iPayment Gateway API (IPG API)

Accepting e-commerce payments for merchants

Version 3.1

Table of Contents

Version control

N	Author	Description	Date posted
1	Yavor Petrov	Version 1.0 (obsolete)	12.2009
2	Yavor Petrov	Version 2.0 (obsolete)	05.2010
3	Yavor Petrov	Version 3.0 first full	11.05.2012
4	Yavor Petrov	Shopping card format changed	15.05.2012
5	Milena Dyankova	Added "E-mail" field	22.05.2012
6	Milena Dyankova	Added "MIDName" and "OrderLink" fields. Added Appendix I.	23.05.2012
7	Milena Dyankova	Version 3.1 "RequestDateTime" and "RequestDateSTAN" parameters are removed from Method standard properties and are added to IPG Methods IPGPurchaseNotify / IPGPurchaseOK.	05.07.2012
8	Milena Dyankova	Added signature example	15.10.2012

Introduction

This document describes the interface for e-commerce payments via payment gateway. The Merchant should integrate the iPayment Gateway API (IPG API) at the site accepting card payments. IPG API will gain access to the entry point of iPayment Gateway (IPG) managed by Intercard Finance AD (iCARD). IPG will handle and guide the cardholder during the payment process, will check the card sensitive data and will process a payment transaction through card schemes (VISA, MasterCard, JCB).

IPG API will provide:

- Secured page and Secured communication channel with the Merchant
- Storing of merchant private data (shopping cart, amount, payment methods, transaction details etc.)
- Financial transactions to VISA, MasterCard, JCB – transparent for the Merchant
- Operations for the front-end: Purchase transaction
- Operations for the back-end: Refund, Reversal, Get Transaction Status
- 3D processing

Out of scope for this document:

- Merchant statements and payouts
- Merchant back-end (iMerchant)

The purpose of this document is to specify the IPG API Interface and demonstrate how it is used in the most common way.

All techniques used within the interface are standard throughout the industry and should be very easy to implement on any platform.

Continue on next page

Security and availability

Connection between Merchant and iCARD is handled through internet using HTTPS protocol (SSL over HTTP). Requests and responses are digitally signed both. iCARD host is located at tier IV datacenter in Luxembourg. Public address for IPG is BGP enabled and available through all first level internet providers.

Exchange folder for partners (if needed) is located at a SFTP server which enables encrypted file sharing between parties. The partner receives the account and password for the SFTP directory via fax, email or SMS.

iCARD supplies an emergency support line via e-mail or phone which is 7x24 enabled and reaches certified engineers.

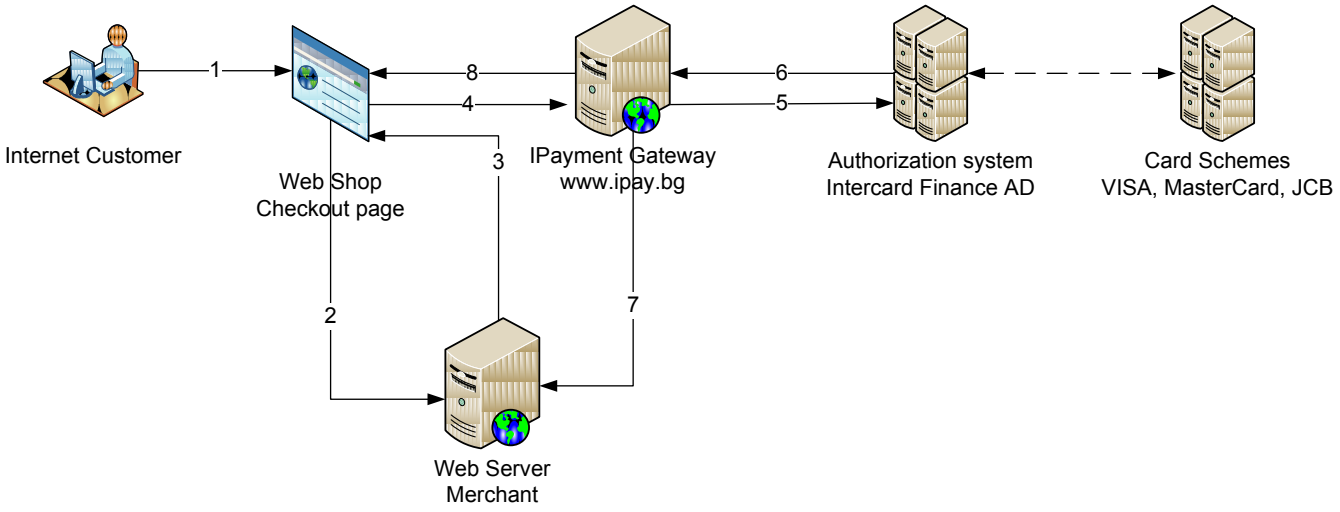
Test IPG API

A “by appointment” test service is available which allows the validation of the API calls. Testers should negotiate an exclusive access to the testing service and ensure monitoring by iCARD engineer.

Continue on next page

Accepting e-commerce payments with IPG interface

Overview



1. Internet customer at web shop checkout page
2. Payment initiated by customer.
3. Merchant web server initiates payment through IPG. Merchant web server should redirect the browser to IPG web address.
4. Customer web browser is redirected to IPG web page.
5. Customer is requested to input the card data and press PAY. IPG handles the 3D secure processing and financial transaction messaging.
6. IPG receives the details for the payment – successful or declined.
7. IPG passes the result to Merchant Web Server.
8. IPG redirects to Web Shop "checkout result" page.

HTTP POST

Data transfer between Merchant and IPG is made by HTTP POST. All the parameters for the requests are in the body in [parameter=value] form. Separator between tokens is [&]. The body is URL Encoded. Character encoding is UTF-8.

Example:

```
POST /somescript.php HTTP/1.1
Host: www.somesite.com
User-Agent: Mozilla/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

userid=joe&password=guessme&user_type=1
```

Data Type Formats

Data Type in document	Description	Example
int	integer	1
String	string	This is a string
Date	ISO 8601 date string YYYY-MM-DD	2012-03-31
DateTime	ISO 8601 datetime string YYYY-MM-DD HH:mm:SS	2012-03-31 23:59:59
A(n)	Alpha string. [n] characters required	Alpha string
AN(n)	Alphanumeric string. [n] characters required	Alphanumeric string
N(n)	Numeric string. [n] characters required. Number is left-padded with zeroes.	000123
double	Numeric string with decimal point. Only point is used (no commas or other characters for decimal point)	34.56
BASE64	Sting used to pass binary data. The binary data should be converted to base64 standard.	YW55IGNhcm5hbCBwbGVhc3VyZQ==
XML	Simple in place XML array.	<body> <param>1</param> <value>2</value> </body>

Signatures

In every message a signature is supplied. The signature is a signed HASH of all the values from properties sent in the request. All values must be URL encoded first.

For signing process, both iCARD and the Merchant generate public and private key pairs and exchange the public keys. Key pairs are generated using RSA algorithm. Every of the parties are using the private key to sign the message and the opposite side authenticate the sender with corresponding public key.

Signatures are calculated using the following mechanism. All data in POST request without the Signature property is used to calculate hash using SHA1 algorithm. Then SHA1 value is signed with RSA. The Signature property is concatenated at the end of the POST string. The opposite side should check the signature in the same way. Calculate SHA1 for the POST string excluding the Signature section then check with VerifySignature.

During the business lifecycle, there could be a need the keys to be changed, or more than one key to be used in communication. IPG supports unlimited number of exchanged keys, for iCARD and Merchant both. A key index is assigned to every key, starting from 1 to MAXINT. The key index of the key used to sign the request is supplied as a parameter in every transmission.

Example

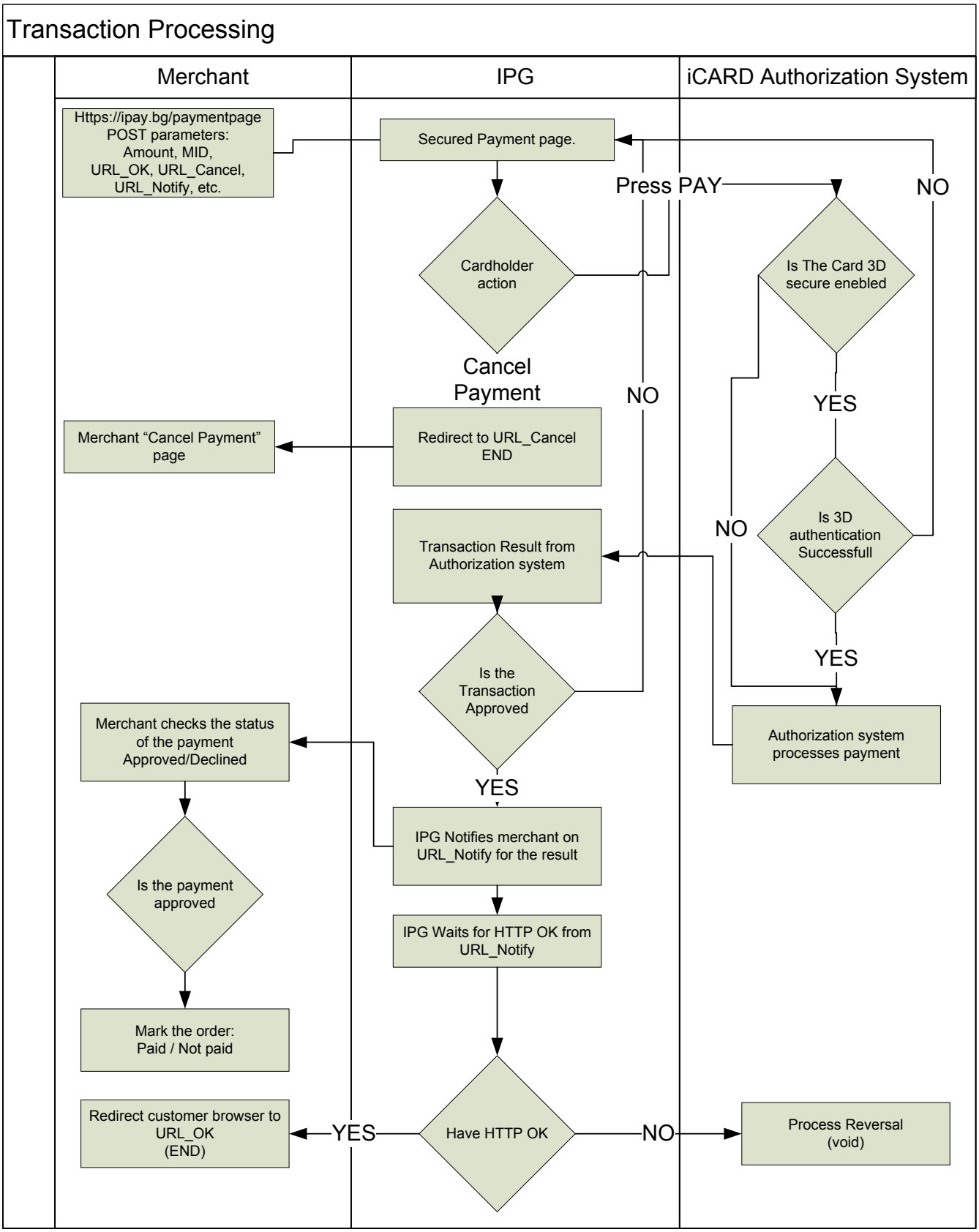
```
<?php
$postData = array('IPGmethod'=>'IPGPurchase', .....); #The $_POST array
$privKey = '-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC8oMfOTxHN0WRPmRojUVeaj992GpzcoKibAc2i6P2yM0mOeYho
TsdSpKzVYDRYrtGlRb8B2+4/R67nxM9O/Tn5YtGXkLEVXI4mWrGTTRZr8afF97zx
t1bThu/fMxpDfKGSQoq/d5sd7wHiu/vAGo4XSVwXk5iQeZy+rP9pCO7rQQIDAQAB
AoGASWTV2bRyXP8IZOBRh0RzLbSlYjLgrgflEssU1DqL2/aQvFsVdGCStdlVVoDk
XU1ITWJh+7szbHPb3lp5v2ZQU8lVacwBpLY7RHZ/BXiwmcg3iMwqEFCF2S+cPijA
EOXrvv0N7G8r1qYGfbEVs3mDtFaTCbJiAQFiUxfWGpmNK0ECQQDtgqf/azooeBWa
43UZnM+YIHVSkdQtsRVaw4gCv+RWZVonqjRg+zanqLzwSTcveRVSIZiu3CfG5/sk
co54Ki51AkEAy0/yzlEiFOJv0q8eANEB5fGj5LVAC+9LBzbsmkO054s9HdbuThQZ
YDGi2TI6YRx/l/uRMNYTSmKjYt79gWQIHQJAdVf3HndgrXve2L6GLVhPLE7lCB1q
YgS6kzRFr24VJyY965jo9f1HnH/+kQzrSfYdtY1JvSKiOGCGsRQ0FWRpvQJBAI4u
sxcmFjeUw68LWGgpwrIUcxGWz9uI1WeOOZkIkJL9BRjBHpbr53MmQ0Sxo7IWRAT9
oWQN0h/LK4gReifq1OECQGaKHRJ9JMLnIrF0a8wHKjM8PbEPu2oelfHK/HeHpJMK
xnecld/4RwEx7ytm2+UvaDo1cjYu0ig0D127pT07+yk=
-----END RSA PRIVATE KEY-----'; #This is an example of RSA private key

$concData = urlencode(stripslashes(implode('', $_POST))); # You need to concatenate all values from $postData and to URL-encode the result
$dataHash = sha1($concData); # Create sha1 hash of concatenated data
$privKey = openssl_get_privatekey($privKey);
openssl_sign($dataHash, $signature, $privKey); # Signed data in binary
$signature = base64_encode($signature); # Base64 encoding of the signature
$postData['Signature'] = $signature; # Now you need to add the signature to the post request
```

Signature verification example

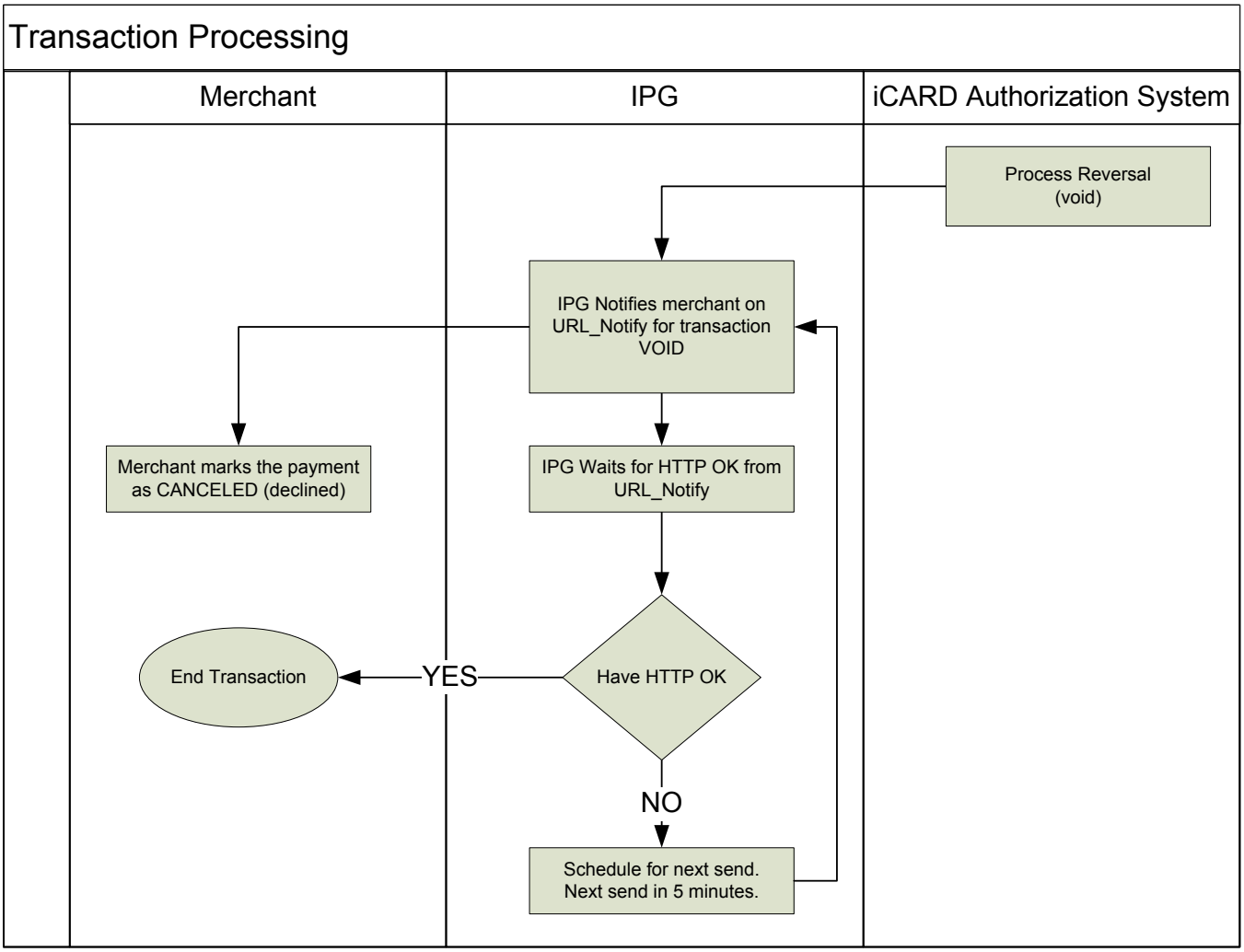
```
$data = $_POST;
$pubKey = '-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC4ur+fZBqNjnm1XJSJrzf8vyIv
xfXew44RKJv9kpPiSEtGaRiAmqZhMWsW/fD2Drnh1A6gCgfWIv/3Zgr18GZ/Heqm
h5n9HmQndHAB2nZnFLOioL9v6awAbqVeqYBMzp97UkruxXDtqejL7w8WkxearqpU
BBbcPHA2gMp0hRN/MwIDAQAB
-----END PUBLIC KEY-----';

$signedData = $data['Signature'];
unset($data['Signature']);
$concData = urlencode(stripslashes(implode('', $data)));

$pubKeyId = openssl_get_publickey($pubKey);
$signedData = base64_decode($signedData);
```

```
$res = openssl_verify(sha1($data), $signedData, $pubKey);
openssl_free_key($pubKeyId);

if($res==1){
        //success
}else{
        //not success
}
```

Understanding transmission mechanism



Continues from previous page



In every request there are several parameters that are always supplied. Bellow they are called 'standard properties'. Once defined bellow they won't be described in every single command listed bellow in the specification, they should be considered as existing to every command.

Method standard properties

Property	Typical value	Type	Description
IPGmethod	IPGPurchase	String	Name of the method requested for execution from IPG.
Signature	Byte[]	BASE64	Signed HASH for all properties in the command. Signature is **ALWAYS THE LAST PARAMETER IN THE POST,** as it is not used to calculate the hash**.**
KeyIndex	1	Int	Identifier of the private key used for signature (if more then 1)
IPGVersion	3.1	string	Version of protocol used for transition.
Language	EN	A(2)	ISO 2-character code for the desired language on the

			payment page. If IPG cannot fulfill the requested language, it will set the English language as defaults. Currently supporting EN, FR, DE, BG, ES, RO.
Originator	100	Int	Value that uniquely identifies the merchant company that has signed a contract with Intercard Finance AD.

Response standard properties

Upon HTTP request, the party should respond with HTTP OK. Every other response should be treated as communication error, call error, server error or system malfunctions.

In every HTTP response the party should include only the string OK. Every other content will be considered as an error status.

IPG methods (alphabetical order)

API function call	Description
MERCHANT TO IPG	
IPGPurchase	This is the standard method for checkout at web shop.
IPGPurchaseIPAY	Same as above but purchase is made using iPAY.eu account.
IPGReversal	This command cancels a previously executed payment (void). Usually for back-office.
IPGRefund	Credit to cardholder, e.g. return money. Usually for back-office.
IPGGetTxnStatus	Returns the status and the parameters of a previously executed payment. Usually for back-office.
IPG TO MERCHANT	
IPGPurchaseNotify	IPG will respond with this method on successful payment. The call will be made on previously supplied URL_Notify.
IPGPurchaseOK	IPG will redirect with this method on successful payment. The call will be made on previously supplied URL_OK.
IPGPurchaseCancel	IPG will redirect with this method when the customer chooses cancel payment. The call will be made on previously supplied URL_Cancel.
IPGPurchaseRollback	IPG will notify that a reversal is passed for previous successful authorization. The merchant should mark the order as not paid (in case, the merchant has received IPGPurchaseNotify method). This is used when IPG do not receive and HTTP OK from the merchant as a response for IPGPurchaseNotify method.

All commands described bellow do not include the standard properties discussed in the previous topic. However the standard properties are mandatory for all commands.

Purchase with payment card (API call: IPGPurchase)

Purpose

This method initiates the beginning of the payment process for a cardholder. The cardholder is placed on a page that requests entering payment card details.

IPG will check:

- Valid MID.
- Valid currency with regards to the MID.

Method properties

Property	Typical value	Type	Required	Description
MID	000000000000123	AN(15)	YES	Identifier of the virtual terminal used for the purchase.
MIDName	Merchant Web Shop	String	YES	User friendly name of the merchant web shop. The cardholder will see this name on some notices in the payment page.
Amount	23.45	Double	YES	The amount of the payment requested.
Currency	978	N(3)	YES	ISO numeric currency code. The currency for the payment must be equal to the currency of the MID.
CustomerIP	82.119.81.30	String	YES	Dotted-decimal string, that holds the customer IP address as reported at merchant web shop.
OrderID	201203319999999	String	YES	Placeholder for the merchant. Used to put some data that will help the merchant to recognize for which order is the payment. Up to 255 characters.
OrderLink	http://site.ext/	String	NO	The link of the page with the order from the merchant web shop.
BannerIndex	1	Int	YES	Index specified in IPG for every banner provided by the Merchant. The Merchant may choose to select a proper banner for every payment. The banner is displayed on the payment page.
URL_OK	http://site.ext/paymentOK	String	YES	The page where the cardholder should be redirected on successful payment.
URL_Cancel	http://site.ext/paymentNOK	String	YES	The page where the cardholder should be redirected when <Cancel> is pressed on the payment page.
URL_Notify	http://site.ext/paymentNotify	String	YES	Address supplied by the partner, where the IPGPurchaseNotify API call will send the parameters for the successful payment.
Note		String	NO	Text associated with the purchase.
CartItems	2	Int	YES	The number of rows (items) in the logical record Cart. If there will be some additional fees/taxes for the cardholder, they need to be added as new items.
Cart	Logical Holder	Logical Record	YES	Array provided by the Merchant. The array describes the content of the shopping cart. The content will be displayed on the IPG payment page.
Email	name@website.com	String	YES	This is the cardholder's email.

*URL_OK, URL_Cancel and URL_Notify could be the same. IPG will supply the proper method in the property "IPGmethod".
Properties MID in combination with OrderID gives a unique identifier for the request of a partner. IPG will reject duplicated transmission.

Cart Logical Record

Cart logical record consists of standard POST parameters with the form name=value. For every consequent item an index is added that shows the logical record number for the item (ex. Atricle_1). Indexes are from 1 to <CartItems>.

Property	Typical value	Type	Description
Article	HP ProBook 6360b sticker	String	Name of an article in the shopping cart.
Quantity	2	Int	How many pieces of an article.
Price	2.34	Double	Price of a single unit.
Amount	4.68	Double	Quantity*Price for the article.
Currency	978	N(3)	Should be the same currency as in the purchase amount.
Signature	Byte[]	BASE64	Signed HASH for all properties in the command. Signature is **ALWAYS THE LAST PARAMETER IN THE POST,** as it is not used to calculate the hash**.**

Example

New lines and tabulators are included for better reading and do not exist in the POST request.

```
MID=000000000000123&
MIDName=Example Web Shop Name&
Amount=23.45&
Currency=978&
CustomerIP=82.119.81.30&
OrderID=1854&
OrderLink=http://site.ext&
BannerIndex=1&
URL_OK= http://site.ext/paymentOK&
URL_Cancel=http://site.ext/paymentNOK&
URL_Notify=http://site.ext/paymentNotify&
Note=note&
CartItems=2&
Email=name@website.com&
Article_1=HP ProBook 6360b sticker&
Quantity_1=2&
Price_1=2.34&
Amount_1=4.68&
Currency_1=978&
Article_2=HP ProBook 6360b sticker&
Quantity_2=1&
Price_2=2.00&
Amount_2=2.00&
Currency_2=978
```

Successful payment notification (API call: IPGPurchaseNotify / IPGPurchaseOK)

Purpose

This method is used by IPG to notify the merchant for a successful payment and to pass all needed parameters for the payment on URL_Notify. After successful response for this method IPG will redirect the customer browser to URL_OK and will pass same parameters with IPGPurchaseOK method.

Method properties

Property	Typical value	Type	Description
MID	000000000000123	AN(15)	Echo from IPGPurchase.
Amount	23.45	Double	Echo from IPGPurchase.
Currency	978	N(3)	Echo from IPGPurchase.
CustomerIP	82.119.81.30	String	Echo from IPGPurchase.
OrderID	201203319999999	string	Echo from IPGPurchase.
Approval	123456	String	Approval code return by the issuer of the card. Used to identify the financial transaction within the card schemes and the issuer.
IPG_Trnref	12345678923	String	Used to uniquely identify a transaction in IPG. Used as a parameter for subsequent refund of reversal if needed.
RequestDateTime	2012-03-31 23:59:59	DateTime	Date/time of the request
RequestSTAN	123456	N(6)	Consequent number from 1 to 999999. Used for request unique match.
Signature	Byte[]	BASE64	Signed HASH for all properties in the command. Signature is **ALWAYS THE LAST PARAMETER IN THE POST**, as it is not used to calculate the hash.

Cancelation of payment notification (API call: IPGPurchaseCancel)

Purpose

This method is used by IPG to notify the merchant that the customer has canceled the payment. IPG will redirect with this method when the customer choose cancel payment. The call will be made on previously supplied URL_Cancel.

Method properties

Property	Typical value	Type	Description
MID	000000000000123	AN(15)	Echo from IPGPurchase.
Amount	23.45	Double	Echo from IPGPurchase.
Currency	978	N(3)	Echo from IPGPurchase.
CustomerIP	82.119.81.30	String	Echo from IPGPurchase.
OrderID	201203319999999	string	Echo from IPGPurchase.
Signature	Byte[]	BASE64	Signed HASH for all properties in the command. Signature is **ALWAYS THE LAST PARAMETER IN THE POST,** as it is not used to calculate the hash**.**

Rollback of previous notification (API call: IPGPurchaseRollback)

Purpose

This method is used by IPG to notify that a reversal is passed for previous successful authorization. The merchant should mark the order as not paid (in case, the merchant has received IPGPurchaseNotify method). This is used when IPG do not receive and HTTP OK from the merchant as a response for IPGPurchaseNotify method. The call will be posted to URL_Notify.

Method properties

Property	Typical value	Type	Description
MID	000000000000123	AN(15)	Echo from IPGPurchase.
Amount	23.45	Double	Echo from IPGPurchase.
Currency	978	N(3)	Echo from IPGPurchase.
CustomerIP	82.119.81.30	String	Echo from IPGPurchase.
OrderID	201203319999999	string	Echo from IPGPurchase.
Signature	Byte[]	BASE64	Signed HASH for all properties in the command. Signature is **ALWAYS THE LAST PARAMETER IN THE POST,** as it is not used to calculate the hash**.**

Get transaction status for previously executed payment (API call: IPGGetTxnStatus)

Purpose

This method is used by Merchant to get the current status of previously executed payment. The IPG API will return an xml with detailed information about a specific OrderID. This method is intended to be

utilized by the Merchant in his website back-end. The Merchant could decide whether or not to use this method.

Method properties

Property	Typical value	Type	Required	Description
OrderID	201203319999999	String	YES	Placeholder for the merchant. Used to put some data that will help the merchant to recognize for which order is the payment. Up to 255 characters.
MID	000000000000123	AN(15)	YES	Identifier of the virtual terminal used for the purchase.
OutputFormat	xml	String	NO	Output format of data. The property can be "xml" or "json". If it is not specified in the request, the default value is "xml".

Example of the xml

```
<?xml version="1.0"?>
<ipg_responce>
  <method>IPGGetTxnStatus</method>
  <order_id>XXXXXX</order_id>
  <log>
    <item>
      <time>18.07.2012 11:43:58</time>
      <action>Received new POST request</action>
      <result>
        <MID>XXXXXXXXXXXXXXX</MID>
        <MIDName>XXXXXXXXXXXXXXXXX</MIDName>
        <IPGmethod>IPGPurchase</IPGmethod>
        <Currency>978</Currency>
        <CustomerIP>193.48.246.14</CustomerIP>
        <OrderID>XXXXXX</OrderID>
        <BannerIndex>1</BannerIndex>
        <URL_OK>http://www.website.com/OK</URL_OK>
        <URL_Cancel>http://www.website.com/Cancel</URL_Cancel>
        <URL_Notify>http://www.website.com/Notify</URL_Notify>
        <Language>EN</Language>
        <IPGVersion>3.1</IPGVersion>
        <Originator>XXX</Originator>
        <KeyIndex>1</KeyIndex>
        <Email>customer@website.com</Email>
        <CartItems>2</CartItems>
        <Article_1>Article name</Article_1>
        <Quantity_1>1</Quantity_1>
        <Price_1>10.00</Price_1>
        <Amount_1>10.00</Amount_1>
        <Currency_1>978</Currency_1>
        <Article_2>Delivery</Article_2>
        <Quantity_2>1</Quantity_2>
        <Price_2>8.90</Price_2>
        <Amount_2>8.90</Amount_2>
```

```
        <Currency_2>978</Currency_2>
        <Amount>18.90</Amount>
<Signature>XwVlhn0yQsp9zLK5WANPSDQkvMWfbNHE8ZCjF5VGlkosTECitZc1/GedVNc+VpWmw5JRTcqv0
4orFv2YWSWdXkF8dmHo3sqJKYO0WJcahPpYk+xXf+F3J9U1tsUUkMGTLhXNCq+zKS/DDTlTVLY2gBiMSsoEn
T116OC5+nMl6vs=</Signature>
      </result>
    </item>
    <item>
      <time>18.07.2012 11:43:58</time>
      <action>Checking is purchase paid on Start page</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:58</time>
      <action>Checking version</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:58</time>
      <action>Checking method</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking signature</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Trying to get MID info </action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking if the MID is related with this CID</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking is amount >0</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking valid currency and if the post currency is the currency of
MID</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking is order_id unique</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Trying to get banner GUID for CID</action>
      <result>OK</result>
    </item>
```

```
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking is cart_items >0</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking the URLs (url_ok, url_cancel, url_notify)</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Checking cart</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Valid parameters</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:43:59</time>
      <action>Display page:</action>
      <result>Payment page</result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Client is redirected to register payment page (page after "Pay"
button)</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Checking if purchase is already paid (on register payment
page)</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Starting card details validation</action>
      <result></result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Checking for required fields</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Checking PAN</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Checking Expire date</action>
      <result>OK</result>
    </item>
    <item>
```

```
      <time>18.07.2012 11:45:04</time>
      <action>Checking CVC</action>
      <result>OK</result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Card successfully validated</action>
      <result></result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Checking Merchant for 3DS for card scheme</action>
      <result>V</result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Merchant is 3DS</action>
      <result></result>
    </item>
    <item>
      <time>18.07.2012 11:45:04</time>
      <action>Start checking card for 3DS.....</action>
      <result></result>
    </item>
    <item>
      <time>18.07.2012 11:45:06</time>
      <action>Card check for 3DS</action>
      <result>Y</result>
    </item>
    <item>
      <time>18.07.2012 11:45:06</time>
      <action>Redirecting to ACS</action>
      <result>
        <ACS_URL>https://ACS.website.com</ACS_URL>
        <TransactionAmount>1890</TransactionAmount>
        <TransactionDisplayAmount>18.90 EUR</TransactionDisplayAmount>
      </result>
    </item>
  </log>
  <status>0</status>
  <status_msg>Success</status_msg>
</ipg_responce>
```

Make a refund for previously executed payment (API call: IPGRefund)

Purpose

This method is used by Merchant to initiate a refund of previously executed payment. The IPG API will return an xml with the result. This method is intended to be utilized by the Merchant in his website back-end. The Merchant could decide whether or not to use this method.

Method properties

Property	Typical value	Type	Required	Description
MID	000000000000123	AN(15)	YES	Identifier of the virtual terminal used for the purchase.
IPG_Trnref	12345678923	String	YES	Used to uniquely identify a transaction in IPG. Used as a parameter for subsequent refund of reversal if needed.
Amount	23.45	Double	YES	The amount of the payment requested.
Currency	978	N(3)	YES	ISO numeric currency code. The currency for the payment must be equal to the currency of the MID.
OutputFormat	xml	String	NO	Output format of data. The property can be "xml" or "json". If it is not specified in the request, the default value is "xml".

Example of the xml

```
<ipg_responce>
        <method>IPGRefund</method>
        <trnref>12345667</trnref>
        <amount>0.1</amount>
        <currency>978</currency>
        <status>0</status>
        <status_msg>Success</status_msg>
</ipg_responce>
```

Make a reversal for previously executed payment (API call: IPGReversal)

Purpose

This method is used by Merchant to initiate a reversal of previously executed payment. The IPG API will return an xml with the result. This method is intended to be utilized by the Merchant in his website back-end. The Merchant could decide whether or not to use this method.

Method properties

Property	Typical value	Type	Required	Description
IPG_Trnref	12345678923	String	Yes	Used to uniquely identify a transaction in IPG. Used as a parameter for subsequent refund of reversal if needed.
OutputFormat	xml	String	No	Output format of data. The property can be "xml" or "json". If it is not specified in the request, the default value is "xml".

Example of the xml

```
<ipg_responce>
        <method>IPGReversal</method>
        <trnref>123456789</trnref>
        <status>0</status>
        <status_msg>Success</status_msg>
</ipg_responce>
```

Appendix I – Error messages

Error	Description
E_MISSING_REQ_PARAMS	Some of required fields from the POST request are missing.
E_SIGNATURE_FAILED	The signature is not valid.
E_IPAY_ERROR	Invalid or missing response from IPG servers. Please contact IPG engineers.
E_INVALID_MID	The MID is not valid.
E_INVALID_PARAMS	Some of required fields are not valid.

Note:
These error messages will be visible on the payment page only in test environment. In production environment the cardholder will see an error page with the following text:

You are not able to proceed with the payment process.
Some of required information is missing. Please try again.
<< Return to [MIDName]