UPayCard Bitcoin

Contents

1. INTRODUCTION

UPayCard documentation provides access for the bitcoin functions and services implemented in UPayCard. Merchants can build their own custom applications, tools, and services to support their programs or components of their programs.

Merchant is free to choose any from the three available bitcoin payment options. Below is short description about each of one. See the relevant chapter for more information on each bitcoin payment option.

Bitcoin purchase

In this case the merchant sends all the encoded details to URL provided by UpayCard. He gets a response back with an UpayCard URL, where to redirect the customer. Merchant redirects the customer to UpayCard login page. Customer needs to login to UpayCard. After login customer is redirected to the confirmation page where he can select UpayCard account from which to pay and confirm or cancel his purchase. When purchase is finished merchant and customer gets an email with information about the purchase.

Bitcoin direct API transfer

In this case the merchant sends all the details through API to UpayCard. In order to finish transfer two calls to UpayCard API need to be done. During first one transfer from customers UpayCard account to merchants UpayCard account is being initialized. In the response request details are provided. Merchant displays requested token code for user. After key code (for provided token code) is filled merchant finishes transfer with request to the UpayCard API with needed data. When transfer is finished merchant and customer gets an email with information about the transfer.

Bitcoin get address

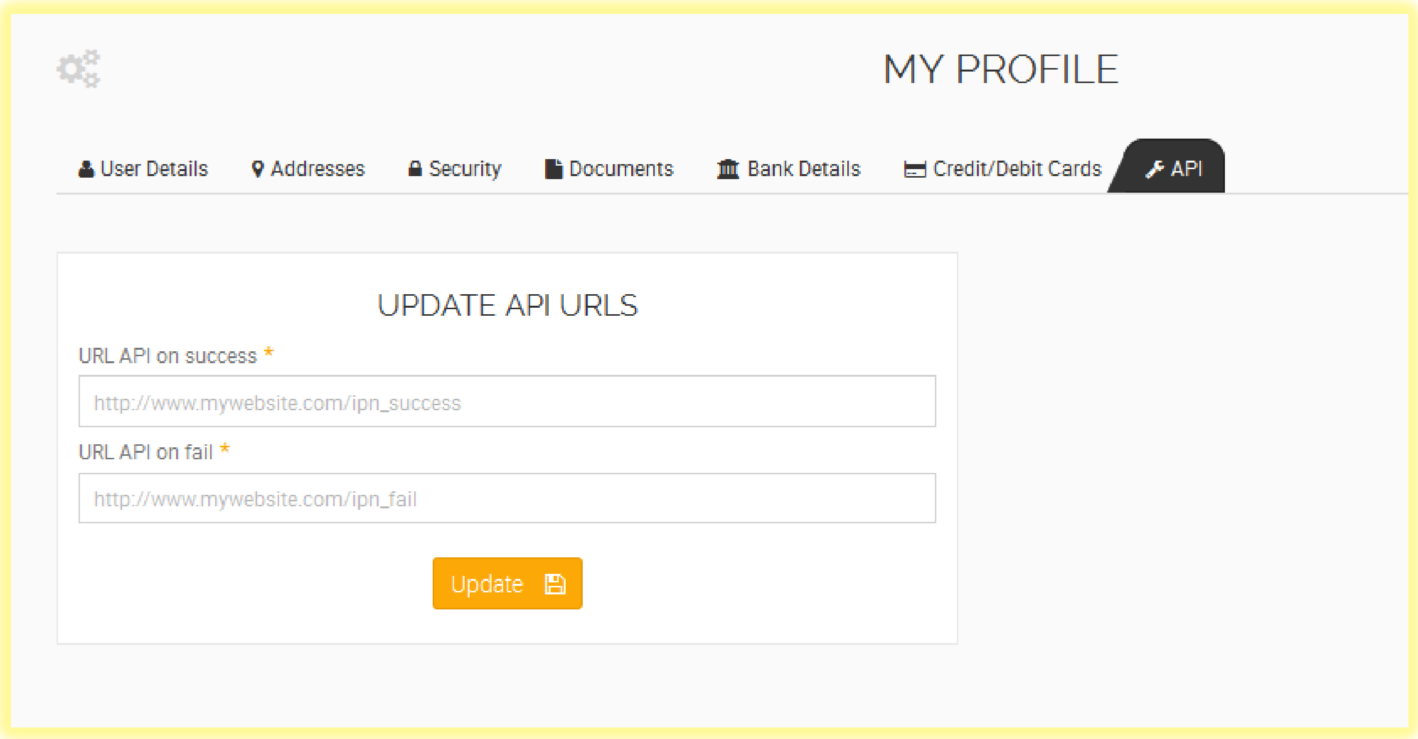
In this case the merchant sends all the details through API to UpayCard. In response bitcoin amount and bitcoin address are provided. Merchant shows these details to the customer, where the customer should send the money.

2. ADMINISTRATION

This chapter describes bitcoin administration through UpayCard website. Merchant can setup URL for IPN (instant payment notification) in order to know when bitcoin transfer is completed.

2.1. Setup URLs for IPN

IPN – instant payment notification. Whenever bitcoin transfer is completed or canceled IPN is sent to the merchant provided URL. Merchant can setup different URL for successfully completed or failed transfer if needed. Of course same URL can be used for both cases. URL setup page can be found under profile tab “API” (see Figure 1: URLs for IPN setup page). Note that POST will be sent to the provided URLs. Example of data sent can be found on APPENDIX B: Bitcoin direct API transfer IPN catcher example and APPENDIX C: Bitcoin Purchase IPN catcher example.



MY PROFILE

UPDATE API URLS

URL API on success *

http://www.mywebsite.com/ipn_success

URL API on fail *

http://www.mywebsite.com/ipn_fail

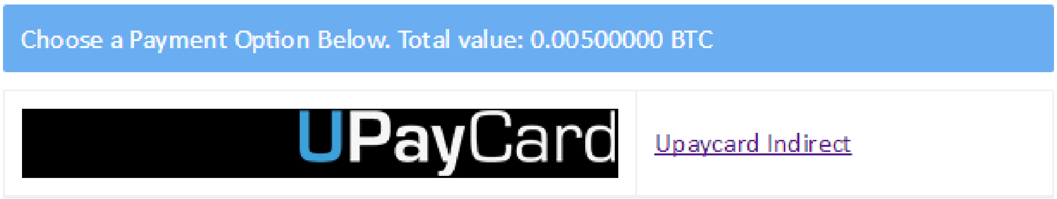
Figure 1: URLs for IPN setup page

3. BITCOIN PURCHASE

3.1. Purchase Flow

Purchase flow step by step:

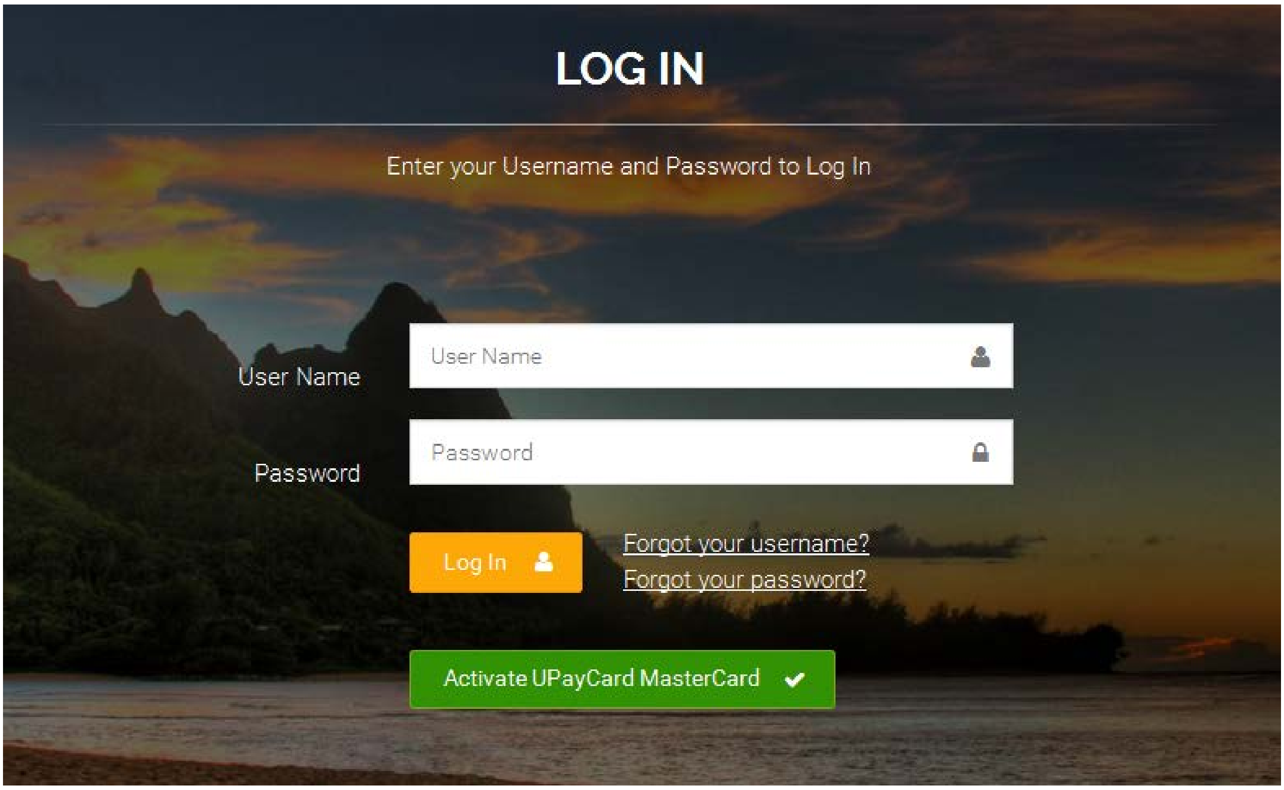
1. Customer comes to the Merchant payments options page (see Figure 2: merchant page with payment options) and chooses to pay with UpayCard.



Choose a Payment Option Below. Total value: 0.00500000 BTC

Figure 2: merchant page with payment options

2. In case of successfully initialised purchase UpayCard redirects user to UpayCard login screen (see Figure 3: UpayCard login page). Customer needs to login to UpayCard. After login customer is redirected to “Purchase Confirmation” page (see Figure 4: Purchase confirmation page) where he can select UpayCard account from which to pay and confirm or cancel his purchase. When customer press either “Confirm” either “Cancel” purchase is being processed (see Figure 5: Purchase Information page – processing). After processing respectively is being showed canceled (see Figure 6: Purchase Information page – cancelled) or completed (see Figure 7: Purchase Information page - completed) purchase.



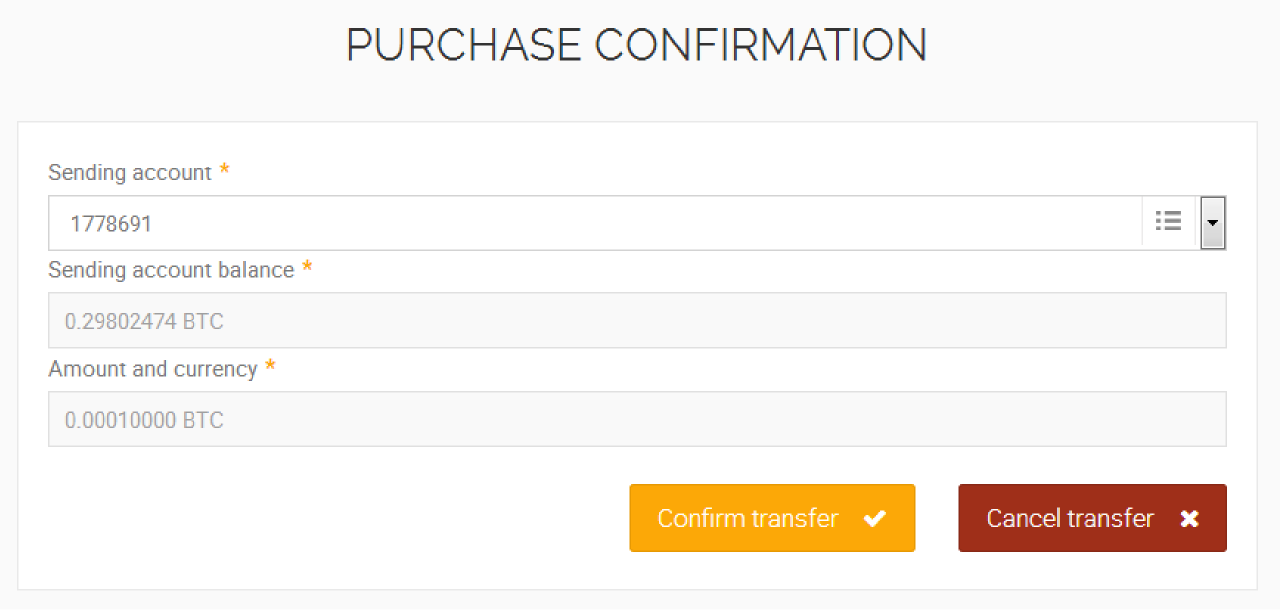
LOG IN

Enter your Username and Password to Log In

User Name: User Name

Password: Password

Figure 3: UpayCard login page



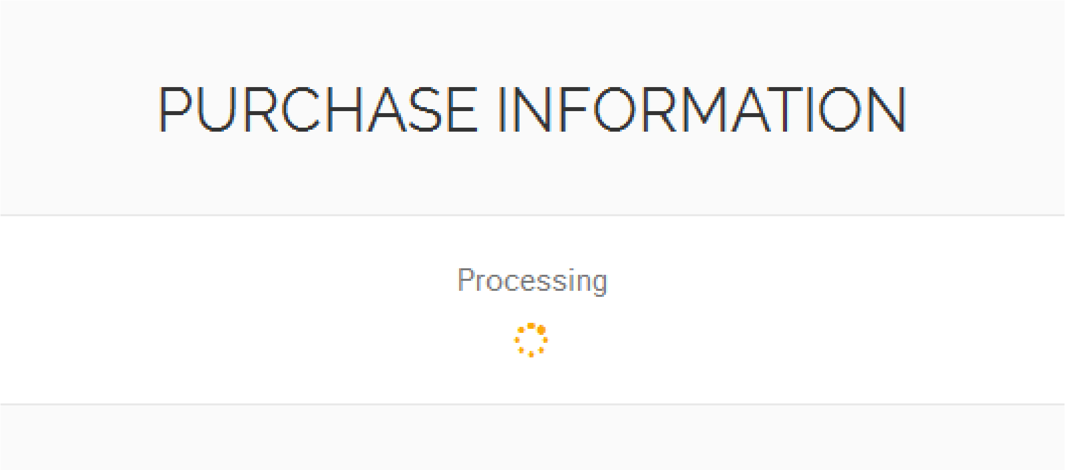
PURCHASE CONFIRMATION

Sending account *: 1778691

Sending account balance *: 0.29802474 BTC

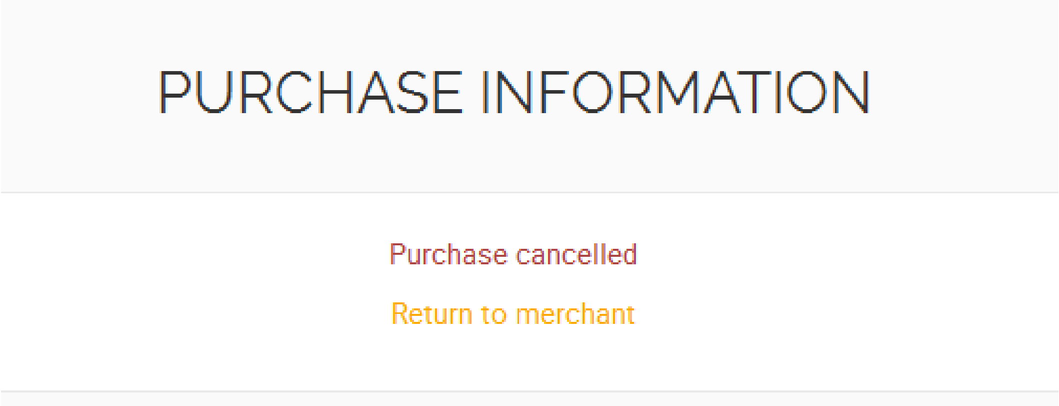
Amount and currency *: 0.00010000 BTC

Figure 4: Purchase confirmation page


PURCHASE INFORMATION

Processing

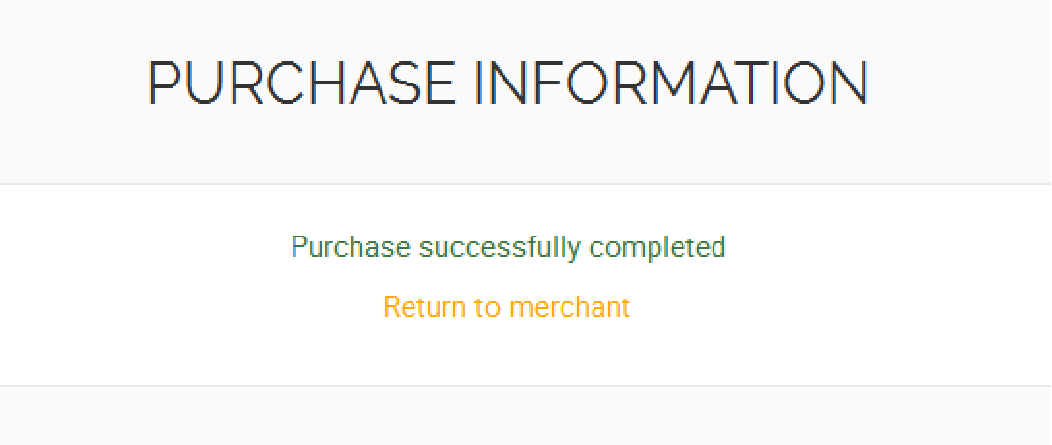
Figure 5: Purchase Information page – processing


PURCHASE INFORMATION

Purchase cancelled

Return to merchant

Figure 6: Purchase Information page – cancelled


PURCHASE INFORMATION

Purchase successfully completed

Return to merchant

Figure 7: Purchase Information page - completed

3. Lastly customer press “Return to merchant” and is redirected to the merchant provided URL (which was provided in parameters during request in first step).
4. If merchant provided “url_api_on_success” and “url_api_on_fail” he will receive IPN after purchase was failed or completed. Note that UpayCard tries to send IPN 10 times. After each failed time, next IPN sending is delayed for some time. See IPN catcher example for bitcoin purchase APPENDIX C: Bitcoin Purchase IPN catcher example and available purchases statuses APPENDIX D: Bitcoin Purchase statuses.

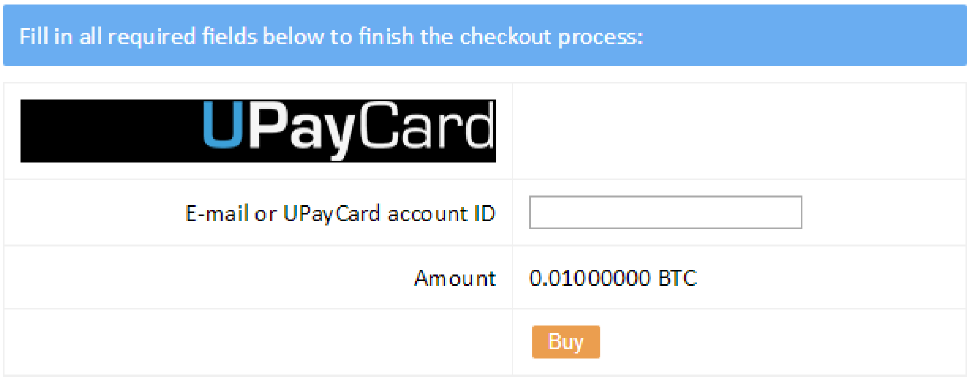
3.2. Programming in PHP

Example below provides Purchase Form for testing. Merchant should implement its form as needed. Then after user press button collect all data, encode it and redirect to UpayCard. Save provided code to the file “gui_purchase_example.php” for testing.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Purchase Test Form</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <style>
            .required {
                color: red;
            }
            input { width: 300px; }
        </style>
    </head>
    <body>
        <h1>Purchase Test Form</h1>

        <form method="post" action="">
            <table>
                <tr>
                    <td>Receiver account <span class="required">*</span></td>
                    <td><input name="receiver_account" value="" placeholder="Your
account ID" required /></td>
                </tr>
                <tr>
                    <td>Amount <span class="required">*</span></td>
                    <td><input name="amount" value="" placeholder="amount"
required /></td>
                </tr>
                <tr>
                    <td>Currency <span class="required">*</span></td>
                    <td><input name="currency" value="" placeholder="BTC" required
/></td>
                </tr>
                <tr>
                    <td>Order ID <span class="required">*</span></td>
                    <td><input name="order_id" value="<?php echo
uniqid('orderId_'); ?>" required /></td>
                </tr>
                <tr>
                    <td>Success URL <span class="required">*</span></td>
                    <td><input name="url_user_on_success" value=""
placeholder="http://www.mysite.com/success_page_for_user" required /></td>
                </tr>
                <tr>
                    <td>Fail URL <span class="required">*</span></td>
                    <td><input name="url_user_on_fail" value=""
placeholder="http://www.mysite.com/failed_page_for_user" required /></td>
                </tr>
                <tr>
                    <td></td>
                    <td><input type="submit" value="Submit" /> </td>
                </tr>
            </table>
        </form>
    </body>
</html>


<?php
    $USE_SANDBOX = 1;

    $url_sandbox =
"https://sandboxauser.upaycard.com/en/purchase/initiatepurchase/?data=";
    $url_live = "https://user.upaycard.com/en/purchase/initiatepurchase/?data=";

    if (isset($_POST) && !empty($_POST)) {
        // create hash add it to the params
        $strToSign = $_POST['receiver_account'] . $_POST['amount'] .
$_POST['order_id'] . $_POST['currency'];
        $hash = md5($strToSign);
        $_POST['hash'] = $hash;

        // endoced params
        $encoded = base64_encode(json_encode($_POST));
        $url = ($USE_SANDBOX == 1 ? $url_sandbox : $url_live) .
urlencode($encoded);

        // redirect user
        header("Location:" . $url);
    }
?>
```

4. BITCOIN DIRECT API TRANSFER

Bitcoin direct API transfer is done in merchant web page (the customer will only see merchant web page). Merchant will need to make two calls to UpayCard API. First one in order to initialize transfer from customers UpayCard account to merchants UpayCard account. Customer enters amount and his UpayCard username or account number (see Figure 9: merchant web page for customer data submission), then merchant collects all data and makes call to API function *initialize_transfer*. In case of success in the response request details are provided (initialized transfer hash and token code for the customer to provide).

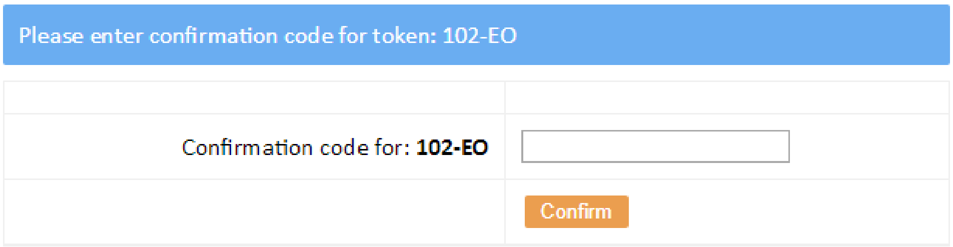


Fill in all required fields below to finish the checkout process:

E-mail or UPayCard account ID	
Amount	0.01000000 BTC

Figure 3: merchant web page for customer data submission

Merchant displays requested token code for customer. Customer need to enter requested token codes key code and press the button (see Figure 10: merchant web page for key code entering).



Please enter confirmation code for token: 102-EO

Confirmation code for: **102-EO**	

Figure 4: merchant web page for key code entering

After key code is filled and submitted merchant makes second call to API function *finish_transfer* in order to finish transfer. When transfer is finished merchant and customer gets an email with information about the transfer.

Below are provided API calls requests and responses information that merchant should use in order to integrate bitcoin transfer through direct API.

4.1. Initialize transfer

Merchant initializes transfers through UpayCard API. In case of success in response data about the initialized transfer is provided. Pay attention to hash and token code. Hash and customer entered key code (for provided token code) need be sent to UpayCard API in order to finish transfer.

URL	http://api.upaycard.com/api/merchant/v/1.0/function/initialize_transfer

4.1.1. Initialize transfer request

The box below lists all fields that could be included in the call request.

KEY	M	TYPE	LENGHT	DESCIPTION
receiver_account	Y	N	11	Your account ID for receive transfer
sender	Y	AN	100	Here can be username or user email or account number from which made transfer. If user did not have requested currency account or have several accounts then there must be account number
amount	Y	N	10,2 / 10,8	For currencies like EUR or USD use 10,2. For bitcoin 10,8 can be used.
currency	Y	A	3	ISO 4217, for bitcoin use BTC
order_id	Y	AN	30	Unique identification of request
description	Y	AN	100	Transfer description
account_by_user_country	N	N	1	If parameter provided and its value = 1, then if user is from Europe, transfer will be prepared to merchant EUR account, otherwise to merchant USD account. If merchants EUR or USD account not found – transfer will be prepared to the account provided in the *receiver_account* parameter.
key	Y	AN	16	Merchant API Key – provided by UPayCard
ts	Y	N	10	Request timestamp
sign	Y	AN	32	See APPENDIX A: SIGN GENERATION

4.1.2. Initialize transfer JSON request sample

```
{
   "receiver_account":"1000001",
   "sender":"user",
   "amount":0.01,
   "currency":"BTC",
   "order_id":"15_20151110080801",
   "description":"Payment for order #15",
   "account_by_user_country":0,
   "key":"_MERCHANT_KEY_",
   "ts":_TIMESTAMP_,
   "sign":"_SIGN_"
}
```

4.1.3. Initialize transfer response

KEY	M	TYPE	LENGHT	DESCIPTION
status	Y	AN	10	success or error
code	N	N	3	See APPENDIX E: Transaction Codes
msg	Y	AN	255	See APPENDIX E: Transaction Codes
description	N	AN		Detailed explanation of error
order_id	N	AN	100	UniqueID of request
hash	N	AN		
token_number	N	AN	10	

4.1.4. Initialize transfer JSON response sample

```
{
    "status":"success",
    "msg":"Transfer initialized.",
    "order_id":"15_20151110080801",
    "hash":"a64fb511c885f9aeff211f7bfefc5648",
    "token_number":"48-QI"
}
```

4.2. Finish transfer

When transfer is initialized it need to be finished. Otherwise transfer will never be processed. In order to finish prepared transfer from any user to merchant account user must enter his key code for given token_number. After key code is entered merchant collects all need data and makes call to UpayCard API.

URL	**https://api.upaycard.com/api/merchant/v/1.0/function/finish_transfer**

4.2.1. Finish transfer request

KEY	M	TYPE	LENGHT	DESCIPTION
receiver_account	Y	N	11	Your account ID for receive transfer
hash	Y	AN		hash code given in InitializeTransfer success response
token_number	Y	AN	10	token_number given in InitializeTransfer success response
token_code	Y	N	6	User’s entered Key Code for given token_number
account_by_user_country	N	N	1	If *InitializeTransfer* was called with this param (value=1), then add this param to the *FinishTransfer* with value=1
key	Y	AN	16	Merchant API Key – provided by UPayCard
ts	Y	N	10	Request timestamp
sign	Y	AN	32	See APPENDIX A: SIGN GENERATION

4.2.2. Finish transfer JSON request sample

```
{
   "receiver_account":"1000001",
   "hash":"5640bf0cb3e5c",
   "token_number":"48-QI"
   "token_code":"123456",
   "account_by_user_country":0,
   "key":"_MERCHANT_KEY_",
   "ts":_TIMESTAMP_,
   "sign":"_SIGN_"
}
```

4.2.3. Finish transfer response

KEY	M	TYPE	LENGHT	DESCIPTION
status	Y	AN	10	success or error
code	N	N	3	See APPENDIX E: Transaction Codes
msg	Y	AN	255	See APPENDIX E: Transaction Codes
description	N	AN		Detailed explanation of error
transaction_id	N	N	11	
order_id	N	AN	100	UniqueID of request

4.2.4. Finish transfer JSON response sample

Success response

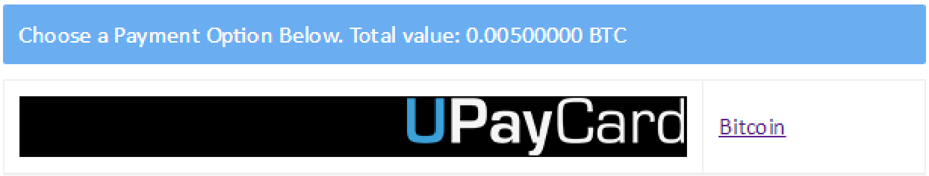
```
{
   "status":"success",
   "code":000,
   "msg":"Transaction successfully completed",
   "transaction_id":"100687",
   "order_id":"15_20151110080801",
}
```

Failed response

```
{
   "status":"error",
   "code":512,
   "msg":"Wrong Key Code provided",
}
```

5. BITCOIN GET ADDRESS

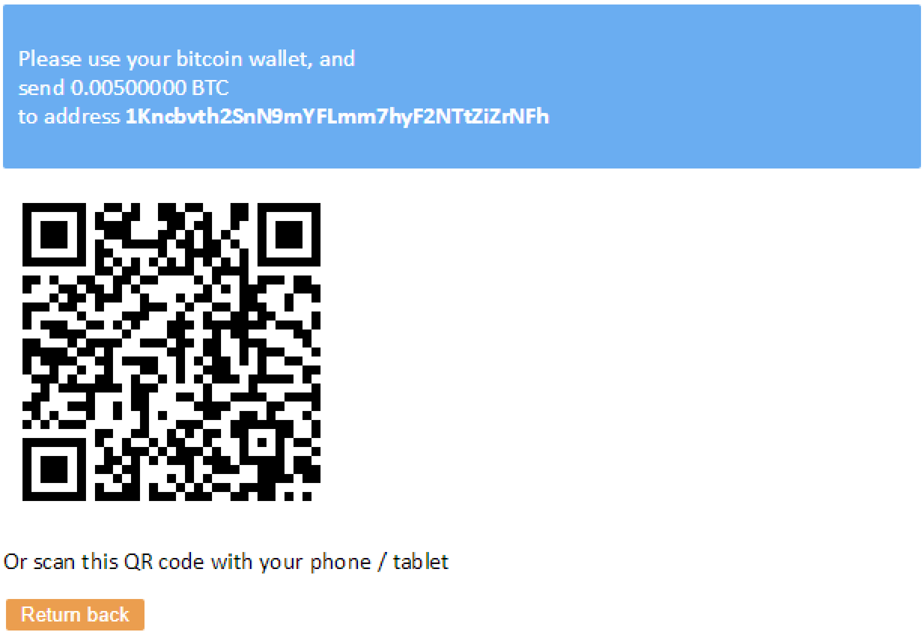
Merchant sends all the details through API to UpayCard (the customer will only see merchant web page). Merchant provides form where transfer amount can be entered or prefilled (See Figure 11: merchant web page form with transfer details).



Choose a Payment Option Below. Total value: 0.00500000 BTC

Figure 5: merchant web page form with transfer details

In response bitcoin amount and bitcoin address are provided. Merchant shows these details to the customer, where the customer should send the money (see Figure 12: merchant web page with bitcoin amount and address displayed)



Please use your bitcoin wallet, and
send 0.00500000 BTC
to address **1Kncbvth2SnN9mYFLmm7hyF2NTtZiZrNFh**

Or scan this QR code with your phone / tablet

Figure 6: merchant web page with bitcoin amount and address displayed

5.1.1. Load from bitcoin request

URL	https://api.upaycard.com/api/merchant/v/1.0/function/load_from_bitcoin

KEY	M	TYPE	LENGHT	DESCIPTION
account_id	Y	N	11	
amount	Y	N	10,2 / 10,8	For currencies like EUR or USD use 10,2. For bitcoin 10,8 can be used.
currency	Y	A	3	ISO 4217, for bitcoin use BTC
key	Y	AN	16	Merchant API Key – provided by UPayCard
ts	Y	N	10	Request timestamp
sign	Y	AN	32	See APPENDIX A: SIGN GENERATION

5.1.2. Load from bitcoin JSON request sample

```
{
   "account_id":"1824406",
   "amount":"0.01",
   "currency":"BTC",
   "key":"_MERCHANT_KEY_",
   "ts":_TIMESTAMP_,
   "sign":"_SIGN_"
}
```

5.1.3. Load from bitcoin response

KEY	M	TYPE	LENGHT	DESCIPTION
status	Y	AN	10	success or error
msg	Y	AN	255	See APPENDIX E: Transaction Codes
description	N	AN		
funds_loads_id	Y	N	11	
bitcoin_amount	Y	N	10,8	
bitcoin_address	Y	AN	35	

5.1.4. Load from bitcoin JSON response sample

```
{
   "status":"success",
   "msg":"Transaction in progress",
   "funds_loads_id":"68044",
   "bitcoin_amount":"0.09791538",
   "bitcoin_address":"1KmJT4rHiCogXyzqCTEMsXLSBDG46ypTN8",
}
```

6. APPENDIX A: Sign Generation

Sign of request, it is MD5 hash of keys, values and secret.

Example of Sign generation:

MD5("key1:value1: key2: value2:...:key:_MERCHANT_KEY_:ts:_TIMESTAMP_:_SECRET_")

Example in PHP:

```
function _sign($params)
{
    $strToSign = '';
    $params['key'] = '_MERCHANT_KEY_';
    $params['ts'] = time();
    foreach ($params as $k => $v)
        if($v !== NULL)
        $strToSign .= "$k:$v:";
    $strToSign .= '_MERCHANT_SECRET_';

    $params['sign'] = md5($strToSign);
    return $params;
}
```

7. APPENDIX B: Bitcoin direct API transfer IPN catcher example

```
if (!empty($_POST)) {
    $type           = $_POST['type'];             // 'api_btc2a'
    $status         = $_POST['status'];           // 'success' or 'fail'
    $order_id       = $_POST['order_id'];         // order ID
    $account_id     = $_POST['account_id'];       // account ID
    $settled_amount = $_POST['settled_amount'];   // settled amount
    $currency       = $_POST['currency'];         // 'BTC'
    $transaction_id = $_POST['transaction_id'];   // transaction ID
    if (!empty($_POST['sign'])) {
        $sign = $_POST['sign'];
        $arrToHash = [];
        ksort($_POST);

        foreach ($_POST as $field => $value) {
            if ($field == 'sign' || $value === null) {
                continue;
            }

            $arrToHash[] = $field . ':' . $value;
        }

        $hash = md5(implode(':', $arrToHash) . ':' . '_MERCHANT_SECRET_');
        if ($hash == $sign) {
            // sign correct - any needed actions further
        }
        else {
            // wrong sign - any needed actions further
        }
    }
    else {
        // no sign - any needed actions further
    }

}
```

8. APPENDIX C: Bitcoin Purchase IPN catcher example

```
if (!empty($_POST)) {
    $status_id = $_POST['status_id'];           // 7
    $reference_id = $_POST['reference_id'];     // 'ap-57fcb50701182'
    $order_id = $_POST['order_id'];             // 'my_order_81815'
    $data_amount = $_POST['data_amount'];       // 1.01
    $currency = $_POST['currency'];             // 'USD'
    $transaction_id = $_POST['transaction_id']; // transaction ID
    $first_name = $_POST['first_name'];         // first name
    $last_name = $_POST['last_name'];           // last name
    $key = $_POST['key'];                       // merchant key
    $time = $_POST['ts'];                       // UNIX timestamp

    // note that transaction ID will be not empty only if status is 8 or 9

    if (!empty($_POST['sign'])) {
        $sign = $_POST['sign'];
        $arrToHash = [];
        ksort($_POST);

        foreach ($_POST as $field => $value) {
```

```
                    if ($field == 'sign' || $value === null) { continue; }
                    $arrToHash[] = $field . ':' . $value;
                }

                $hash = md5(implode(':', $arrToHash) . ':' . _MERCHANT_SECRET_);

                if ($hash == $sign) {
                    // sign correct - any needed actions further
                }
                else {
                    // wrong sign - any needed actions further
                }
            }
            else {
                // no sign - any needed actions further
            }
        }
```

9. APPENDIX D: Bitcoin Purchase statuses

Type ID	Description
1	Created
2	Logged in
4	Processing
7	Canceled
8	Failed
9	Successful

10. APPENDIX E: Transaction Codes

Code	Description
000	Transaction successfully completed
100	Load limit exceeded (value of transactions)
101	Load limit exceeded (number of transactions)
102	Transfer limit exceeded (maximum transaction amount allowed)
103	Transfer limit exceeded (value of transactions)
104	Transfer limit exceeded (number of transactions)
105	Withdrawal limit exceeded (value of transactions)
106	Withdrawal limit exceeded (number of transactions)
107	Withdrawal limit exceeded (maximum transaction amount allowed)
108	Card throughput limit exceeded (must provide KYC documents)
110	Transfer restricted
111	Load restricted

112	Recipient cannot accept transfers
113	Account balance exceeded
114	Operation is not allowed
200	Insufficient funds
300	Card is inactive
400	Could not find currency rate.
500	Invalid signature
501	Error creating session
502	Operation is not allowed
503	Missing field
504	Field format error
505	Invalid receiver account
506	User not found
507	Invalid currency code
508	Invalid sender account
509	Define sender account
510	Duplicate order_id
511	Initialized transaction not found
512	Wrong Key Code provided
513	Transfer request already confirmed
514	Transaction not found
515	Transaction cannot be refunded
516	Cannot refund this amount
517	Your transaction request was sent to our Bank for processing
520	Invalid username provided
521	Invalid account provided
522	Invalid data provided
523	Invalid external card id provided
524	Wrong verification amount
525	Verification attempts limit reached
526	Verification failed
527	Invalid card status
528	S3D cards not supported
529	Configuration error
999	Unknown error

11. External Bitcoin Load in Sandbox Test Environment

Merchant will be provided with the test credentials for the merchant himself, three personal sponsored accounts and one additional simple user (name contains “BTC”), which will have only Bitcoin currency account with balance of 2 Bitcoins.

In order to test external bitcoin load fully (“Bitcoin Get Address" function, see page 15) in test environment:

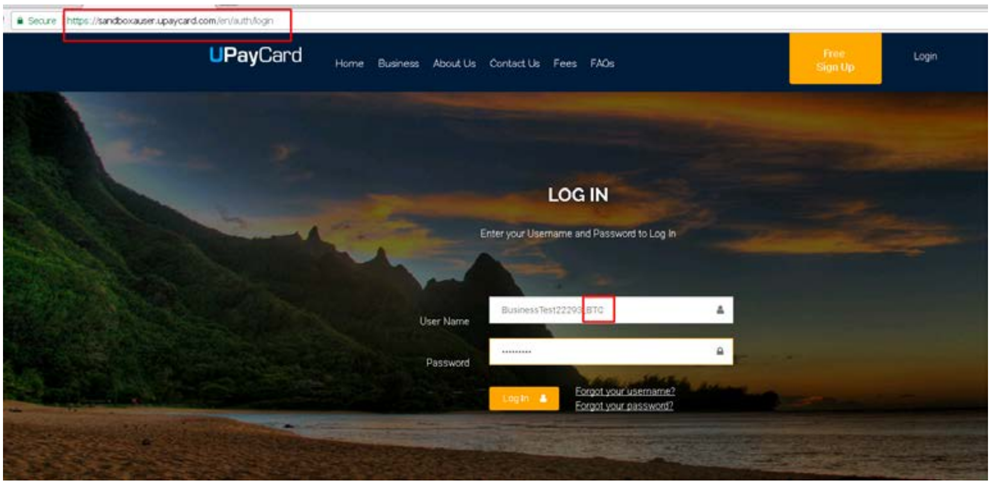
11.1. On Merchants Payment Page

1. Customer goes to Merchant payment page and choose to option: Pay with Bitcoin.
2. Customer enters how many bitcoins he wants to send to the merchant.
3. Merchants sends an API request to hit bitcoin provider, which is UPayCard and request an address he can give to his customer. UPayCard sends back an API respond with a Bitcoin Address.
4. Merchant displays the bitcoin address to the user and asks him to send the bitcoin from his bitcoin provider to this address. This provider can be any in the world! <u>BUT FOR THIS INTEGRATION USING OUR SANDBOX, YOU CAN ONLY SEND TO THIS ADDRESS FROM 1 BITCOIN PROVIDER. (UPAYCARD).</u>

11.2. Customer Sends the Bitcoin from His Wallet

5. Customer logs into his bitcoin provider and sends the relevant bitcoin amount to the bitcoin address the merchant gave him:

a) For the purpose of checking the integration, then the customer must use bitcoin user wallet account (BTC USER) Which we have created in Sandbox. So please log into this account (credentials provided).
b) Navigate to Bitcoin Transfer section, then to Withdraw (Send).
c) Enter the bitcoin address the customer were given from the merchant.
d) Send the amount to the address. See screenshots bellow:



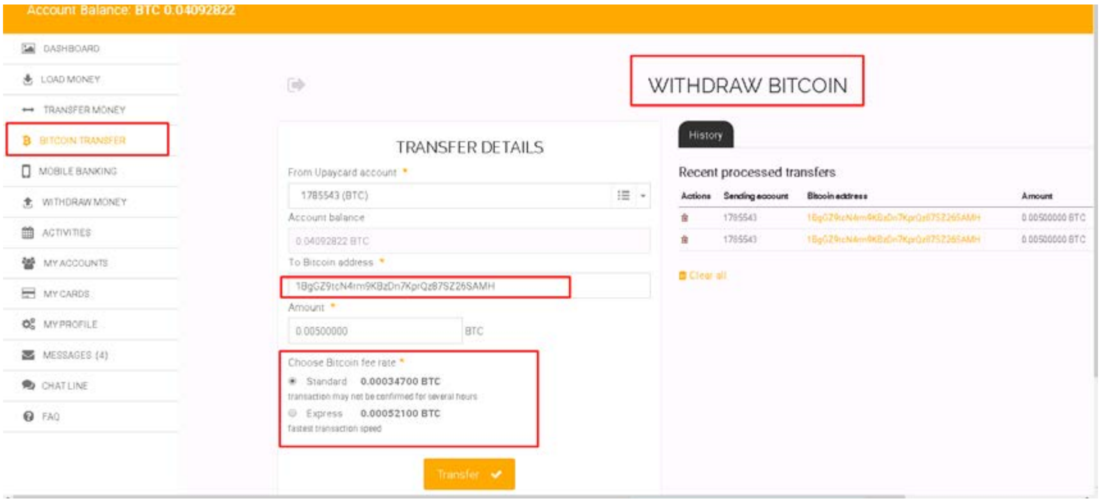
Secure https://sandboxauser.upaycard.com/en/auth/login

LOG IN

Enter your Username and Password to Log In

User Name: BusinessTest22293BTC

Password: ••••••••••



Account Balance: **BTC 0.04092822**

WITHDRAW BITCOIN

TRANSFER DETAILS

From Upaycard account *: 1785543 (BTC)

Account balance: 0.04092822 BTC

To Bitcoin address *: 1BgGZ9tcN4rm9KBzDn7KprQz87SZ26SAMH

Amount *: 0.00500000 BTC

Choose Bitcoin fee rate *

● Standard **0.00034700 BTC**
transaction may not be confirmed for several hours

○ Express **0.00052100 BTC**
fastest transaction speed

Recent processed transfers

Actions	Sending account	Bitcoin address	Amount
🗑	1785543	1BgGZ9tcN4rm9KBzDn7KprQz87SZ26SAMH	0.00500000 BTC
🗑	1785543	1BgGZ9tcN4rm9KBzDn7KprQz87SZ26SAMH	0.00500000 BTC

11.3. Merchant checks his Wallet account

6. Merchant will get a notification on the notification URL that he was given.
7. When Merchant gets this notification, then he can give the purchased product to his customer on his website.
8. Merchant can also log into UpayCard (sandbox for now) and check that he received the amount.

*Please note, that even though Sandbox is a test environment, Bitcoin addresses generated in sandbox are real, therefore please **DO NOT** use these addresses anywhere else, except **ONLY** in UPayCard Sandbox environment, otherwise real bitcoins will be lost.*