = new Icepay_Result();
```

10.1 Methods

Method	Parameter	Returns	Description
setMerchantID	MerchantID (uint)		
setSecretCode	SecretCode (String)		
validate		True/False	Checks whether the data is valid ICEPAY data and stores it in the class.
getStatus	True/False (Boolean)	False (default): Icepay_StatusCode status True: String	Retrieve the status of the transaction. When a true is submitted through the getStatus, the status will include the status and statuscode in a readable string. Only use this feature for informing the customer. This data is only available after calling the *validate* method.
getOrderID		String	Get just the orderID from the ICEPAY data.This data is only available after calling the *validate* method.
getResultData		Object: status, statusCode, merchant, orderID, paymentID, reference, transactionID, checksum	Get all the ICEPAY data in an object, note that the GET parameters are named differently. This data is only available after calling the *validate* method.

10.2 Example

```
$icepay = new Icepay_Result();
$icepay->setMerchantID(10000)
        ->setSecretCode("mySecretCode");

try {
    if($icepay->validate()){

        switch ($icepay->getStatus()){
            case Icepay_StatusCode::OPEN:
                // Close the cart
                echo("Thank you, awaiting payment verification.");
                break;
            case Icepay_StatusCode::SUCCESS:
                // Close the cart
                echo("Thank you for your purchase. The payment has been received.");
                break;
            case Icepay_StatusCode::ERROR:
                //Redirect to cart
                echo(sprintf("An error occurred: %s",$icepay->getStatus(true)));
                break;
        }

    } else die ("Unable to validate data");
} catch (Exception $e){
    echo($e->getMessage());
}
```

11 Class: Icepay_Postback - Handling the ICEPAY transaction

Using the class:

```
require_once('icepay_api_basic.php');
/* also available from: require_once('icepay_api_webservice.php'); */

$api = new Icepay_Postback();
```

11.1 Methods

Use the followings methods to validate the postback.

Method	Required	Parameters	Returns	Description
validate	Yes!		True/False	Checks whether the data is valid ICEPAY data. This method is required since it also sets the postback object.
doIPCheck	No	Boolean		Use IP range check in the validate method
canUpdateStatus	No	Icepay_StatusCode (String)	True/False	Compares the Icepay_StatusCode parameter against the postback statuscode. The statuscode param should be the current Icepay_StatusCode of the order. Validates whether the new (postback) statuscode can override the current statuscode, thus adding an extra layer of security. For example, an OK (paid) status can never change to an ERR (error) status.

11.2 Setters

Method	Required	Parameters
setMerchantID	Yes	MerchantID (int)
setSecretCode	Yes	SecretCode (String)
addToWhitelist	No	IP(s) (String)

11.3 Getters

Method	Returns	Datatype	Description
getStatus	Icepay_StatusCode status	String	Retrieve the statuscode of the transaction
getOrderID	The Order ID	String	Retrieve the ICEPAY Order ID previously set with the setOrderID method
getPostback	Read “The postback object” for more info	Object	Retrieve the postback object previouslt set with the validate method.
getTransactionString	"Paymentmethod: x | OrderID: x | Status: x | StatusCode: x | PaymentID: x | TransactionID: x | Amount: x"	String	Retrieve a human readable string with transaction IDs. For example to place within the order history comments.

11.4 Example

```
$logger = Icepay_Api_Logger::getInstance();
$logger->enableLogging(true)
      ->setLoggingLevel(Icepay_Api_Logger::LEVEL_ERRORS_AND_TRANSACTION)
      ->logToFile()
      ->logToScreen();

$icepay = new Icepay_Postback();
$icepay->setMerchantID(10000)
      ->setSecretCode("mySecretCode");

try {
   if($icepay->validate()){
      /* Actions based on statuscode */
                  switch ($icepay->getStatus()){
                              case Icepay_StatusCode::OPEN:
                                          break;
                              case Icepay_StatusCode::SUCCESS:
                                          break;
                              case Icepay_StatusCode::ERROR:
                                          break;
                              case Icepay_StatusCode::REFUND:
                                          break;
                              case Icepay_StatusCode::CHARGEBACK:
                                          break;
                  }
   } else die ("Unable to validate postback data");

} catch (Exception $e){
   echo($e->getMessage());
}
```

11.5 The postback object

Example within the postback script:

```
$icepay = new Icepay_Postback();
$icepay->setMerchantID(10000)
       ->setSecretCode("mySecretCode ");

if(!$icepay->validate()) die (“Unable to validate ICEPAY postback data”);

$postbackObj = $icepay->getPostback();
```

Example within the postback script with custom whitelist:

```
$icepay = new Icepay_Postback();
$icepay->setMerchantID(10000)
       ->setSecretCode("mySecretCode ")
       ->doIPCheck(true)
       ->addToWhitelist(‘1.1.1.0-1.1.1.2’) // IP Range
       ->addToWhitelist(‘1.1.1.0’) // Single IP
       ->addToWhitelist(‘1.1.1.0’,’1.1.1.8-1.1.1.9’) // Both single IP and IP Range

if(!$icepay->validate()) die (“Unable to validate ICEPAY postback data”);

$postbackObj = $icepay->getPostback();
```

Object	Description
$postbackObj->status	Contains the status of the payment. You can expect one of the following string values: OK: Payment completed OPEN: Payment not yet completed. Expect another postback in the near future. ERR: Payment was cancelled, failed or expired. REFUND: Payment was refunded by the merchant. CBACK: Payment was charged back by the end-user.
$postbackObj->statusCode	A description of the status.
$postbackObj->merchant	This is the merchant ID that is part of your API Key.
$postbackObj->orderID	This is the order ID generated by ICEPAY or set by the method setOrderID.
$postbackObj->paymentID	This is the ICEPAY transaction ID. Please mention this value when you ever contact ICEPAY support regarding a payment.
$postbackObj->reference	This is the reference of the payment that you may have set using the setReference method.
$postbackObj->transactionID	This is the transaction ID generated by the issuer.
$postbackObj->consumerName	The name of the end-user. This information is only available for iDEAL payments and wire transfers.
$postbackObj->consumerAccountNumber	The account number of the end-user. Only the last four digits is available. This information is only available for wire transfers.
$postbackObj->consumerAddress	Address of the end-user.
$postbackObj->consumerHouseNumber	Address' house number of the end-user.
$postbackObj->consumerCity	City in which the end-user resides.
$postbackObj->consumerCountry	Country in which the end-user resides.
$postbackObj->consumerEmail	E-mail of the end-user.
$postbackObj->consumerPhoneNumber	Phone number of the end-user.
$postbackObj->consumerIPAddress	IP address of the end-user.
$postbackObj->amount	Amount that is paid by the end-user.
$postbackObj->currency	ISO 4217 Currency Code
$postbackObj->duration	The call duration in seconds. This information is only available for phone payments.
$postbackObj->paymentMethod	Specifies which payment method was used for this payment, e.g. CREDITCARD, IDEAL, etc.

12 Class: Icepay_Api_Webservice – In general

The PHP API allows webservice integration as well. The SOAP PHP library is required in order to take advantages of these classes, this library should be present in most modern hosting providers.

Webservices offer a few similar features to the regular API, such as:

- Starting a payment and redirecting the end-user to the payment screen. (The webservices return more useful data in the payment request)

The Icepay_Result and Icepay_Postback classes are still required for integration. However, several new functionalities are available through use of the webservices, such as:

- Retrieve all the payment method data currently active for your merchant.
- Retrieve information of a transaction
- Use the phone services to create your own phone payments interface
- Use the reporting services to receive information related to transactions
- Request a refund

All Webservice related classes require you to include icepay_api_webservice.php and start an instance of the Icepay_Api_Webservice class.

Using the class:

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance();
```

These are the services available:

Method	Description
Icepay_Api_Webservice::getInstance()->paymentMethodService()	Retrieve payment methods
Icepay_Api_Webservice::getInstance()->paymentService()	Start a payment or get details of an existing payment
Icepay_Api_Webservice::getInstance()->refundService()	Request a refund
Icepay_Api_Webservice::getInstance()->reportingService()	Offers statistics. A pincode is required.

These are the service helpers available:

Method	Description
Icepay_Api_Webservice::getInstance()->filtering()	Filter retrieved payment methods
Icepay_Api_Webservice::getInstance()->singleMethod()	Get details of a single payment method from the retrieved payment method data.

13 Webservices: Retrieve the payment methods

This service is used to retrieve all the payment method data for a specific merchant. The data that is possible to retrieve is far more than the data stored in the Basic payment method classes. The webservice data comes directly out of our database, therefore it will always be up to date and mirroring the merchant configuration.

Since a large data object is retrieved, it should be cached and/or stored in either a file or database instead of being retrieved continuously. Payment method data should only be retrieved again once new payment methods and/or issuers have been added by the merchant or ICEPAY.

Start the Webservice instance:

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance()->paymentmethodService();
```

13.1 Setters

Method	Required	Parameters
setMerchantID	Yes	MerchantID (uint)
setSecretCode	Yes	SecretCode (String)

13.2 Methods

Method	Parameters	Returns	Description
retrieveAllPaymentmethods			Method to start the retrieval through the webservices
asObject		Object	Return the returned data in full as an Object
asArray		Array	Returns the data as a cleaned, usable array.
exportAsString		String	Returns the array as a string for easy storage
saveToFile	Filename(string), Directory (string)		Saves the getWebserviceDataAsString data to a file.

14 Webservices: Use the retrieved payment method to get specific data

This is a helper class intended to work in combination with the paymentmethodsService.

Start the Webservice instance:

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance()->singleMethod();
```

14.1 Load the payment method data

Payment method data previously saved by the paymentmethodsService can easily be loaded by one of the following methods:

Method	Parameters	Description
loadFromFile	Filename(string), Directory (string)	Load the payment method data saved by paymentmethodsService()->saveToFile()
loadFromArray	Array	Load the payment method data saved in the same formatting as retrieved in paymentmethodsService() ->retrieveAllPaymentmethods() ->asArray()
importFromString	String	Load the payment method data saved in the same formatting as retrieved in paymentmethodsService() ->retrieveAllPaymentmethods() ->getWebserviceDataAsString()

14.2 Select a payment method

In order to get specific information on a payment method, one must first be selected out of the saved data.

Method	Parameters	Description
selectPaymentMethodByCode	Payment method code (String)	Use this method to select a payment method when the issuer is not yet known. Example: “CREDITCARD”
selectIssuerByKeyword	Issuer keyword (String)	Use this method when the issuer is known. Example: “VISA”
selectCountry	ISO 3166-1-alpha-2 Country Code (String)	Selecting the country is required to get specific data since this differs per country. Example: “NL”

14.3 Get payment method information

After the selection is completed, it is possible to retrieve information through the following methods:

Method	Returns	Description
getCountries	Array	Get a list of supported countries
getCurrencies	Array	Get a list of supported currencies
getIssuerData	Array	Complete issuer data array
getIssuers	Array	Gets a list of issuers of the payment method
getMaximumAmount	Integer	Gets the maximum amount
getMinimumAmount	Integer	Gets the minimum amount
getPaymentmethodData	Array	Complete payment method data array

14.4 Example

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance()->singleMethod();

$all_payment_methods = $service->readFromFile("data",realpath("../wsdata"));

$paymentmethod = $service->selectPaymentMethodByCode(‘CREDITCARD’)->selectCountry(‘NL’);

var_dump($paymentmethod->getIssuers()); //Display all the Creditcard issuers available
```

15 Webservices: Use the retrieved payment method data and filter it

This is a helper class intended to work in combination with the paymentmethodsService. It is useful to present the end-user with optional payment methods filtered by several parameters.

Start the Webservice instance:

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance()->filtering ();
```

15.1 Load the payment method data

Payment method data previously saved by the paymentmethodsService can easily be loaded by one of the following methods:

Method	Parameters	Description
readFromFile	Filename(string), Directory (string)	Load the payment method data saved by paymentmethodsService()->saveToFile()
fromArray	Array	Load the payment method data saved in the same formatting as retrieved in paymentmethodsService() ->retrieveAllPaymentmethods() ->asArray()
fromString	String	Load the payment method data saved in the same formatting as retrieved in paymentmethodsService() ->retrieveAllPaymentmethods() ->getWebserviceDataAsString()

15.2 Filter the payment method data

Filter the loaded data

Method	Parameters	Description
filterByAmount	Integer	Filters out all payment methods that don't support this amount
filterByCountry	ISO 3166-1-alpha-2 Country Code (String)	Filters out all payment methods that don't support this country
filterByCurrency	ISO 4217 Currency Code (String)	Filters out all payment methods that don't support this currency.

15.3 Get the filtered data

Several options are available to retrieve the filtered data:

Method	Returns	Description
getPaymentMethods	Array	Returns the unfiltered payment method array
getFilteredPaymentMethods	Array	Returns the filtered payment method array
exportAsString	String	Export the filtered payment method array as string.

16 Webservices: The Payment Service

Start the Webservice instance:

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance()->paymentService ();
```

16.1 Class setters

Use the following methods to set data. These are chainable.

Method	Required	Parameters	Default value	Description
setMerchantID	Yes	MerchantID (int)		
setSecretCode	Yes	SecretCode (String)		
setSuccessURL	No	URL (String)		The URL used to redirect the customer to when the transaction is completed
setErrorURL	No	URL (String)		The URL used to redirect the customer to when there’s an error during the transaction

16.2 Start a checkout

An Icepay_PaymentObject must be created and passed to the checkOut method. More information about configuring the payment object can be found in section 5.

Use the checkout method to start a transaction.

Method / Service	Parameters	Returns	Description
checkOut	Icepay_PaymentObject	Icepay_TransactionObject	Start a transaction and returns the transaction data.

Returned transaction object:

Method	Description
getPaymentID	The ICEPAY transaction ID
getProviderTransactionID	The ID at the payment method provider
getTestMode	Is testmode transaction Y/N
getTimestamp	The ICEPAY timestamp
getEndUserIP	The customer/end-user IP address
getPaymentScreenURL	URL to the payment screen. Use this URL to forward the customer.

Sample:

```
$paymentObj = new Icepay_PaymentObject();
$paymentObj
          ->setPaymentMethod(“IDEAL”)
          ->setIssuer(“ING”)
          ->setAmount(1000)
          ->setCountry("NL")
          ->setLanguage("NL")
          ->setCurrency("EUR")
          ->setOrderID(101);

$service = Icepay_Api_Webservice::getInstance()->paymentService();
$service->setMerchantID(MERCHANTID)
       ->setSecretCode(SECRETCODE);

$transactionObj = $service->checkOut($paymentObj);
echo($transactionObj->getPaymentScreenURL());
```

.

16.3 Start a recurring payment

As of API 2.3.0 we offer recurring payments to be made through the API.

Instead of using the regular checkout, customers have to make a payment using the vaultCheckout method. This method will store the payment information in a secure place after a successful payment is done with the consumer id as a reference point.

Calling the vaultCheckout is basically the same as the regular checkout:

```
$paymentObj = new Icepay_PaymentObject();
$paymentObj
        ->setPaymentMethod('CREDITCARD')
        ->setIssuer('VISA')
        ->setAmount(1000)
        ->setCountry('NL')
        ->setLanguage('NL')
        ->setCurrency('EUR')
        ->setOrderID(icetest01);

$service = Icepay_Api_Webservice::getInstance()->paymentService();
$service->setMerchantID(MERCHANTID)
      ->setSecretCode(SECRETCODE);

$transactionObj = $service->vaultCheckout($paymentObj, '123456');
echo($transactionObj->getPaymentScreenURL());
```

.

The checkout method is replaced with vaultCheckout and the second argument you pass along is the consumer id.

Once the consumer id is known with ICEPAY, the merchant can start a payment on their own using the autoCheckout method.

```
$paymentObj = new Icepay_PaymentObject();
$paymentObj
        ->setPaymentMethod('CREDITCARD')
        ->setIssuer('CCAUTOCHECKOUT')
        ->setAmount(1000)
        ->setCountry('NL')
        ->setLanguage('NL')
        ->setCurrency('EUR')
        ->setOrderID(icetest01);

$service = Icepay_Api_Webservice::getInstance()->paymentService();
$service->setMerchantID(MERCHANTID)
      ->setSecretCode(SECRETCODE);

$transactionObj = $service->autoCheckout($paymentObj, '123456');
echo($transactionObj->getPaymentScreenURL());
```

Only two payment methods currently allow recurring payment methods; Creditcard and Direct Debit. Instead of using the normal issuers the following **must** be set in order for autoCheckout to work:

Method	Issuer
CREDITCARD	CCAUTOCHECKOUT
DDEBIT	IDEALINCASSO

Note: To test autoCheckout, using 123456 as consumer id will always set the success parameter to true in the result object.

Note: In order to achieve automatic billing, you still need to call our webservice each month with the autoCheckout function. You could do this by using a cron job for example.

16.4 Start an extended checkout

It is also possible to start an extended checkout through the API. An extended checkout includes more details about the order such as customer and product information. Some payment methods require an extended checkout in order to work.

Payment methods that currently require extended checkout:

- Afterpay

Note: All payment methods support the extended checkout method, but only extended checkout is advised if you are going to use Afterpay.

The difference between the extended checkout and the normal checkout is the Icepay_Order object. The Icepay_Order object includes extra information about the order and is required for the extended checkout method. Keep in mind that you must create the Icepay_Order object before calling the extendedCheckout method.

The Icepay_Order object is a singleton, and returns itself, so method chaining is possible.

Important: The total amount (All products + shipping + discounts) must equal the total amount in the PaymentObj. You can set the shipping costs and discounts using the proper methods as shown below in the sample.

Full Sample:

```
Icepay_Order::getInstance()
    ->setConsumer(Icepay_Order_Consumer::create()
        ->setConsumerID('1')
        ->setEmail('support@icepay.com')
        ->setPhone('0611223344')
    )
    ->setShippingAddress(Icepay_Order_Address::create()
        ->setInitials('John')
        ->setPrefix('')
        ->setLastName('Doe')
        ->setStreet('Zandstraat')
        ->setHouseNumber('10')
        ->setHouseNumberAddition('')
        ->setZipCode('2500AA')
        ->setCity('Amsterdam')
        ->setCountry('NL))
    ->setBillingAddress(Icepay_Order_Address::create()
        ->setInitials('John')
        ->setPrefix('')
        ->setLastName('Doe')
        ->setStreet('Zandstraat')
        ->setHouseNumber('10')
        ->setHouseNumberAddition('')
        ->setZipCode('2500AA')
```

```
            ->setCity('Amsterdam')
            ->setCountry('NL))
      ->addProduct(Icepay_Order_Product::create()
            ->setProductID('1')
            ->setProductName('iPhone')
            ->setDescription('Test Description')
            ->setQuantity('2')
            ->setUnitPrice('200')
            ->setVATCategory(Icepay_Order_VAT::getCategoryForPercentage(21))
      )
      ->setShippingCosts('1000')
      ->setOrderDiscountAmount('200')
);

$paymentObj = new Icepay_PaymentObject();
$paymentObj->setAmount(1200)
      ->setCountry("NL")
      ->setLanguage("NL")
      ->setIssuer('ACCEPTGIRO')
      ->setPaymentMethod('AFTERPAY')
      ->setReference("My Sample Website")
      ->setDescription("My Sample Payment")
      ->setCurrency("EUR")
      ->setOrderID(1000);

$webservice = Icepay_Api_Webservice::getInstance()->paymentService();
$webservice->setMerchantID(MERCHANTID)
         ->setSecretCode(SECRETCODE);

  $transactionObj = $webservice->extendedCheckout($paymentObj);

  echo($transactionObj->getPaymentScreenURL());
```

16.4.1 Icepay_Order

This singleton class contains all extra information about an order used for the extended checkout method in the webservices.

Method	Required	Param	Default value	Description
setConsumer	Yes	Icepay_Order_Consumer object		Set consumer id, email and phonenumber
setShippingAddress	Yes	Icepay_Order_Shipping object		Set shipping address
setBillingAddress	Yes	Icepay_Order_Shipping object		Set billing address
addProduct	Yes	Icepay_Order_Product object		Add products
setOrderDiscountAmount	No	Amount (String) in cents		Add a total order discount to your order, must be a numeric amount.
setShippingCosts	No	Amount (int) VATCategory name	 -1 Shipping Costs	

Note; Standard shipping costs are without VAT, to set the VAT Category you can also use the Icepay_Order_VAT class that determines the category based on the percentage taxes.

16.4.2 Icepay_Order_Consumer

The Icepay_Order_consumer object contains information about the consumer.

Method	Required	Parameters	Default value	Description
setConsumerID	Yes	Consumer ID (String)		Most of the time this is your webshop’s customer id
setEmail	Yes	Email (String)		Must be a valid email
setPhone	Yes	URL (String)		Must be a valid phonenumber

Sample: Setting the Consumer Object for Icepay_Order

```
Icepay_Order::getInstance()
    ->setConsumer(Icepay_Order_Consumer::create()
        ->setConsumerID('1')
        ->setEmail('test@test.com')
        ->setPhone('0612345678')
    )
```

16.4.3 Icepay_Order_Address

The address object contains all information about the customers address.

Method	Required	Parameters	Default value	Description
setInitials	Yes	Initials (String)		
setPrefix	No	Prefix (String)		
setLastname	Yes	Lastname (String)		
setStreet	Yes	Street (String)		
setHouseNumber	Yes	Housenumber (String)		If the housenumber is 11A, the housenumber would be 11
setHouseNumberAddtion	No	Housenumber addition (String)		If the housenumber is 11A, the housenumberaddtion would be A
setZipCode	Yes	Zipcode (String)		
setCity	Yes	City (String)		
setCountry	Yes	Country (String)		

Sample: Setting the Consumer Object for Icepay_Order

```
Icepay_Order::getInstance()
    ->setShippingAddress(
        Icepay_Order_Address::create()
            ->setInitials('John')
            ->setPrefix('')
            ->setLastName('Doe')
            ->setStreet('Zandstraat')
            ->setHouseNumber('10')
            ->setHouseNumberAddition('')
            ->setZipCode('2500AA')
            ->setCity('Amsterdam')
            ->setCountry('NL')
       )
```

To set the billing address, simply use setBillingAddress instead of setShippingAddress.

Note: If you want to use Afterpay, the billing country and shipping country must be the same.

16.4.4 Icepay_Order_Product

The product object contains all information about the customers address

Method	Required	Parameters	Default value	Description
setProductID	Yes	Product ID (String)		Most of the time this is your webshop's customer id
setProductName	Yes	Name (String)		Must be a valid email
setDescription	Yes	Description (String)		Must be a valid phonenumber
setQuanity	No	Quanity (String)	1	
setUnitPrice	Yes	Unitprice (String)		
setVATCategory	No	Category (String)	Standard	- zero - reduced-low - reduced-medium - standard - exempt

Sample: Setting the Product Object(s) for Icepay_Order

```
Icepay_Order::getInstance()
    ->addProduct(Icepay_Order_Product::create()
            ->setProductID('1')
            ->setProductName('Test Name')
            ->setDescription('Test Description')
            ->setQuantity('1')
            ->setUnitPrice('200')
            ->setVATCategory('standard'))
```

You can add as many products as you want, just remember that the total amount for the products must match the total amount of the Icepay Payment Object.

Note: Shipping costs must be added separate as a product.

16.4.5 Icepay_Order_VAT

The Icepay_Order_VAT object is used to determine what VAT category a product belongs to.

The default VAT categories in the extended checkout are:

- Zero
- Exempt
- Reduced-low
- Reduced-medium
- Standard

Method	Required	Param	Default value	Description
setCategories()	No	Tax ranges Array		
getCategories	No			
getCategoryForPercentage	No	Tax percentage (string)		

By default we have set the following categories for you.

```
$ranges = array(
            ‘exempt’ => -1,
            'zero' => 0,
            'reduced-low' => array('1', '5'),
            'reduced-medium' => array('6', '12'),
            'standard' => array('13', '100')
        );
```

If you want to change the tax ranges, use the above array setup to pass as argument for the setCategories() function.

Using the getCategoryForPercentage function:

```
Icepay_Order_VAT::getCategoryForPercentage(21)
```

The above example will return ‘standard’.

16.5 Phone payments

Several services are available to seamlessly integrate phone payments.

Method / Service	Parameters	Returns	Description
getPremiumRateNumbers		Array	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 enduser without having to start a new transaction.
phoneCheckOut	Icepay_PaymentObject	Array	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 array, which contains extra members such as the phone number etc., making seamless integration possible.
phoneDirectCheckout	Icepay_PaymentObject	Array	The phoneDirectCheckout web method allows you to initialize a new payment in the ICEPAY system with paymentmethod Phone with Pincode
validatePhoneCode	PaymentID (int), SMS code (int)	Boolean	The validatePhoneCode web method verifies the code that the end-user must provide in order to start a phone payment.

16.6 SMS payments

Several services are available to seamlessly integrate SMS text payments.

Method / Service	Parameters	Returns	Description
getPremiumRateNumbers		Array	Start a transaction and returns the transaction data.
smsCheckout	Icepay_PaymentObject	Array	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
validateSmsCode	PaymentID (int), SMS code (int)	Array	The validateSmsCode web method validates the code that the end-user must provide.

16.7 Retrieve payment data

Retrieve the transaction data of a specific payment.

Method / Service	Parameters	Returns	Description
getPayment	PaymentID (int)	Array	Retrieve payment real-time payment data

Returned transaction array:

Array key
MerchantID
Checksum
Timestamp
PaymentID
Amount
Currency
Description
Duration
ConsumerName
ConsumerAccountNumber
ConsumerAddress
ConsumerHouseNumber
ConsumerCity
ConsumerCountry
ConsumerEmail
ConsumerPhoneNumber
ConsumerIPAddress
Issuer
OrderID
OrderTime
PaymentMethod
PaymentTime
Reference
Status
StatusCode

For detailed information on the returned data, please read the webservices documentation.

17 Webservices: The Refund Service

Start the Webservice instance:

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance()->refundService ();
```

17.1 Class setters

Use the following methods to set data. These are chainable.

Method	Required	Parameters	Default value	Description
setMerchantID	Yes	MerchantID (int)		
setSecretCode	Yes	SecretCode (String)		

17.2 Services

The refund services

Method / Services	Parameters	Returns	Description
getPaymentRefunds	Payment ID (int)	Array	Returns the refund data of a payment.
cancelRefund	Refund ID (int), Payment ID (int)		Cancel a refund.
requestRefund	Payment ID (int), RefundAmount (int), refundCurrency (String)		Request a refund for a payment.

Returned getPaymentRefunds Array

Array key
RefundID
Status
RefundAmount
DateCreated

For detailed information on the returned data, please read the webservices documentation.

18 Webservices: Reporting Service

The reporting webservice offers various statistics. While developing, it is important to know a session is required at ICEPAY and another one cannot be created unless killed or expired. The expiry on a reporting session is 1 hour.

When creating a session an ICEPAY session ID is given. Our recommendation is to store this in your application while the session is active.

Start the Webservice instance:

```
require_once('icepay_api_webservice.php');
$service = Icepay_Api_Webservice::getInstance()->reportingService ();
```

18.1 Class setters

Use the following methods to set data. These are chainable.

Method	Required	Parameters	Description
setMerchantID	Yes	MerchantID (int)	
setSecretCode	Yes	SecretCode (String)	
setPinCode	Yes	PinCode (int)	Reporting requires a pincode.
setUserName	Yes		
setUserAgent	Yes		

18.2 Session methods

A session is required during the usage of the reporting service. To make things easier we give the option to store the session in a cookie or the PHP session. The initSession method will take care of loading the session data.

Method	Required	Parameters	Description
useCookie	No	Boolean	Store the Session ID in a cookie.
usePHPSession	No	Boolean	Use the PHP Session functionality.
initSession	Yes		This will start the session at ICEPAY and handles loading/saving the cookie and/or PHP Session. Returns a Boolean.
setSessionName	No	Session Name (String)	Change the name of the cookie/session.

Or Get/Set the session ID using the following methods

Method	Parameters	Returns	Description
getSessionID		String	Returns the ICEPAY session ID.
setSessionID	ICEPAY Session ID (String)		Set the session ID for usage by the services.

18.3 Services

Use the following services, a session ID must be set.

Method / Service	Parameters	Returns	Description
getMerchants		Array	Returns the merchants
monthlyTurnoverTotals	Month (int), Year (int), Currency (String)	Array	
searchPayments	searchOptions (Array)	Array	The searchoptions are listed in the next table.
killSession			Destroys the ICEPAY session and also the stored cookie and phpsessions if used

18.4 Search Options

Searching for payments requires an array with search options.

Example usage:

```
$reporting->searchPayments(
            array(
            "MerchantID" => “10000”,
            “OrderID” => “17”
            “Page” => 1
            ));
```

These are the search options, each of these is completely optional.

Array key	Array key
MerchantID	Amount
PaymentID	PaymentMethod
OrderID	ConsumerAccountNumber
Reference	ConsumerName
Description	ConsumerAddress
Status	ConsumerHouseNumber
OrderTime1	ConsumerPostCode
OrderTime2	ConsumerCity
PaymentTime1	ConsumerCountry
PaymentTime2	ConsumerEmail
CountryCode	ConsumerPhoneNumber
CurrencyCode	ConsumerIPAddress
Page	

For detailed information on the returned data, please read the webservices documentation.

18.5 Monthly Turnover totals

Retrieve the monthly turnover totals will result in an array with days and totals.

Returned array:

Array key
Day
Month
Year
TransactionsCount
Turnover
Duration

For detailed information on the returned data, please read the webservices documentation.