

Forum:[Hack Forums](#)

Thread: [\[PHP\] Accepting Bitcoin on Your Site](#)

c x y k - March 13, 2015, 3:31 pm

Hello!

I recently saw a thread about a user wanting help regarding how to accept bitcoin payments on their site using php. I thought that since I was in need of so the other day, I'd quickly whip up a class.

I know this isn't perfect, but it should be enough to get somebody going. If you're in need of any assistance, feel free to reply.

For the following demonstration, we will need six files in the following format:

```
| index.php
| module | .htaccess
| module | BlockPay.php
| module | init.php
| module | KondoSQL.php
| module | Validation.php
```

I will write the guide in order of these files. If you do not wish to read the guide and just want the code, please view <https://github.com/TaylerKing/BlockPay/blob/master/dist/BlockPay.php>

Index - Here we will create and store transactions

First off, we need to require our init file. We can do this via the following:

PHP Code

```
<?php
require_once "module/init.php"
```

Within our init file we will store the database credentials and connection information, and also the `spl_autoload_register` which will automatically register classes we call.

Now we've got the init file included via a require, we need to create a new instance of the `BlockPay.php` class. We can do so like this:

PHP Code

```
<?php
$blockpay = new BlockPay("http://***.***.***.***/BlockPay/example/callback.php?hash=",
"15ks4PWRUyTqk89szb9q5baji8Y6pTxtYB", "ffa57c77-7921-4fc6-bfbb-d134b57c5c7b");
```

Here we are initiating the `BlockPay.php` class. We then store it in the variable `$blockpay`. The arguments are as follows:

PHP Code

```
<?php
[Callback URL - This is what the blockchain api will call upon payment being received,
```

Receiving Address - This is what blockchain will forward the payment to,
API Key - This is optional. If you have an official API Key issued via a member of staff at blockchain , you can use this to bypass request limits]

Now we've got this created, we need to go ahead and generate our new address. For the sake of my mortality and keeping this tutorial as short as possible, I will skip checking whether or not methods have failed. If you want to see the full code which incorporates these checks, please refer to <https://github.com/TaylerKing/BlockPay/blob/master/example/index.php>.

To generate an address, it is a simple line. You can use multiple lines also, if you want to declare variables for this address method.

PHP Code

```
<?php
$hash = hash('sha256', uniqid());
$time = time();
$blockpay->getAddr($hash . "|" . $time);
```

With the above, we are doing the following:

- > Generating a unique and random sha256 hash.
- > Storing the current server time to a variable.
- > Generating an address with the additional data of "*randomhash*|*currenttime*"

If we take a look at the code behind this method, we can see what's going on:

PHP Code

```
<?php
public function getAddr($data = null) {
    $this->_error = false;
    $callback = urlencode($this->_callback . (is_null($data) ? "" : $data));
    $resp = file_get_contents("https://blockchain.info/api/receive?"
        . "method=create&"
        . "address=" . $this->_addr . "&"
        . "anonymous=false&"
        . "callback=" . $callback
        . (is_null($this->_apikey) ? "" : "&api_key=" . $this->_apikey)
    );
    if($resp === false)
        $this->_error = true;
    else {
        $resp = json_decode($resp);
        $this->_data = [$resp->input_address, $resp->destination, urldecode($callback)];
    }
}
```

Explanation: First of all, we're encoding the callback url and adding additional data. This ensures that the url is encoded if it contains get parameters, instead of breaking the api values. We then call the file_get_contents method which adds the correct get parameters to the URL and it submits them for the server to request; after completion returning an array of data.

So, we've got our address. Nearly done, right? Not quite. We now need to get the amount of the product you want in bitcoin, so we call a simple function:

PHP Code

```
<?php
$amount = $blockpay->getValue(10);
```

All this function does is call a request to <https://www.bitstamp.net/api/ticker/>. It then gets the vwap value (Average price of bitcoin within the last 24 hours) - one is divided by this amount and then multiplied by the amount, therefor leaving you with the value of the \$USD you provided in bitcoin.

We've got everything we now need to receive a transaction. We do, however, need to log this - so we simple put the information in our database:

PHP Code

```
<?php
$sql->insert("temp", array(
    "address" => $data[0],
    "amount" => $amount * 1000000,
    "paid" => 0,
    "confirmed" => 0,
    "txid" => "null",
    "hash" => $hash,
    "time" => $time));
```

This should be pretty self-explanatory. It inserts the array values into a table called "temp". If you wish to utilize the same method, please take a look at <https://github.com/TaylerKing/KondoSQL>

For the final part of the index, we simply have to tell the user the amount and address for the transaction. We can do so with the following:

PHP Code

```
<?php
echo "Hello, and welcome to our shop. Please send " . $amount . " to " . $data[0];
```

The above will produce something looking along the lines of this [\(Click to View\)](#)

Hopefully you don't just output the raw value like I did, as bitcoin amounts can only go up to 8 decimal points. I suggest you round the amount when notifying your users.

.htaccess - Quick fix

In this file, put the line 'Deny from all' as the contents. Shouldn't need an explanation; but I'll explain anyway. It stops users from accessing the folder and any files within, but not the server itself.

BlockPay.php

I'm not going to post much about this class. I've documented it (kind of) at <https://github.com/TaylerKing/BlockPay/blob/master/example/module/BlockPay.php>, so that will be more help combined with the above information I've supplied you with.

init.php - core class loading

This file is an incredibly small file, with only four lines. It could be less if we wanted, although it's this for an example. Here'e the contents of it:

PHP Code

```
<?php
const DB = ["localhost", "root", "", "pdo"];
spl_autoload_register(function($a) {
    require_once dirname(__FILE__) . "\\ " . $a . ".php";
});
?>
```

In the KondoSQL.php class, we reference a constant. This constant is known as 'DB'. It's an array that contains connection information for the database, as you can quite clearly see.

Next up is spl_autoload_register. This is called when a class object is called, such as "new Test()". \$a would be 'Test', and as you can see it requires the class based upon it's name.

KondoSql.php - Validation.php

You can find more information on these two classes [here](#) and [here](#).

I'm incredibly sorry if this thread was extremely long. I tried to keep to the minimum as to not bloat it with information. If you need any assistance or wish to put forward feedback, please do so by responding with a reply. Before you ask why I used file_get_contents over cURL - I've found that it has a speed increase for blockchain, and if people were learning the basics of PHP, it's just a quicker way of achieving the same with cURL in that sense.

Cheers! (:

Ego Inc - March 13, 2015, 5:30 pm

Nicey done, however I would say, in my personal opinion, the code is spread out over too many files. However the code is nice tho, good job~

c x y k - March 14, 2015, 12:12 am

Thanks for the feedback.

It actually only requires three files - index, bitpay and callback, but I wanted to add a little more for additional features.

I'll be sure to add the notice Yani.

Cheers.

c??? - March 15, 2015, 8:54 pm

Nicely well written done tutorial! Good job.

c x y k - March 16, 2015, 12:16 am

(Mar 15, 2015 08:54 PM)Jerry_ Wrote: [\[CITE\]](#) Nicely well written done tutorial! Good job. Thanks for the feedback, bud! (:

If anybody requires assistance, ask away.