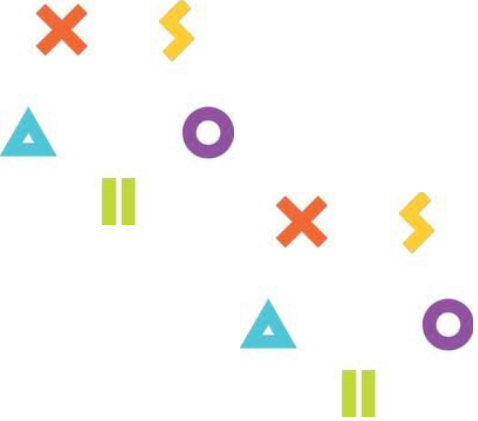

Virtual Currency

API Guide

Xsolla

March 2012

Revised: 3/28/2012

Copyright Notice

Trademarks

Xsolla

www.xsolla.com

Contents

API Characteristics

The Virtual Currency API enables developers to sell virtual currency without spending limits, at a specified currency exchange rate. The API can convert any payment amount into virtual currency, without change. The Advanced Virtual Currency API allows converting real currency into virtual currency according to any formula chosen by the developer.

The Virtual Currency API is an easy and accessible solution for those projects which have in-game virtual currency of a predetermined value. Users receive the preset amount of virtual currency when replenishing their accounts from within a game. Players can make payments from e-wallets, payment kiosks, online-banking, et cetera.

The Virtual Currency API offers the following benefits:

Entire payments. The API converts the whole amount of payment into virtual currency. Your game's and the payment system's minimum or maximum payment amount are the only limits.

Recurrent payments. Once players receive their Xsolla number, they can recharge the game account balance anywhere, anytime, without visiting Xsolla's website or filling out forms in the game. They need only click the Xsolla button on the payment system's page and enter a few details such as Xsolla number, email, and ID.

Co-branding with payment systems. The Virtual Currency API allows us to place our games catalogue on other payment systems' websites, typically in the "Where to spend" section. We can add your projects to the catalogue, enabling users to make payments directly at the payment system or on the Xsolla website.

Audience

This document is intended for developers who want to integrate the Xsolla Virtual Currency API into their websites, games, and applications. It assumes you are familiar with the basic concepts of APIs, HTTP request methods, and the REST style of software architecture.

Anatomy of Xsolla Transaction

The Xsolla system processes payments in 2 mandatory steps: user status check and transaction processing. Optionally, you may also cancel a transaction. You pass the request type in the command parameter as **check**, **pay** and **cancel**. During the user

status check, you must verify that your database contains a user with the specified ID. During the transaction processing step, you perform an internal check of the payment ID and amount increase the user's balance.

Your database must not contain two successfully processed payments with the same id. If the system repeats a request with an id already existing in your database, then your application must return the processing result of the previous request.

Requirements for Your Application

Your application:

Should accept HTTP or HTTPS requests from the following IP addresses: 94.103.26.178 and 94.103.26.181

Must handle parameters passed by the GET method in CP1251 XML encoding

Must create a response in XML format in CP1251 (if the response contains characters of national alphabets)

Must perform data exchange in the request-response mode. Response must take up max. 60 seconds. Otherwise, connection will be terminated by timeout

Request Format

The Xsolla system sends you requests in an HTTP or HTTPS URL, for example:

```
https://test.project.com?
project=133&command=pay&id=14332453&v1=BilMURka&v2=&v3=&sum=902.481&date=2012-03-
26+08%3A14%3A43&md5=92f93db6770adccf54d8063612858693
```

Each payment in the Xsolla system has a unique identifier which is passed to the developer via the **id** parameter. It is used to accomplish mutual settlement and matching.

In a payment request, the Xsolla system passes the payment date via the **date** parameter in the following format: YYYYMMDDHHMMSS. Date is used to accomplish mutual settlement and matching.

sum is a fractional number accurate to hundredths. “.” (point) is used as a separator. It stands for the amount of virtual currency purchased by the user.

Response Format

Return your response to system requests in XML format in CP1251 (if the response contains characters of national alphabets), for example, with the following structure:

```
<?xml version="1.0" encoding="windows-1251"?>
<response>
  <id>1234567</id>
  <id_shop>9876543</id_shop>
  <result>0</result>
  <comment>Success</comment>
</response>
```

Result Codes

Code	Description	Fatal?
0	Ok	No
1	Temporary error, retry request later	No
2	Invalid user ID	Yes
3	Invalid MD5 signature	Yes
4	Invalid request format (invalid amount, some parameters are missing)	Yes
5	Another error (to be described in *comment*)	Yes
7	Certain user's payment cannot be processed/denied due to technical reasons	Yes

User ID Existence Request

Check

Xsolla server sends the user ID existence request to your Payment Script URL specified in your project details in your Xsolla Merchant Account. Transaction details are sent by the GET method in CP1251 encoded XML and contain the parameters listed below.

Request Parameters

Field Name	Type	Description	Required?	Example
command	String	For the user ID existence check, the value must be **check**. Means that user ID existence is being checked	Yes	
v1	String	The user ID that is unique to your platform (for example, nick or email). Can contain letters, digits and special characters. Length limit is 255 characters.	Yes	demo
v2	String	Additional ID parameter, for example game server. Length limit is 200 characters.	No	0
v3	String	Additional ID parameter, for example game server. Length limit is 100 characters.	No	0
md5	String	The string for unsanctioned access prevention	Yes	bdfa807b47c58c43e3d6dcaaa3a1301d

Generating the md5 Signature

The signature is required for payment processing security. It is generated with md5 hash algorithm from the string that is derived by concatenating the parameters specified below.

```
md5(<command><v1><secret_key>)
```

secret_key is the secret word that you chose for your Merchant Account.

Example

The string for signature computing contains the following parameters:

```
command = check
v1 = demo
secret_key = password
```

The string before applying MD5 hash would be:

```
checkdemopassword
```

MD5 hash of this string would be:

```
bdfa807b47c58c43e3d6dcaaa3a1301d
```

As a response to this request, your payment application should generate one of the following responses in XML.

If user ID exists in developer's system, balance increase can be done:

```
<response>
  <result>0</result>
</response>
```

If user ID does not exist in developer's system, balance increase cannot be done:

```
<response>
  <result>7</result>
  <comment>Account is disabled or not present</comment>
</response>
```

Result Field Values

Result	Description
0	It is possible to increase balance of the user ID indicated in the request. After successful status check, the system will send balance increase request.
7	Transaction for this user ID cannot be processed.

Additional anti-fraud protection

You can pass additional parameters, which can provide more information about the user. These additional counter-measures will help us to trust the users and properly customize the filters. The XML response will be in the following format:

```
<response>
  <result>...</result>
  <comment>...</comment>
</response>
```

```
<specification>
   <s1>...</s1>
   <s2>...</s2>
   …
</specification>
```

All parameters should be passed in accordance with the protocol documentation. Sum is transferred in a variable with the type Float, separator is "." (0, 1, 2 characters after point). Date is transferred in a variable with the type String in the format YYYYMMDDHHMMSS. For example you can use s1 parameter for the authority (e.g. trust the user w/ authority 100+), s2 date of registaration, s3 amount of virtual currency spent for the whole time, s4 Date of birth, etc. Please ask your account or integration manager to set up these parameters and add the meaning of each one.

Sample Code

This sample code illustrates how to perform the user existence check. See Appendix A for sample code illustrating how to check the md5 signature and IP address.

```php
<?php

$v1 = $_GET['v1'];

// checking nickname
// check_nickname - your function which checks nickname existence in your database
$check = check_nickname($v1);

// if nickname exists
if ($check) {
   $code = '0';
   $comment = 'success';
} // if not
else {
   $code = '7';
   $comment = 'fail';
}

echo '<xml version="1.0" encoding="utf-8">
   <response>
         <code>' . $code . '</code>
         <comment>' . $comment . '</comment>
   </response>';

// check function
function check_nickname()
{
   // your code here

   return true;
}
```

Balance Increase Request

Pay

In each successful transactional event, Xsolla server will send the transaction details to developer's Payment Script URL. This is specified in your project details at Xsolla Merchant Account. Transaction details are sent by the GET method in cp1251 encoded XML and contain the parameters listed below.

Request Parameters

Field Name	Type	Description	Required?	Example
command	String	For the balance increase request, the value must be **pay**	Yes	
id	String	Order ID in Xsolla system	Yes	7555545
v1	String	The user ID that is unique to your platform. Length limit is 255 characters.	Yes	demo
v2	String	Additional ID parameter. Length limit is 200 characters.	No	0
v3	String	Additional ID parameter. Length limit is 100 characters.	No	0
sum	Float	Payment amount in virtual currency. Separator is "." (0, 1, 2 characters after point)	Yes	100
date	String	Date in the following format YYYYDDMMHHMMSS	Yes	20060425180622
test	Integer	Accepted values: **test=1** - The system makes a test transaction. No payment is made. **test=0** - No test is run. This is a real transaction.	No	1
bonus	String	Promotional campaign is a secret word supplied by your account manager*	No	bonussum

Field Name	Type	Description	Required?	Example
md5	String	The string for unsanctioned access prevention	Yes	9286b1ff8c5226b666a20ddb4cc03c2b

* You can always find out more information about promotional campaigns from your Xsolla account manager.

Generating the md5 Signature

The signature is required for payment processing security. It is generated with an md5 hash algorithm from the string that is derived by concatenating the parameters specified below.

```
md5(<command><v1><id><secret_key>)
```

secret_key is the secret word that you chosen for your Merchant Account.

Example

The string for signature computing contains the following parameters:

```
secret key = "password"

command = pay

id = 7555545

v1 = demo

date=20060425180622
```

The string before applying MD5 hash would be:

```
paydemo7555545password
```

MD5 hash of this string would be:

```
9286b1ff8c5226b666a20ddb4cc03c2b
```

As a response to this request, your payment application should generate one of the following responses in XML:

If balance increase was successful:

```
<response>
  <id>7555545</id>
  <id_shop>1234</id_shop>
  <sum>10</sum>
  <result>0</result>
</response>
```

If balance increase was not successful:

```
<response>
  <id>7555545</id>
  <id_shop>1234</id_shop>
  <sum>10</sum>
  <result>1</result>
  <comment>Temporary database error</comment>
</response>
```

Result Field Values

Code	Description	Fatal?
0	Ok	No
1	Temporary error, retry request later	No
2	Invalid user ID	Yes
3	Invalid MD5 signature	Yes
4	Invalid request format (invalid amount, some parameters are missing)	Yes
5	Another error (to be described in *comment*)	Yes
7	Certain user's payment cannot be processed/denied due to technical reasons	Yes

Duplicate balance increase request should be responded correctly without an actual balance increase.

Sample Code

This sample code illustrates how to increase the balance. See Appendix A for sample code illustrating how to check the md5 signature and IP address.

```
<?php

$v1 = $_GET['v1'];
$sum = $_GET['sum'];
$id = $_GET['id'];

// making payment, returning it's id
$paymentId = pay($v1, $sum, $id);

// answer
// if payment successful
if ($paymentId)
        echo '<response>
        <id>' . $id . '</id>
        <id_shop>' . $paymentId . '</id_shop>
        <sum>' . $sum . '</sum>
        <result>0</result>
        <comment>Success</comment>
   </response>';
else
        echo '<response>
        <id>' . $id . '</id>
        <id_shop>' . $paymentId . '</id_shop>
        <sum>' . $sum . '</sum>
        <result>1</result>
        <comment>Temporarily database error</comment>
   </response>';

// This is pay function
function pay($v1, $sum, $id)
{
   // your code here

   return $paymentId;
}
```

Roll-Back Request

Cancel

This function is optional. Your Xsolla account manager can advise you on whether you need to implement payment roll-back. Our server sends payment details for roll-back to the Payment Script URL specified in the project details of your Xsolla Merchant Account. Payment details are sent by the GET method in CP1251 encoded XML and contain the parameters listed below.

Request Parameters

Field Name	Type	Description	Required?	Example
command	String	For the rollback request, the value must be **cancel**	Yes	
id	String	Order ID in Xsolla system passed via *id* parameter of *pay* request	Yes	7555545
md5	String	The string for unsanctioned access prevention	Yes	e9b9777e9c0a4595ad009eca90ba9977

Generating the md5 Signature

The signature is required for payment processing security. It is generated with an md5 hash algorithm from the string that is derived by concatenating the parameters specified below.

```
md5(<command><id><secret_key>)
```

secret_key is the secret word that you chosen for your Merchant Account.

Example

The string for signature computing contains the following parameters:

```
secret key = "password"

id = "7555545"

command = "cancel"
```

The string before applying MD5 hash would be:

```
cancel7555545password
```

MD5 hash of this string would be

```
e9b9777e9c0a4595ad009eca90ba9977
```

As a response to this request, your payment application should generate one of the following responses in XML:

If payment roll-back was successful:

```
<response>
  <result>0</result>
</response>
```

If payment roll-back was not successful:

```
<response>
  <result>2</result>
  <comment>this payment ID does not exist</comment>
</response>
```

Result Field Values

Result	Description
0	Payment indicated in the request was successfully cancelled
2	Payment indicated in the request was not found
7	Payment indicated in the request cannot be cancelled

Sample Code

This sample code illustrates how rollback the balance increase. See Appendix A for sample code illustrating how to check the md5 signature and IP address.

```
<?php

$id = $_GET['id'];

// canceling payment
$cancelResult = cancel($id);
// if successfuly canceled
if ($cancelResult)
   echo '<xml version="1.0" encoding="utf-8">
         <response>
               <result>0</result>
         </response>';
else
   echo '<xml version="1.0" encoding="utf-8">
         <response>
               <result>2</result>
               <comment>Payment with given ID does not exist</comment>
         </response>';

// This is cancel function
function cancel($id)
{
   // your code here

   return true;
}
```

Sample Code

This appendix contains sample code for performing an IP check and for checking the md5 signature.

Checking IP Address

```php
<?php
/**
* This is IP check function
*/

$isAllowed = checkIP($_SERVER['REMOTE_ADDR']);

/**
* Checks signature
*
* @return Boolean
*/
function checkIP($ip)
{
        $allowedIP = '94.103.26.178';

        return $ip === $allowedIP;
}
```

Checking MD5 Signature

```php
<?php
/**
* This is sign check function
*/

$command = $_GET["command"];
$v1 = $_GET["v1"];
$id = $_GET["id"];
$md5 = $_GET["md5"];
$key = 'secretKey';

$check = checkSignature($command, $v1, $id, $key, $md5);

/**
```

```
* Checks signature
*
* @return Boolean
*/
function checkSignature($command, $v1, $id, $key, $md5)
{
        return md5($command . $v1 . $id . $key) === $md5;
}
```