Bitcoin and Altcoin Wallets 5.0.16 reference manual

dashed-slug info@dashed-slug.net

Abstract

Integrating cryptocurrency wallets into your WordPress site, using the **Bitcoin and Altcoin Wallets** plugin, version **5.0.16**.

At a glance

Offer your users custodial cryptocurrency wallets, backed by full nodes that you control.

Use **Bitcoin and Altcoin Wallets** to maintain cryptocurrency balances for the users of your WordPress site.

Install *coin adapters* to support more coins or *app extensions* to provide additional cryptocurrency functionalities, such as payment gateways for products and services.

Pricing

The core wallets plugin is free.

The coin adapters require free signup to dashed-slug.net.

Access to premium app extensions, updates and support require a paid subscription of $9.95 per 3 months.

If you choose to cancel your subscription you can continue using the plugins as long as you like, but you will lose access to updates and premium support.

Disclaimer

By continuing to use the *Bitcoin and Altcoin Wallets* plugin, you indicate that you have understood and agreed to this disclaimer.

You accept all responsibility for handling the account balances for all your users. Under no circumstances is dashed-slug.net or any of its affiliates responsible for any damages incurred by the use of this plugin.

Every effort has been made to harden the security of this plugin, but its safe operation depends on your site being secure overall.

You, the administrator, must take all necessary precautions to secure your WordPress installation before you connect it to any live wallets.

You are strongly advised to take the following actions (at a minimum):

- educate yourself about hardening WordPress security
- install a security plugin such as Wordfence
- enable **SSL/TLS** on your site if you have not already done so

Getting started

As a new user, you should first read the glossary section of the documentation to familiarize yourself with some basic concepts. The troubleshooting section for the main plugin is also found in the documentation. The support forum for the main plugin is at WordPress.org, but please first read this notice before posting.

First, understand the tradeoff between setting up a full node or using the cloud wallets.

Full node

A **full node** is harder to setup and maintain, but gives you performance and freedom to control network fee settings. With a full node you do not rely on a third party to do transaction verification. If you are interested in connecting to your own custom coin you might be interested in setting up a full node.

If you are interested in installing a **full node**, then follow the instructions in the YouTube video or article.

Cloud wallets

Cloud wallets on the other hand are easier to use and provide more coins, but are somewhat slower, and you rely on a third party service.

- If you are interested in installing the CoinPayments adapter:
- TL;DR installation instructions are on the coin adapter page.
- More detailed step-by-step instructions are on this page and in this YouTube video.
- If you prefer to install the block.io coin adapter then the installation instructions are on the block.io coin adapter page.

There is a Troubleshooting section on the coin adapter pages. It lists common problems and their solutions.

The support forums for the two web wallet coin adapters are here and here.

Feature overview

Crypto deposits

Your users can now deposit and withdraw Bitcoins.

User transactions

Users can transfer coins to other users.

Site fees

You set the fees that your users pay to you when they transact or withdraw coins.

Altcoins integration

Easily interface to other cryptocurrencies besides Bitcoin. Simply install the provided altcoin adapters for the currency you require.

Front-end UI

Present a simple, knockout.js-enabled frontend UI using shortcodes. Or roll your own.

E-mail notifications and confirmations

Users are notified by email whenever they perform transactions.

The admin can configure message stringss via the admin interface, or bind to WordPress actions and code your own notification functions in PHP.

Capability-based access control

Enable or disable user actions via WordPress capabilities.

DB storage

All accounting data is stored safely and efficiently in your database.

CSV Backups

Export accounting data to .CSV backup files with one click. Import again with ease.

Pluggable

PHP and JSON APIs give access to the wallets from your theme or other plugins. Look out for more apps that build on this awesome APIs in the near future, or create your own!

Documentation

PDF Documentation explains the ins and outs of this plugin in detail. Full PHPdoc is included for the PHP API.

Security

Protection against common plugin security vulnerabilities such as CSRF attacks and SQL injections.

You can extend the FREE Bitcoin and Altcoin Wallets plugin with:

- **coin adapter extensions** that connect to local nodes and other cloud wallets, and
- **app extensions**, such as payment gateways or user rewards.

Because these two concerns are decoupled, you can combine any of the existing app extensions with any of the coin adapters provided.

By mixing and matching plugins, you are in control of which currencies and services you provide.

Frequently Asked Questions

How can I integrate the plugin with my site?

Just insert the shortcodes anywhere to create forms to let a logged in user.

These are the most important shortcodes; consult the documentation for a complete list.

- **deposit funds:** `[wallets_deposit]`
- **withdraw funds:** `[wallets_withdraw]`
- **transfer funds to other users:** `[wallets_move]`
- **view their balance:** `[wallets_balance]`
- **view past transactions:** `[wallets_transactions]`

These shortcodes render knockout.js-enabled forms. Read the shortcodes documentation for more details.

You can enter the same UI elements into your theme's widget area. Simply go to *Appearance → Widgets* and use the provided front-end widgets.

You can also use a special menu item to display the user balances as part of a nav menu. See the *Frontend* section of the documentation for details.

Which coins are currently available?

Using the built-in coin adapter you can connect to a Bitcoin core full node.

You can connect to any Bitcoin-like full node that you manage yourself using the *Bitcoin and Altcoin Wallets: Full Node Multi Coin Adapter*. This would include coins such as Litecoin, Dogecoin, etc.

You can connect to any CryptoNote-based coin using the *Bitcoin and Altcoin Wallets: Monero Coin Adapter*. This would include coins such as Monero, Bytecoin, DigitalNote, Aeon, Haven, etc.

Also, if you are OK with using a web wallet service, then you can install the *CoinPayments adapter*. You then automatically get all of the coins that platform supports.

Dashed-Slug does not offer a full node adapter for ERC-20 tokens, but you can check this third-party solution.

How secure is it?

The Bitcoin and Altcoin Wallets plugin is only as secure as your WordPress installation. Regardless of whether you choose to install this plugin, you should have already taken steps to secure your WordPress installation. At a minimum you should do the following:

- Install a security plugin such as Wordfence.

- Read the Codex resources on Hardening WordPress.
- If you are connecting to an RPC API on a different machine than that of your WordPress server over an untrusted network, make sure to tunnel your connection via `ssh` or `stunnel`. See here.

I want to run a full node but bitcoind is too resource-hungry for my server.

Running a full node requires you to set up the daemon on a VPS or other machine that you own and administer. Normally the full blockchain needs to be downloaded, so you need to make sure that your server can handle the disk and network requirements. Here's some advice on how to run a bitcoind wallet in a low memory environment. If you are concerned about your available disk space, you may run a pruned node. The same instructions will apply to many wallets that are Bitcoin forks.

If running a full node is not important for you, you can choose to install the CoinPayments Adapter extension

I am setting up a Bitcoin full node. In my .conf file I have provided the rpcuser and rpcpassword parameters. Instead, the plugin recommends the rpcauth parameter.

You should *either* use `rpcuser` and `rpcpassword` to specify login credentials to the RPC API, *or* `rpcauth`, but not both. The `rpcauth` parameter is simply a way to specify a hashed/salted version of the username and password, rather than the plaintext values. The plugin recommends a hash that contains the username and password you have provided in the coin adapter settings. It uses the algorithm from `rpcauth.py`.

Can you install/configure the plugin for me? / I am having trouble with the bitcoin.conf file

I am available to answer any specific questions, if you attempt to install the plugin and you face some problem. Unfortunately I do not undertake installation and configuration of the plugin.

Once you install this software, it then needs to be maintained. If you find that you are having trouble installing the plugin or connecting it to a wallet, even with help, this is a good indication that you will find it hard to maintain it. In that case perhaps you should not be running a wallet with people's money.

Remember that you have two options: stand-alone wallets or third-party wallets. Running a web wallet is considerably easier than a stand-alone wallet, as it does not require system administration skills. As a general rule, if you have trouble using Linux from the command line, you will be better off installing a third-party wallet.

I don't like the built-in forms. Can I change them or provide my own?

1. First of all, the forms can be styled with CSS. They have convenient HTML classes that you can use. Please see the documentation under *Frontend → Style the frontend UIs with CSS*.
2. If you wish to translate the form texts to a different language, see the documentation under *Localization*.
3. If you wish to change the texts to something else, you may use the `wallets_ui_text_*` WordPress filters. See the documentation uder *Frontend → Template Modifications → Modifying text*.
4. If you wish to create forms with completely different markup, you can modify the provided template files in your theme or child theme. Please see the documentation under *Frontend → Template Modifications → Modifying Markup*. Theme developers can use this to provide customized templates for this plugin.

I want to do transactions from JavaScript. I don't want to use the provided shortcodes and their associated forms.

The provided built-in forms talk to a JSON API that is available to logged in users. If you choose to build your own front-end UI, you can do your AJAX calls directly to the JSON API.

Refer to the JSON API documentation for details.

I want to do transactions from the PHP code of my theme or plugin.

You can use the PHP API directly.

Refer to the PHP API documentation for details. Additionally, you may consult the following guide: HOWTO: Create/update transactions from PHP.

I want to replace an adapter with another one.

You can only have one coin adapter enabled per each coin. The plugin will warn you about this. To replace the adapter for a coin with a new adapter:

1. Deactivate the old adapter.
2. Install and configure the new adapter.
3. Enable the new adapter.
4. Got to *Wallets → Adapters*. Under the new adapter, select *Renew deposit addresses*. This will renew any user deposit addresses, as well as the cold storage deposit address for that coin.

Can you add XYZ coin for me?

Unfortunately no. I cannot cater to requests to add new coin adapters. I can only provide assistance to coin adapter developers by answering specific questions.

If your coin belongs to one of the types listed in this article, then you will most likely be able to use your coin with the plugin.

If you are interested in creating your own token, perhaps the best way is to create it as a TurtleCoin fork, then use the available TurtleCoin adapter extension.

Can I use an exchange to facilitate cryptocurrency trading on top of this plugin?

Yes, an Exchange extension is now available to premium dashed-slug members. The relevant blog post announcement is here.

Alternatively you can use the ShapeShift app extension that lets you display a front-end UI to the ShapeShift.io service.

I want to do a token sale. Can I use WooCommerce to do so?

It is not recommended that you use WooCommerce and the WooCommerce payment gateway extension to sell tokens or other coins.

Instead, consider using the Exchange extension to provide a way for your users to buy or sell your token. This requires that you have a working *coin adapter* for your coin or token.

In the adapters list template, why is the Hot Wallet Balance not equal to the Sum of User Balances?

The Hot Wallet Balance and the Users Balances are not the same thing. The difference is explained in the *Glossary* section of the documentation.

Reasons why the two amounts would diverge:

1. As users pay fees for transactions, their total balances decrease but the wallet balance stays the same.
2. If you have used the cold storage feature, or alternatively if you have deposited/withdrawn directly from the wallet and not via the plugin, then the balances will not match. See the *Cold Storage* section in the documentation for more.

Does the plugin support multisite network installations?

The plugin supports network installations in two modes, depending on whether the plugin is network-activated or not:

If you activate the plugin individually per each site, users will maintain separate balances on each site on your network. You will find the Wallets menu under the network admin menu. The `manage_wallets` capability will be assigned to the network super-admin. In case you need to trigger the cron jobs manually, use the distinct URL given in each case.

If you network activate the plugin, users will see the same balances on each site on your network. The `manage_wallets` capability must be assigned to each site's admin. If you need to trigger the cron jobs manually, use the URL given in the network admin pages. It will trigger cron jobs for all sites.

If you network-activate the parent plugin, make sure to also network-activate all the app extensions and coin adapters. If you activate the plugin separately on each subsite of your network, do the same with all the app extensions and coin adapters.

My site was hacked and some funds were stolen. Can you help?

I regret that I cannot help you. As you may know, cryptocurrency transactions are non-reversible. This is why the security disclaimer is there - you, as an administrator, are solely responsible for the security of your site. There is no way I could assume responsibility for lost funds. I do not hold private keys to any of your or your users' funds.

If you have evidence that your funds were stolen you should go to the police. In some situations it may be possible to track down the thieves.

I want to pay for premium membership but cannot or do not want to pay via PayPal.

You may contact me directly at info@dashed-slug.net if you wish to pay by Bitcoin, Litecoin or Dogecoin.

I want to do changes to the plugin's code. Where can I find X function/variable/command/etc?

The plugin and its extensions are yours to edit and you are free to hack them as much as you like. However, you are generally discouraged from doing so:

Firstly, I cannot provide support to modified versions of the plugin. Editing code can have unintended consequences.

But more importantly, if you do any modifications to the code, any subsequent update will overwrite your changes. Therefore, it is not recommended to simply fire away your favorite editor and hack away themes or plugins.

Here's what you should do instead: If possible, use an existing hook (action or filter) to modify the behavior of the plugin. Then, add your code to a child theme or in separate plugin file (any PHP file with the right headers is a valid plugin file).

If you can't find a hook that allows you to do the modifications you need, contact me to discuss about your need and I might be able to add a hook to the next patch of the plugin.

Additionally, if the change you intend to do is helpful to other users, I might be able to add it to the plugin code.

I am encountering some problem with the Bitcoin and Altcoin Wallets plugin

First check the *Troubleshooting* section of the documentation: Go to the dashed-slug downloads area and grab the *bundle* package of the plugin. This includes the documentation in PDF form. Scroll to the troubleshooting section and see if your problem is listed. If not, read the *Contact* section for ways in which you can ask for support. Some additional info about how to ask for support is found here.

I am encountering some problem with an Extension to the Bitcoin and Altcoin Wallets plugin

You should first have a look at the extension's page on dashed-slug.net. If your solution is not found there, you can also scan the appropriate subsection of the support forums. You can also post your own question. Please try to use the appropriate forum, and to post a new thread for each distinct issue.

How can I get support or submit feedback?

Please use the support forum on WordPress.org for all issues and inquiries regarding the plugin.

To get support on the provided extensions, subscribe to dashed-slug and go to the support forums.

For all other communication, please contact info@dashed-slug.net.

Are you available for custom development work?

Unfortunately I do not undertake custom projects. If you have an idea about a cool extension then please let me know about it. If it is a good fit for the project, it will be added to the backlog.

When implemented, It will be available either to all users for free, or for dashed-slug premium members.

Available extensions

Bitcoin and Altcoin Wallets *Turn your blog into a bank: Let your users deposit, withdraw, and transfer bitcoins and altcoins on your site.* Support forum

- This plugin is always available at the dashed-slug download area and at wordpress.org.
- The extensions can be downloaded at the dashed-slug download area.

App extensions

Bitcoin and Altcoin Wallets: Author Payroll extension *Pay your article authors with cryptocurrencies based on Google Analytics metrics.* Support forum

Bitcoin and Altcoin Wallets: Events Manager Cryptocurrency Payment Gateway *Let logged in users pay for tickets in Events Manager Pro from their cryptocurrency wallet.* Support forum

Bitcoin and Altcoin Wallets: Faucet Extension *Reward your users for solving CAPTCHAs.* Support forum

Bitcoin and Altcoin Wallets: ShapeShift Exchange extension *Lets your users exchange cryptocurrencies using ShapeShift.io from within your website.* Support forum

Bitcoin and Altcoin Wallets: Tip the Author *Allows users with cryptocurrency wallets to tip content authors.* Support forum

Bitcoin and Altcoin Wallets: WooCommerce Cryptocurrency Payment Gateway *Let logged in users pay at WooCommerce checkout from their cryptocurrency wallet.* Support forum

Cloud wallet coin adapters

Bitcoin and Altcoin Wallets: BlockIO Cloud Wallet Adapter *Allows your Bitcoin and Altcoin Wallets WordPress plugin to interface with your block.io cloud wallet account.* Support forum

Bitcoin and Altcoin Wallets: CoinPayments Adapter *Allows your Bitcoin and Altcoin Wallets WordPress plugin to interface with your CoinPayments.net cloud wallet account.* Support forum

Full node wallet adapters

The old coin adapters for full node wallets have been merged into the MultiAdapter

Bitcoin and Altcoin Wallets: Full Node Multi Coin Adapter *Allows the Bitcoin and Altcoin Wallets WordPress plugin to connect to a large number of Bitcoin-like full node wallets via their JSON RPC AP I.* Support forum

Transactions

Users can perform three types of transactions:

- *Deposits*, by sending funds from a blockchain to a deposit address.
- *Withdrawals*, by requesting to send funds to an external address on the blockchain.
- *Internal transfers* (aka *moves*) to other users.

Also, feature plugin extensions can perform such operations on behalf of the users with the `do_move()` and `do_withdraw()` functions. See the PHP API section for more.

Admin panel

As of release `2.3.0`, administrators with the `manage_wallets` capability can view all transactions on the system, via the admin panel titled *Transactions*.

Confirmations

As of release `2.3.0`, administrators with the `manage_wallets` capability can choose what kind of confirmation, if any, move and withdraw transactions need before they can change from `unconfirmed` to `pending`. This is set via the admin panel titled *Confirms*.

There are two possible types of confirmation that may be required for a withdraw or move transaction.

- *admin confirmation* This is given via the admin transactions table. For unconfirmed and pending transactions, there are actions to confirm and unconfirm a transaction in the column titled *Accepted by admin*.
- *user confirmation* The user that originates the transaction request receives an email containing a nonce and an ID that uniquely identifies the transaction request. If the user clicks on the link, the transaction is considered to be confirmed by the user. The body of the email is configurable. Administrators can also modify the *Verified by user* column via the *Transactions* admin panel.

Lifecycle (states)

As of release `2.3.0`, transactions can have one of the following states:

- `unconfirmed` This is the initial state for all transactions.
- `pending` For moves and withdrawals, this is the state that a transaction gets when it has been confirmed by a user and/or admin. You can choose which types of confirmation is required in the admin panel, under “Confirms”.
- `done` Transactions that have been executed and affect a user’s balance are in this state. Also, deposits that have reached the required number of network confirmations. The required confirmation count is set at the coin adapter settings for each coin separately.

- `failed` Transactions that could not be executed are marked as failed. They do not affect user balances.

Withdrawals and wallet locks

A *Coin Adapter* can process withdrawals only when it is in an *unlocked* state.

Navigate to *Wallets → Adapters* in your admin interface. In the *coin adapters list* you can see which adapters are *locked* or *unlocked*. Next to that you can also see how many pending withdrawals are associated with this adapter.

Typically you can unlock an adapter by setting a secret passphrase or PIN in the adapter settings. The wallet can remain unlocked indefinitely, or it may lock again automatically after a specified number of minutes. To control this behavior, go to *Wallets → Cron job* and set the *Time to retain withdrawal secrets* option.

While an adapter is unlocked, it will process withdrawals in batches as specified in the other cron job options.

If you choose to keep the wallet unlocked for a few minutes only, this improves the security of your site and wallets. But it also means that pending withdrawals will be processed only when you enter the wallet passphrase.

Alternatively you can choose to keep the wallets always unlocked by setting the *Time to retain withdrawal secrets* to zero (0). In this case, any secret passphrase that you enter in your coin adapters will have to be saved to the database, so that the wallets can remain unlocked.

Frontend

UIs

You can display UI forms to the front-end that let the user do basic operations with their cryptocurrency accounts.

Display the UI forms using shortcodes or Widgets, if your theme allows widget areas.

All the operations accessible via shortcodes are only relevant to logged in users. The shortcodes will only display if:

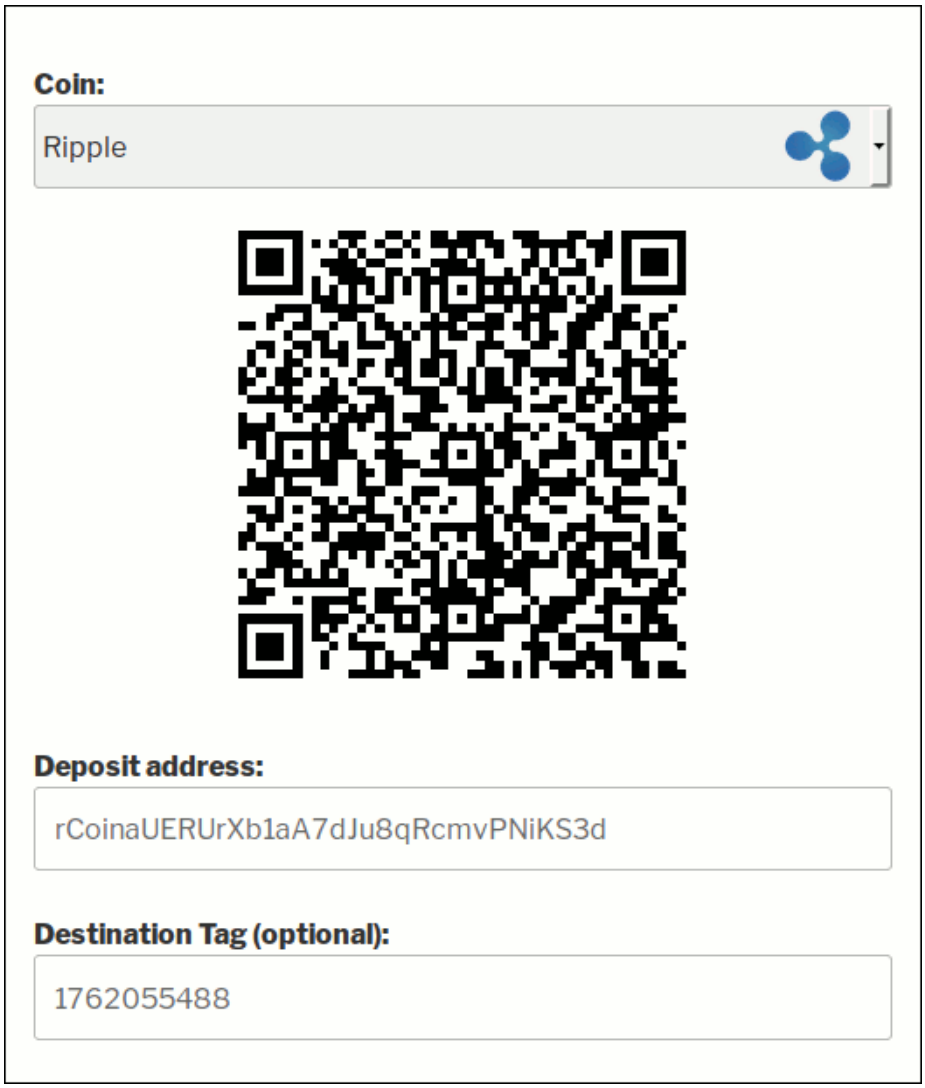
1. the user is logged in,
2. there is at least one wallet online, and
3. the user has the necessary capabilities, including `has_wallets`.

Some notes on the various types of templates available:

1. By default templates are *dynamic* templates. For example, `[wallets_balance]` will display the balance of the currently selected coin, will synchronize the selected coin with the coin selection in other dynamic templates, and will retrieve/update its value via the JSON API. They are rendered with no opacity when they have the HTML class `wallets-ready`.
2. Templates that include the `static` slug are rendered on the server side. The value is not retrieved via the JSON API and does not get updated while the page is running in the browser. These templates have the HTML class `static`.
3. Templates that include the `textonly` slug are rendered as a pure value, enclosed in a `<span>` tag. These have the `textonly` HTML class and can be embedded directly into text, without any UI.
4. Templates that include the `textonly_static` slug are rendered on the server side as a pure value, enclosed in a `<span>` tag. The value is not retrieved via the JSON API and does not get updated while the page is running in the browser. These have the `textonly` and `static` HTML classes. These templates can be embedded directly into text, without any UI.

Deposit funds

Required capabilities: `has_wallets`



Use the `[wallets_deposit]` shortcode to display a UI that will let the user know which address they can send coins to if they wish to make a deposit.

As of Bitcoin and Altcoin Wallets version `3.6.2`, this UI accepts an optional `qrsize` argument.

If set to an integer, determines the size (width and height) of the deposit QR code, in pixels.

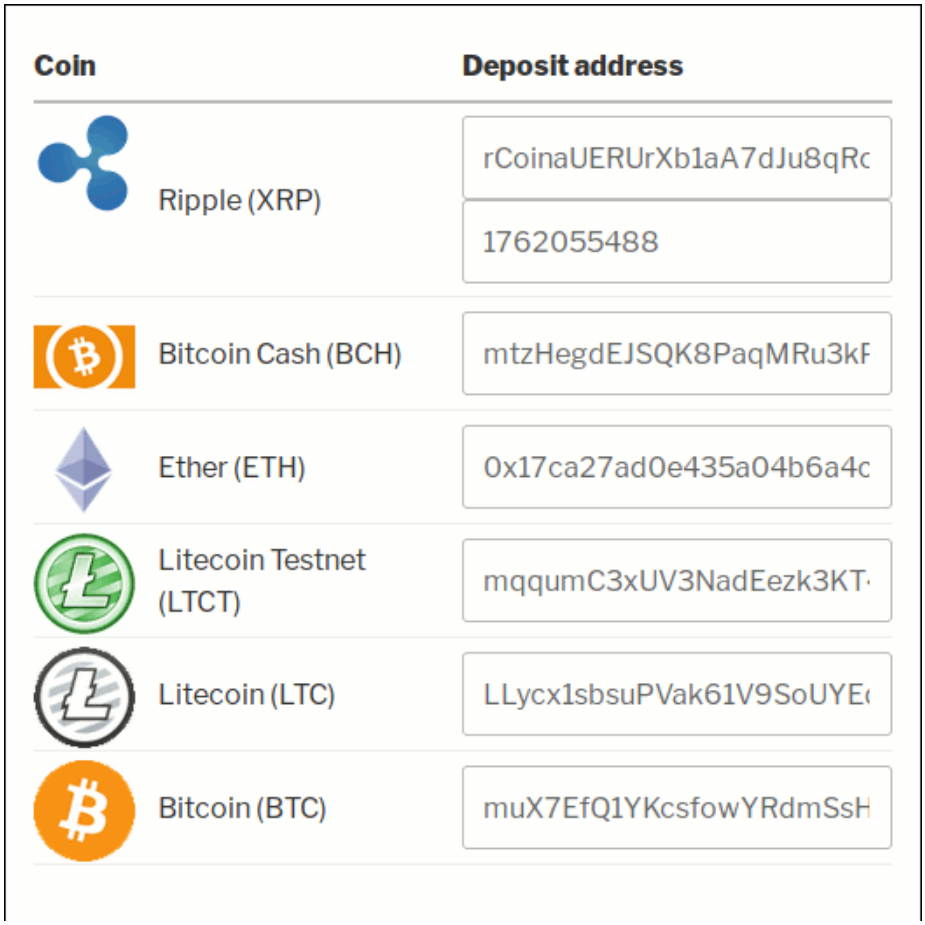
For example, to display a deposit UI with a QR code sized at 120x120 pixels:

```
[wallets_deposit qrsize="120"]
```

If the argument is left empty, no explicit size is set.

List template

You can also use `[wallets_deposit template="list"]` if you prefer to show a list of deposit addresses:



Static template

To render a deposit UI statically for Bitcoin, use `[wallets_deposit template="static" symbol="BTC"]`. The UI will not require the JSON API and will not refresh. The user will not be able to renew their deposit address.

Additionally, to display a deposit UI for the user with login name 'luser':

```
[wallets_deposit template="static" symbol="BTC" user="luser"]
```

Or to do the same for user with User ID 5:

```
[wallets_deposit template="static" symbol="BTC" user_id="5"]
```

Textonly template

You can display the deposit address of the currently selected coin as “textonly”. Use this if you need to embed the deposit address into your text, but also need the value to be updated via the JSON-API. These two shortcodes will display the deposit address and the “extra” field, if any, in `<span>` tags that can be embedded in any text.

```
[wallets_deposit template="textonly"]
[wallets_depositextra template="textonly"]
```

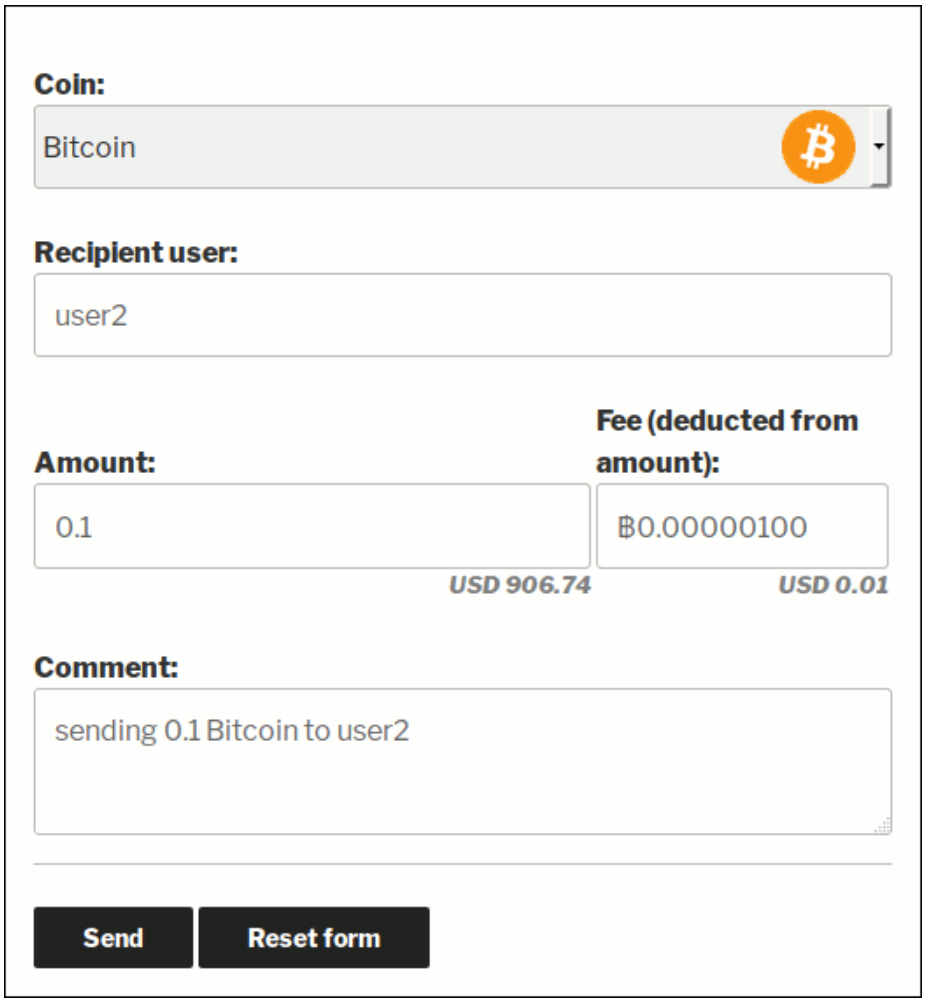
Static Textonly template

You can display the deposit address of the currently selected coin as “textonly”. Use this if you need to embed the deposit address into your text, and you want the text to be rendered once on the server-side. These two shortcodes will display the deposit address and the “extra” field, if any, in `<span>` tags that can be embedded in any text. The `user` attribute is optional, default is current user.

```
[wallets_deposit template="static_textonly" symbol="BTC" user="luser"]
[wallets_depositextra template="static_textonly" symbol="BTC"]
```

Withdraw funds

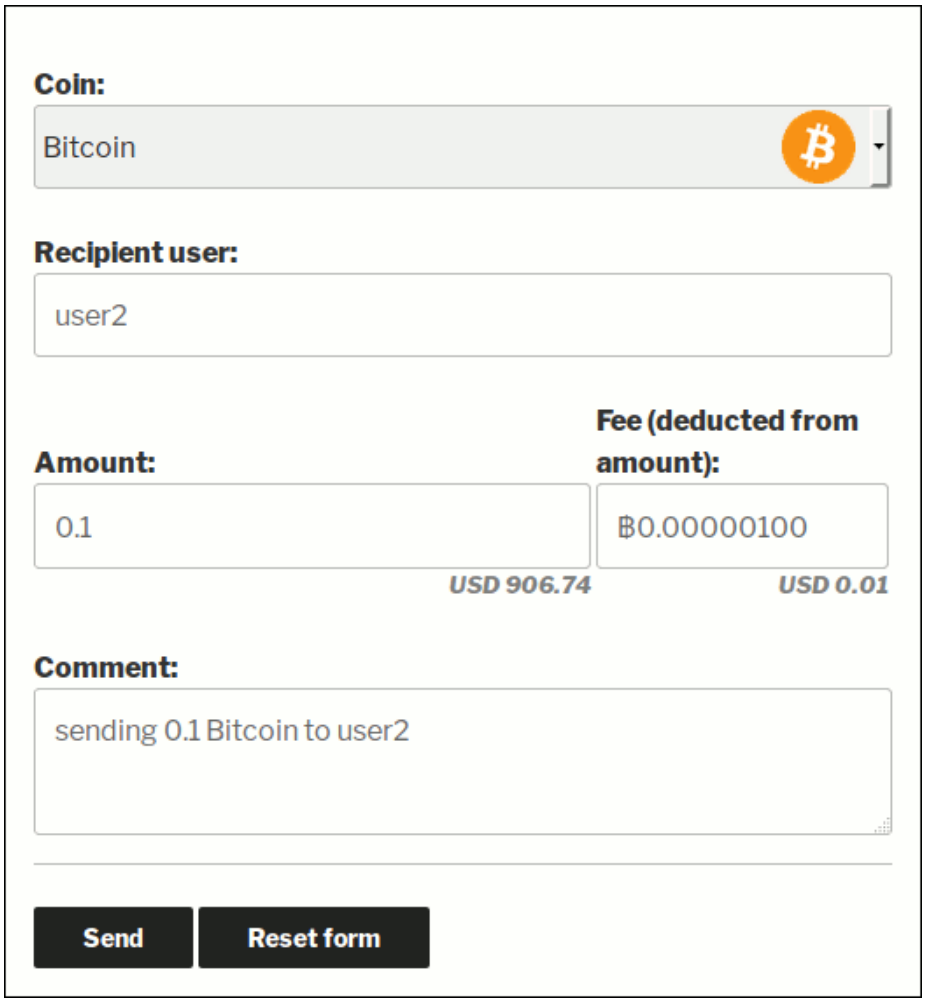
Required capabilities: `has_wallets`, `withdraw_funds_from_wallets`



Use the `[wallets_withdraw]` shortcode to display a form that will let the user withdraw funds. The user will be notified by email when the withdrawal succeeds.

Transfer funds to other users

Required capabilities: `has_wallets`, `send_funds_to_user`



Use the `[wallets_move]` shortcode to display a form that lets the user transfer coins to other users on your site. Both users will be notified by email when the transaction succeeds.

Display user balance(s)

Required capabilities: `has_wallets`

Use the `[wallets_balance]` shortcode to show the current user's balances one-by-one, with a dropdown to select current coin.

List template

Use `[wallets_balance template="list"]` to show all the current user's balances in a list.

Static template

These templates do not require the JSON API and will not refresh. They require the `symbol` attribute:

To render static HTML with the current user's BTC balance:

```
[wallets_balance template="static" symbol="BTC"]
```

To render static HTML with the BTC balance of user with ID 2.

```
[wallets_balance template="static" user_id="2" symbol="BTC"]
```

To render static HTML with the LTC balance of user with `login_name` equal to `luser`:

```
[wallets_balance template="static" user="luser" symbol="LTC"]
```

Textonly template

You can display the available balance of the currently selected coin as "textonly". Use this if you need to embed the balance into your text, but also need the value to be updated via the JSON-API. This shortcode will display the balance in a `<span>` tag that can be embedded in any text.

```
[wallets_balance template="textonly"]
```

Static Textonly template

You can display the available balance of the currently selected coin as “textonly”. Use this if you need to embed the balance into your text, and you want the text to be rendered once on the server-side. This shortcode will display the balance in a `<span>` tag that can be embedded in any text.

```
[wallets_balance template="static_textonly" symbol="BTC"]
```

View past transactions

Required capabilities: `has_wallets`, `list_wallet_transactions`

Coin:

Bitcoin

Rows per page:

10

Page:

1

Type	Tags	Time	Amount (+fee)	Fee
withdraw		4/30/2018, 8:10:05 PM	฿-0.01000000	฿0.00100000
withdraw		4/30/2018, 7:53:22 PM	฿-0.00001000	฿0.00000001
deposit	airdrop ad-5ae329b9134d0	4/27/2018, 7:47:08 PM	฿1.00000000	฿0.00000000
transfer	send wallets-faucet payout	4/26/2018, 4:46:27 PM	฿-0.00000100	฿0.00000000
transfer	send wallets-faucet payout	4/26/2018, 4:44:01 PM	฿-0.00000100	฿0.00000000
transfer	send move	11/21/2017, 1:18:45 PM	฿-0.00200000	฿0.00000100
transfer	send move	11/21/2017, 1:16:38 PM	฿-0.00100000	฿0.00000100

Use the `[wallets_transactions]` shortcode to display an interactive table that shows past deposits, withdrawals and transfers that affect the user's account. The table is paginated and data is loaded dynamically.

As of Bitcoin and Altcoin Wallets version `3.6.0`, the `[wallets_transactions]` shortcode accepts an optional `columns` argument. The value is a comma separated list of columns. You can control the columns displayed in the table, as well as their ordering. The default, if a value is not given, is to include all columns:

```
`type,tags,time,amount,fee,from_user,to_user,txid,comment,`
`confirmations,status,retries,admin_confirm,user_confirm`
```

For example, to display a simplified table of transactions, one might use the following shortcode:

```
`[wallets_transactions columns="time,type,amount,from_user,to_user,txid,status"]`
```

Rows template

To display transactions as a list of rows, where each box is one row:

```
[wallets_transactions template="rows"]
```

Static template

This template does not require the JSON API and does not refresh. It requires the `symbol` argument.

To render a table of the latest 10 Bitcoin transactions statically, use the following shortcode:

```
[wallets_transactions symbol="BTC"]
```

The static shortcode accepts additional optional attributes:

Categories

A comma-separated list of any of the following: `deposit`, `withdraw`, `move`, `trade`.

To only display recent Bitcoin deposits and withdrawals:

```
[wallets_transactions symbol="BTC" categories="deposit,withdraw"]
```

Tags

A comma-separated list of tags to look for in transactions. Transactions with any one of the specified tags are listed.

To display the latest 20 Litecoin payouts from the Faucet extension (assuming admin is giving out the payouts):

```
[wallets_transactions
    user="admin"
    template="static"
    symbol="LTC"
    categories="move"
    tags="wallets-faucet payout"
    rowcount="20"]
```

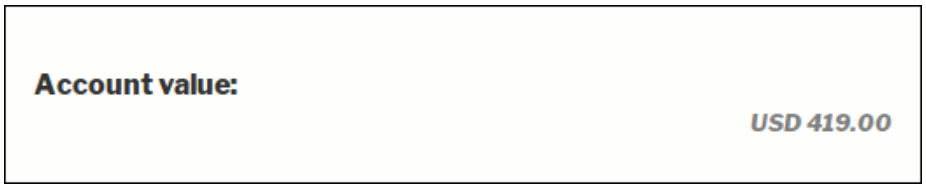
User and user_id

To display transactions for another user, use `user` to specify a `login_name` or `user_id` for a numeric User ID. To display all DOGE deposits of user with login name `shibe`:

```
[wallets_transactions symbol="DOGE" user="shibe" categories="deposit"]
```

Display total account value in the selected fiat currency

Required capabilities: `has_wallets`



Use the `[wallets_account_value]` shortcode to show the account's total value expressed in the default fiat currency.

This requires that you have first set up an API key for the fixer service. See the *Exchange Rates* section of this documentation for details.

Static template

To render the UI statically use `[wallets_account_value tempate="static"]`. The template will not require the JSON API and will not refresh.

Textonly template

You can display the total account value of the currently selected coin as "textonly", measured in the default fiat currency selected by the user or admin. Use this if you need to embed the total account value into your text, but also need the value to be updated via the JSON-API. This shortcode will display the balance in a `<span>` tag that can be embedded in any text.

```
[wallets_account_value template="textonly"]
```

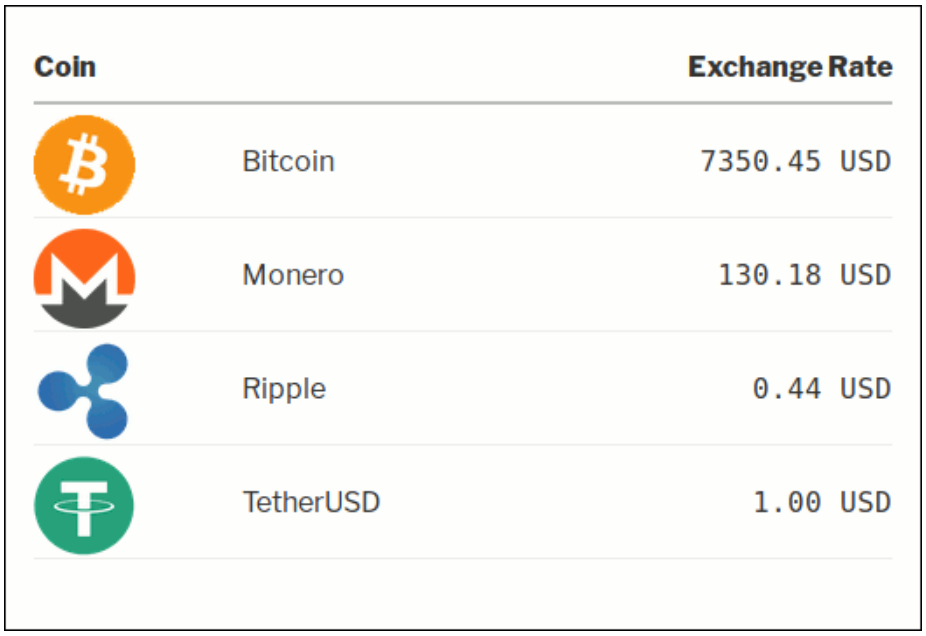
Static Textonly template

You can display the total account value of the currently selected coin as “textonly”, measured in the default fiat currency selected by the user or admin. Use this if you need to embed the total account value into your text, and you want the text to be rendered once on the server-side. This shortcode will display the account value in a `<span>` tag that can be embedded in any text.

```
[wallets_account_value template="static_textonly"]
```

Display exchange rates for each coin

Required capabilities: `has_wallets`



Use the `[wallets_rates]` shortcode to show the exchange rates between each coin and the default fiat currency.

Only currencies whose exchange rate is known will be listed.

The UI will not show, if the user set their default fiat currency to “none”. This can be set in the user profile admin screen.

To control the number of decimal digits to display exchange rates in, use the optional “decimals” attribute (introduced in version `4.0.4`). For example: `[wallets_rates decimals=2]`.

Static template

To render the same UI with static HTML, use `[wallets_rates template="static"]`. The UI will not require the JSON API and will not refresh.

Display total user balances per coin

Required capabilities: `has_wallets`

Use the `[wallets_total_balances]` shortcode to show the total user balances for each coin. The total is over all users on the site.

Unlike other UIs, this one is static by default. It is rendered as an HTML when the page loads, and it is not updated with the live polling mechanism (and does not use the JSON API).

API key

Required capabilities: `has_wallets`, `access_wallets_api`

The user can display their remote access API key, or click on a button to refresh their key to a new value, therefore invalidating the old API key. Use the `[wallets_api_key]` to display the UI.

Textonly template

You can display the API key as a "textonly" value. Use this tempate if you need to embed the API key into your text, but also need the value to be retrieved via the JSON-API. This shortcode will display the key in a `<span>` tag that can be embedded in any text. Note that the user can't refresh the key to a new value using the textonly template.

```
`[wallets_api_key template="textonly"]`
```

Displaying user balances as a menu

You can display the user balances for all enabled coins, as part of a WordPress menu.

1. Go to *Appearance* → *Menus*.
2. At the top right side of the screen, click *Screen Options*.
3. Under *Boxes*, make sure that *Bitcoin and Altcoin Wallets balances* is selected.
4. Now you are free to create a menu that includes the user balances.
5. Assign your menu to one of the menu areas of your theme.

Alternatively you can display this list of balances with a shortcode anywhere in your pages, using the Menu Shortcode plugin.

Frontend UI JavaScript event bindings

To bind JavaScript code to the plugin's frontend using a `wp_enqueue_script()` call, make sure to list `wallets_ko` as a dependency. This ensures that your script is loaded after the knockout scripts of the plugin. For example:

```
wp_enqueue_script(
    'mywalletsscript',
    'https://url/to/mywalletsscript.js',
    array( 'wallets_ko' ),
    '1.0.0'
);
```

Bind to data loaded events

You can use JavaScript to bind to the events of first loading coin info, nonces or both. The events are:

- `wallets_coins_ready` — Triggered on the `body` DOM element when the first `get_coins_info` call is successful. The list of available coins is available, as are exchange rates, user balances, etc. Coins are available in the `wp.wallets.viewModels.wallets.coins()` observable. Example:

  ```
  jQuery( 'body' ).on( 'wallets_coins_ready', function( event, coins ) {
      console.log( 'Loaded the following coins info:', coins );
  } );
  ```

- `wallets_nonces_ready` — Triggered on the `body` DOM element when the first `get_nonces` call is successful. The nonces required for performing transactions have been loaded and are available in the `wp.wallets.viewModels.wallets.nonces()` observable. Example:

  ```
  jQuery( 'body' ).on( 'wallets_nonces_ready', function( event, nonces ) {
      console.log( 'Loaded the following nonces:', nonces );
  } );
  ```

- `wallets_ready` — Triggered when both `get_nonces` and `get_coins_info` have returned successfully. Both the nonces and the coins info are available. For example, here's how you would make sure that Bitcoin is selected when the page first loads, if Bitcoin is available:

  ```
  jQuery( 'body' ).on( 'wallets_ready', function( event, coins, nonces ) {
      if ( 'object' === typeof coins.BTC ) {
          wp.wallets.viewModels.wallets.selectedCoin('BTC');
      }
  } );
  ```

Bind to transaction events

Withdraw and move forms use JavaScript to emit bubbling DOM events to denote that a withdrawal or a transfer to another user has succeeded.

The default behavior is to display informative message boxes using the browser's `alert()` function.

You can bind handlers to these DOM events to override, or to add to, this behavior.

The event handlers have access to the response of the JSON API, and to the form values entered by the user. Here's some example JavaScript code:

- `wallets_do_move` — Triggered after an internal transaction request is submitted to the server. Example code:

```
jQuery( 'body' ).on( 'wallets_do_move', function( event, response, symbol, amount, toac
    if ( 'string' === typeof response.result ) {
        if ( response.result == 'success' ) {
            alert( 'Successfully sent ' + amount + ' ' + symbol );
        } else {
            alert( "Move failed: \n" + response.message );
        }
    } else {
        alert( "Move failed!" );
    }
    event.preventDefault(); // Delete this line and the default action (alert box) will
});
```

- `wallets_do_withdraw` — Triggered after a withdrawal transaction request is submitted to the server. Example code:

```
jQuery( 'body' ).on( 'wallets_do_withdraw', function( event, response, symbol, amount,
    if ( 'string' === typeof response.result ) {
        if ( response.result == 'success' ) {
            alert( 'Successfully withdrew ' + amount + ' ' + symbol + ' to ' + address
        } else {
            alert( "Move failed: \n" + response.message );
        }
    } else {
        alert( "Move failed!" );
    }
    event.preventDefault(); // Delete this line and the default action (alert box) will
});
```

The above examples override the default handlers. If you wish to add to the default behavior, simply delete the `event.preventDefault()` statements and let the events bubble up.

Template modifications

Modifying text

All the UI elements are implemented as templates. (See the directory `wp-content/plugins/wallets/templates`.)

If you wish to provide translations to these templates, see the *Localization* section of this document.

If you wish to modify the text of the labels in some other way, you may use WordPress filters to provide the strings.

For example here is how you would bind to the filter for the *Coin* label:

```
function my_wallets_ui_text_coin( $text ) {
    return "<b>$text</b>";
}
add_filter( 'wallets_ui_text_coin', 'my_wallets_ui_text_coin' );
```

Any values you produce in these filters will not be passed to `esc_html()` or `esc_attr()`. This means that you can use the filters to modify the HTML markup. It is your responsibility to call these functions if you need them.

Form labels

Labels for input fields in the UI forms.

- `wallets_ui_text_adminconfirm`
- `wallets_ui_text_amount`
- `wallets_ui_text_amountafterfee`
- `wallets_ui_text_amountplusfee`
- `wallets_ui_text_apikey`
- `wallets_ui_text_available_balance`
- `wallets_ui_text_balance`
- `wallets_ui_text_coin`
- `wallets_ui_text_comment`
- `wallets_ui_text_confirmations`
- `wallets_ui_text_depositaddress`
- `wallets_ui_text_exchangerate`
- `wallets_ui_text_fee`
- `wallets_ui_text_feedeductedfromamount`
- `wallets_ui_text_from`
- `wallets_ui_text_max`
- `wallets_ui_text_me`
- `wallets_ui_text_page`
- `wallets_ui_text_reload`
- `wallets_ui_text_recipientuser`
- `wallets_ui_text_resetform`
- `wallets_ui_text_retriesleft`
- `wallets_ui_text_rowsperpage`
- `wallets_ui_text_send`
- `wallets_ui_text_show_zero_balances`
- `wallets_ui_text_status`
- `wallets_ui_text_tags`
- `wallets_ui_text_time`
- `wallets_ui_text_to`

- wallets_ui_text_totaluserbalances
- wallets_ui_text_txid
- wallets_ui_text_type
- wallets_ui_text_reload
- wallets_ui_text_renew
- wallets_ui_text_userconfirm
- wallets_ui_text_withdrawtoaddress
- wallets_ui_text_enterusernameoremail

Transaction categories

Categories of transactions, shown in the table of the [wallets_transactions] shortcode.

- wallets_ui_text_deposit Deposit
- wallets_ui_text_move Transfer
- wallets_ui_text_withdraw Withdrawal

Transaction statuses

Statuses of transactions, shown in the table of the [wallets_transactions] shortcode.

- wallets_ui_text_unconfirmed Unconfirmed
- wallets_ui_text_pending Pending
- wallets_ui_text_done Done
- wallets_ui_text_failed Failed
- wallets_ui_text_cancelled Cancelled

Popup text

- wallets_ui_text_submit_tx Popup text after submitting internal transfer.
- wallets_ui_text_submit_wd Popup text after submitting withdrawal.
- wallets_ui_text_op_failed Popup text after failing to submit transaction.
- wallets_ui_text_op_failed_msg Popup text after failing to submit transaction, includes error message.
- wallets_ui_text_contact_fail Popup text after failing to contact server.
- wallets_ui_text_get_new_address Popup text prompt before requesting a new deposit address.
- wallets_ui_text_apikey_renew_confirm Popup text prompt before requesting a new JSON-API key.
- wallets_ui_text_qrcode_scan_failed Popup text after failing to scan a QR code.

Form validation error messages

- wallets_ui_text_invalid_add Error message indicating that a withdrawal address does not pass checksum validation.

- `wallets_ui_text_amount_positive`Error message indicating that an amount is not positive.
- `wallets_ui_text_insufficient_balance` Error message indicating that an amount specified is larger than the user balance.
- `wallets_ui_text_minimum_withdraw` Error message indicating that a withdrawal amount is smaller than the minimal withdrawal amount as specified in the coin adapter settings.
- `wallets_ui_text_amount_less_than_fee` Error message indicating that a requested transaction amount is less than the fees it would incur, as specified in the coin adapter settings.

Mouseover text

- `wallets_ui_text_copy_to_clipboard` Shown when hovering mouse over copy to clipboard button.
- `wallets_ui_text_reload` Shown when hovering over the *Reload data from server* button.

Shortcode rendering error messages

If a shortcode cannot be rendered, an error message is displayed. The HTML container gets the class `error`. An error can be caused if: - the user is not logged in - the user does not have the necessary capabilities - the template throws an exception

Use the following filter to modify the error message:

- `wallets_ui_text_shortcode_error` Shown when a shortcode cannot be rendered.

The following strings are substituted in the error message pattern:

- `%1$s` Template *generic* part (e.g. `deposit`, `move`, `withdraw`, etc.)
- `%2$s` Template *specialized* part (e.g. `default`, `static`, etc.)
- `%3$s` View file (e.g. `/wp-content/plugins/wallets/templates/deposit/default.php`)
- `%4$s` The text of the error message.

This is how to override the error message. Use this code in your child theme and edit the text as needed.

```
function wallets_ui_text_shortcode_error_filter( $pattern ) {
    return 'Error while rendering the <code>%1$s</code> shortcode ' .
        'with its <code>%2$s</code> template in <code>%3$s</code>: %4$s';
}
add_filter( 'wallets_ui_text_shortcode_error', 'wallets_ui_text_shortcode_error_filter' );
```

Modifying markup

If you need to modify the HTML markup for the UI elements (i.e. not just the text):

Starting from version 5.0.0, the templates for the UIs are in the `wp-content/plugins/wallets/templates` directory.

shortcode	template file
[wallets_api_key]	api_key.php
[wallets_account_value]	account_value.php
[wallets_account_value template="static.php"]	static.php
[wallets_account_value template="text.tpl.php"]	text.tpl.php
[wallets_account_value template="static_text.tpl.php"]	static_text.tpl.php
[wallets_api_key]	api_key.php
[wallets_api_key template="text.tpl.php"]	text.tpl.php
[wallets_balance]	balance.php
[wallets_balance template="list.tpl.php"]	list.tpl.php
[wallets_balance]	balance.php
[wallets_balance template="static.php"]	static.php
[wallets_balance template="text.tpl.php"]	text.tpl.php
[wallets_balance template="static_text.tpl.php"]	static_text.tpl.php
[wallets_deposit]	deposit.php
[wallets_deposit template="list.tpl.php"]	list.tpl.php
[wallets_deposit]	deposit.php
[wallets_deposit template="static.php"]	static.php
[wallets_deposit]	deposit.php
[wallets_deposit template="text.tpl.php"]	text.tpl.php
[wallets_deposit]	deposit.php
[wallets_deposit template="static_text.tpl.php"]	static_text.tpl.php
[wallets_deposit_extra]	deposit_extra.php
[wallets_deposit_extra template="text.tpl.php"]	text.tpl.php
[wallets_deposit_extra]	deposit_extra.php
[wallets_deposit_extra template="static_text.tpl.php"]	static_text.tpl.php
[wallets_move]	move.php
[wallets_rates]	rates.php
[wallets_rates]	rates.php
[wallets_rates template="static.php"]	static.php
[wallets_total_balances]	total_balances.php
[wallets_transactions]	transactions.php

shortcode	template file
[wallets_transactions]	templates/transactions.php
[wallets_transactions_static]	templates/transactions-static.php
[wallets_withdraw]	templates/withdraw.php

To override the code in any of these templates, do not modify the plugin's copy of the files. Any changes you make will be reverted when the plugin is updated. Instead, copy any of these files in `wp-content/themes/YOURTHEME/templates/wallets`. When you modify the markup, make sure to preserve the bindings to knockout observables (the `data-bind` HTML attributes). Child themes take precedence (i.e. overwrite) parent plugin templates. When no theme templates are found for wallets, the files that accompany the plugin are used.

Dynamic templates

Most templates rely on knockout.js observables. They can be refreshed when the user clicks on the refresh button. They can also refresh via polling the JSON API, if this is enabled under *Wallets → Frontend settings*. Finally, the content of these dynamic templates refreshes when the user switches to this page from another tab or window.

Static templates

The templates marked "static" are rendered on the server side. They do not make use of knockout.js observables, but rather only use some light jQuery code. The content of these refreshes only if the page is refreshed.

Output additional markup around the UIs

All UIs when displayed activate the following two WordPress actions:

- `wallets_ui_before`
- `wallets_ui_after`

Additionally the following actions exist to differentiate between UIs:

- `wallets_ui_before_move`
- `wallets_ui_after_move`
- `wallets_ui_before_withdraw`
- `wallets_ui_after_withdraw`
- `wallets_ui_before_balance`
- `wallets_ui_after_balance`
- `wallets_ui_before_deposit`

- `wallets_ui_after_deposit`
- `wallets_ui_before_transactions`
- `wallets_ui_after_transactions`
- `wallets_ui_before_account_value`
- `wallets_ui_after_account_value`
- `wallets_ui_before_total_balances`
- `wallets_ui_after_total_balances`
- `wallets_ui_before_rates`
- `wallets_ui_after_rates`
- `wallets_ui_before_wallets_api_key`
- `wallets_ui_after_wallets_api_key`

Style the frontend UIs with CSS

All the UI elements have the `dashed-slug-wallets` class to help you with CSS styling. Here's some selectors you can use:

- `.dashed-slug-wallets.api-key { ... }`
- `.dashed-slug-wallets.account-value { ... }`
- `.dashed-slug-wallets.balance { ... }`
- `.dashed-slug-wallets.deposit { ... }`
- `.dashed-slug-wallets.move { ... }`
- `.dashed-slug-wallets.rates { ... }`
- `.dashed-slug-wallets.total-balances { ... }`
- `.dashed-slug-wallets.transactions { ... }`
- `.dashed-slug-wallets.withdraw { ... }`

Additionally, starting with version `4.1.0`, you can now use the Customizer to control UI styling. Look for the *Bitcoin and Altcoin Wallets* section in your customizer menu.

Capabilities

WordPress does access control using the concept of Roles and Capabilities. You can read about it in the WordPress Codex.

Bitcoin and Altcoin Wallets capabilities

You can assign capabilities to roles using the admin interface. Just navigate to *Wallets* → *Capabilities* and check capabilities in the matrix, then hit *Save changes*.

Capabilities control which shortcodes and JSON API endpoints are available to a user.

By default, the PHP API overrides capabilities checks. You can enforce checking with the new argument, `$override_capabilities`.

As of 2.2.4 you cannot modify the capabilities of the administrator role using this interface. This is to prevent admin accounts from locking themselves out of the `manage_wallets` capability.

manage_wallets

- Enables admin interface. For administrators only.

has_wallets

- Needed for all shortcodes.
- Needed for the `get_transactions`, `do_withdraw` and `do_move` JSON APIs.
- Needed for the `do_move()`, `do_wifhdraw()`, `get_coin_adapters()`, `get_balance()`, `get_new_address()`, `get_account_id_for_address()` and `get_transactions()` PHP APIs. Only checked when `$override_capabilities` is `true`.

list_wallet_transactions

- Need for the `wallets_transactions` shortcode.
- Needed for the `get_transactions` JSON API.
- Needed for the `get_transactions()` PHP API. Only checked when `$override_capabilities` is `true`.

send_funds_to_user

- Need for the `wallets_move` shortcode.
- Needed for the `get_users_info` JSON API.

- Needed for the `do_move()` PHP API. Only checked when `$override_capabilities` is `true`.

withdraw_funds_from_wallet

- Needed for the `wallets_withdraw` shortcode.
- Needed for the `do_withdraw` JSON API.
- Needed for the `do_withdraw()` PHP API. Only checked when `$override_capabilities` is `true`.

view_wallets_profile

- Needed to view the *Bitcoin and Altcoin Wallets* section in the WordPress user profile admin page. Users with the `manage_wallets` capability can always view that section for any user with `has_wallets`.

Capabilities auto-repair feature

Since `4.4.4`, the plugin checks on each cron to see if at least one user in the *Administrator* role has the `manage_wallets` capability. If none have it, then the capability is assigned to the *Administrator* role.

This circumvents any capability-related problems that could otherwise lock the administrator out of the *Wallets* menu.

Notifications

Users are notified by e-mail or BuddyPress private messages when they perform deposits or withdrawals and when they send or receive funds. In the admin interface navigate to *Wallets → Notification settings*.

- You can control whether you want BuddyPress private messages.
- You can control whether you want email messages.
- You can control which types of notifications are sent.
- You can edit the subject line templates.
- You can edit the text body templates.

You can use the following variable substitutions in your templates:

- `###ACCOUNT###` — Account username.
- `###ACCOUNT_ID###` — WordPress account ID.
- `###OTHER_ACCOUNT###` — Username of other account (for fund transfers between users)
- `###OTHER_ACCOUNT_ID###` — WordPress account ID of other account (for fund transfers between users)
- `###TXID###` — Transaction ID. (This is normally the same as the txid on the blockchain. Internal transactions are also assigned a unique ID.)
- `###SYMBOL###` — The coin symbol for this transaction (e.g. “BTC” for Bitcoin)
- `###AMOUNT###` — The amount transacted.
- `###FEE###` — For withdrawals and transfers, the fees paid to the site.
- `###AMOUNT_WITHOUT_FEE###` — The amount transacted, minus any fees paid.
- `###FIAT_SYMBOL###` — The fiat currency that the user has selected to see equivalent amounts in (falling back to the site-wide default fiat currency)
- `###FIAT_AMOUNT###` — Same as ###AMOUNT###, but expressed in the fiat currency that the user has selected to see equivalent amounts in (or in the site-wide default fiat currency)
- `###FIAT_FEE###` — Same as ###FEE###, but expressed in the fiat currency that the user has selected to see equivalent amounts in (or in the site-wide default fiat currency)
- `###FIAT_AMOUNT_WITHOUT_FEE###` — Same as ###AMOUNT_WITHOUT_FEE###, but expressed in the fiat currency that the user has selected to see equivalent amounts in (or in the site-wide default fiat currency)
- `###CREATED_TIME_LOCAL###` — The date and time of the transaction in the local timezone. Format: YYYY-MM-DD hh:mm:ss
- `###CREATED_TIME###` — The UTC date and time of the transaction. Format: YYYY-MM-DD hh:mm:ss
- `###COMMENT###` — For internal fund transfers, the comment attached to the transaction.
- `###ADDRESS###` — For deposits and withdrawals, the external address.
- `###EXTRA###` — For some coins, extra info on transactions. E.g. Monero *Payment ID*, Ripple *Destination Tag*, etc.

WordPress actions in PHP

You can use the following wordpress actions to get notified about transactions in your PHP code.

The first argument to the action is always a `stdClass` Object, describing the transaction status.

- `wallets_withdraw`
- `wallets_move_send`
- `wallets_move_receive`
- `wallets_deposit`
- `wallets_withdraw_failed`
- `wallets_move_send_failed`

Example:

```
public function my_action_move_send( $data ) {
    error_log( 'SENDING INTERNAL TRANSFER: ', print_r( $data, true ) );
}

add_action( 'wallets_move_send', 'my_action_move_send' );
```

BuddyPress private messages

To enable private messages: 1. Go to *Settings → BuddyPress* and enable the *Private Messages* component. This will enable private messages for BuddyPress. 2. Go to *Wallets → Notifications →* and enable *Send BuddyPress private messages*. This will make *Bitcoin and Altcoin Wallets* use BuddyPress to send notifications as private messages.

I heard you liked notifications, so I…

If you choose to enable BuddyPress notifications you might decide to disable email notifications. Keep in mind that BuddyPress can also send its own email notifications for private messages, so your notification will arrive twice to the user's inbox if everything is enabled.

Localization

Basic concepts

The plugin has been internationalized. All strings use the `__()` function and its friends.

To familiarize yourself with some basic localization concepts, you can read this page in the Codex.

Language domains

The localized strings have been split into two *language domains*. The domains are named `wallets` and `wallets-front`. This allows you to easily localize the frontend without localizing the backend (WordPress admin interface). The frontend requires considerably less effort to localize than the entire plugin.

The `.pot` files are found in `wp-content/plugins/wallets/languages/wallets.pot` and `wallets-front.pot`.

Translating the plugin to your language

1. Copy the `.pot` file to a `.po` file that includes the ISO 639-1 language code you want to translate. For example, to translate the frontend to Spanish, copy `wallets-front.pot` to `wallets-front-es.po`. For a list of language codes available, consult the gettext documentation.
2. Edit the `.po` file and translate the strings to your language. You can use any text editor to do this, but you may find it easier if you use POEdit.
3. Save the `.po` file and convert it to a machine-readable `.mo` file. If you use POEdit there is an option to compile to `.mo`. Place both files in the `wp-content/plugins/wallets/languages` directory.
4. That's it. To test your translation, go to your WordPress admin screens, and to *Settings* → *General* → *Site Language*. Select your language and click on *Save Changes*.

Donating translations

If you have produced a translation for a language that is not already available, and you wish to donate it to the project, you can email it to info@dashed-slug.net. It will be much appreciated!

Block explorers

Transactions and addresses are displayed in the admin interface or frontend as links to block explorers.

- The plugin knows about a few common coins and their block explorers, and provides sane defaults for these coins.
- Site owners can provide alternative or additional block explorer URIs by binding to WordPress filters.
- Coin adapter developers can provide alternative or additional block explorer URIs by overriding the appropriate functions from the abstract adapter class.
- In the coin URIs, the symbols `%s` are substituted with the address or transaction ID string.
- The chainz.cryptoid.info block explorer is used as fallback because it supports several coins.

Site administrators

Site administrators can provide URI patterns by binding to the following filters: `wallets_explorer_uri_add_XXX` and `wallets_explorer_uri_tx_XXX`, where `XXX` is a coin symbol. Example for DOGE:

```
function explorer_uri_address_doge( $uri ) {
    return "https://dogechain.info/address/%s";
}
add_filter( 'wallets_explorer_uri_add_DOGE', 'explorer_uri_address_doge' );

function explorer_uri_transaction_doge( $uri ) {
    return "https://dogechain.info/tx/%s";
}
add_filter( 'wallets_explorer_uri_tx_DOGE', 'explorer_uri_transaction_doge' );
```

Coin adapter developers

Coin adapter developers can provide URI patterns by overriding the base coin adapter functions.
An example from the DOGE coin adapter:

```
{
    public function explorer_uri_address( $uri ) {
        return "https://dogechain.info/address/%s";
    }

    public function explorer_uri_transaction( $uri ) {
        return "https://dogechain.info/tx/%s";
    }
}
```

Cold Storage

There is a cold storage section in the admin panel of the Bitcoin and Altcoin Wallets free WordPress plugin.

What is Cold Storage

When users hold cryptocurrencies on your site, these funds are stored on your back-end wallets. The plugin uses these wallets to communicate with the blockchains.

If a malicious hacker were to somehow gain access to your WordPress, they could steal your users' money. This is why you must:

- keep your site secure, and
- plan for the possibility that your site can be breached. No system is 100% safe.

One very popular security measure is to keep part of the funds in *cold storage*. This is an offline wallet, to which your site's code does not have access to.

The concept is not much unlike that of fractional reserve banking: You do not keep all of the deposited funds on the live wallet. Instead you keep a fraction of those funds online, and the rest you keep on another wallet. Preferably a wallet that is not even connected to the internet. Paper wallets and hardware wallets are good for this, but you could also use a computer that you keep offline at all times.

The *Cold Storage* admin page makes it easy to split funds between a hot (live) and a cold (offline) wallet. For every currency you have installed, the table will show you the following information:

- *Sum of User Balances*: The total of all your users' balances.
- *Hot Wallet Balance*: How many coins you have in your hot wallet. This column also shows you what percentage of the *Sum of User Balances* are currently backed by the hot wallet balance.
- *Sum of Pending Withdrawals*: How many funds are locked in `unaccepted` or `pending` withdrawals. Withdrawals can only proceed if there are enough funds in the hot wallet.

Using Cold Storage to secure your site

You can now target to keep a **fixed percentage of the total balances** online at all times. Send the rest of the coins to cold storage and keep them safe. At a later time you can deposit them again into your site as needed.

The admin page will let you **deposit** and **withdraw** funds to and from an external cold storage wallet. These transactions are not recorded in your transactions table. They do not affect any user balances. They only affect the online wallet balance. In the event of a security breach, all of the offline coins are safe and your risk is minimized.

You will want to **keep online more money than you expect your users to withdraw** all at once. If there are not enough funds online, user-requested withdrawals will fail.

On the other hand, you could decide that all withdrawals require **confirmation by an admin**. Then you can keep all of the funds in cold storage and only put them back online before you manually approve user withdrawals.

Any administrator with the manage_wallets capability has access to that page. Keep in mind that you need to wait for transactions to confirm before they affect your balance. If you are using the CoinPayments adapter, withdrawals might take a few minutes to execute.

Hardware wallets

Consider getting a hardware wallet to use for cold storage. A Trezor or Ledger Wallet will do nicely.

APIs, APIs everywhere!

Bitcoin and Altcoin Wallets plugin is built so that:

- It provides an **abstraction layer** to wallets. Bitcoin core and other wallets can be accessed uniformly.
- It exposes **programmatic access** to wallets via its *PHP API*.
- It exposes a *JSON API* over HTTPS for **remote access** to the wallets and plugin.

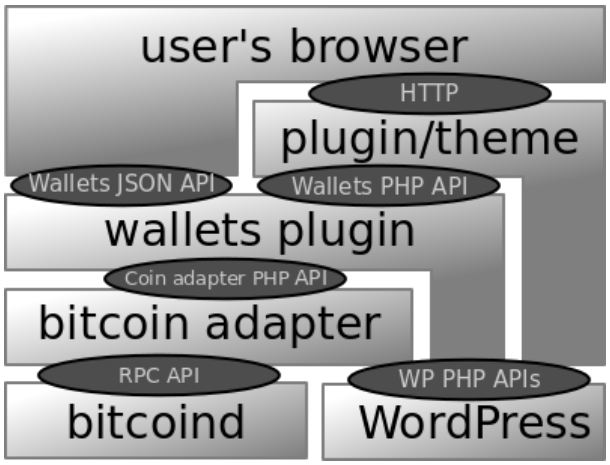


Figure 1: software stack

Wallets JSON API

The Bitcoin and Altcoin Wallets WordPress plugin exposes the following JSON APIs over HTTP:

1. The *transaction API* enables logged in users to perform and view transactions from the frontend.
2. The *notification API* lets the wallet daemons notify the system of incoming transactions. (For `bitcoind`, this maps to `-walletnotify` and `-blocknotify`).

Versioning

The current version of the JSON API is **version 3**.

By default, this is the only version enabled. To allow earlier versions to be available for backwards-compatibility with third party code, go to *Wallets → Frontend settings → JSON API Settings* and check *Enable legacy JSON APIs*.

Transaction API

You can control which parts of the API are available by setting capabilities in the admin screens.

Get coins info

Retrieves information relevant to all the coins enabled on the site.

Used by all shortcode front-ends to display a drop-down selection of available coins.

Pretty URI path: `/wallets/api3/get_coins_info` **Ugly URI path:** `?__wallets_action=get_coins_info&__walle`

Parameters: none

Required capabilities: `has_wallets`

Example response:


```
{
    "coins":{
        "LTC":{
            "name":"Litecoin",
            "symbol":"LTC",
            "is_fiat":true,
            "is_crypto":false,
            "icon_url":"http:\/\/example.com\/wp-content\/plugins\/wallets-litecoin\/assets\
            "sprintf":"\u0141%01.8f",
            "extra_desc":"Destination address label (optional)",
            "explorer_uri_address": "https://live.blockcypher.com/ltc/address/%s/",
            "explorer_uri_tx": "https://live.blockcypher.com/ltc/tx/%s/",
            "balance":0.00659988,
```

```
            "move_fee":0,
            "move_fee_proportional":0,
            "withdraw_fee":0.0002,
            "withdraw_fee_proportional":0,
            "deposit_address":"LPsj8nDiuBjbvRFR8CDoYhSbd4nDekbBQV",
        },
        "BTC":{
            "name":"Bitcoin",
            "symbol":"BTC",
            "is_fiat":true,
            "is_crypto":false,
            "icon_url":"http:\/\/example.com\/wp-content\/plugins\/wallets\/includes\/..\/as
            "sprintf":"\u0e3f%01.8f",
            "extra_desc":"Destination address label (optional)",
            "explorer_uri_address": "https://blockchain.info/address/%s",
            "explorer_uri_tx": "https://blockchain.info/tx/%s",
            "balance":0.00091565,
            "move_fee":0,
            "move_fee_proportional":0,
            "withdraw_fee":0.0002,
            "withdraw_fee_proportional":0,
            "deposit_address":"mrXEs8Kbj7mcMU1ZAq84Kdm85Vdd2Xg2b2",
        }
    },
    "result":"success"
}
```


Get transactions

Retrieve past transaction info (deposits, withdrawals and transfers to other users) of the currently logged in user.

Used by the `[wallets_transactions]` shortcode UI to display a paginated table of transactions.

URI path: `/wallets/api3/get_transactions/SYMBOL/COUNT/FROM`

Parameters:

- **SYMBOL**: The coin's symbol, e.g. BTC, LTC, etc. Only transactions regarding this coin will be retrieved.
- **COUNT**: Retrieve this many transactions
- **FROM**: Start retrieving transactions from this offset (for pagination)

Required capabilities: `has_wallets`, `list_wallet_transactions`

Example response:


```
{
```

```
"transactions":[
   {
      "category":"deposit",
      "tags": "",
      "account":"1",
      "other_account":null,
      "address":"1rXEs8Kbj7mcMU1ZAq84Kdm85Vdd2Xg2b2",
      "txid":"c9c30612ea6ec2509c4505463b6f965ac25e8e2cff6451e480aa3b307377df97",
      "symbol":"BTC",
      "amount":"0.0000100000",
      "fee":"0.0000000000",
      "comment":null,
      "created_time":"2016-12-17 15:22:14",
      "updated_time":"2016-12-30 11:33:14",
      "confirmations":"3249",
      "status":"done",
      "retries":0,
      "admin_confirm":0,
      "user_confirm":0,
      "extra":null,
      "other_account_name":null
   },
   {
      "category":"move",
      "tags": "send move",
      "account":"1",
      "other_account":"2",
      "address":"",
      "txid":"move-58629b173cb8f0.44669543-send",
      "symbol":"BTC",
      "amount":"-0.0000133400",
      "fee":"0.0000010000",
      "comment":"comment test",
      "created_time":"2016-12-27 16:47:19",
      "updated_time":"2016-12-27 16:47:19",
      "confirmations":"0",
      "status":"done",
      "retries":0,
      "admin_confirm":0,
      "user_confirm":1,
      "extra":null,
      "other_account_name":"luser"
   },
   {
      "category":"withdraw",
      "tags": "",
```

```
        "account":"1",
        "other_account":null,
        "address":"1i1B4pkLQ2VmLZwhuEGto3NAJdeh4xJr1W",
        "txid":"fed08f9a90c526f2bb791059a8718d422b8fdcb55f719bd36b6e3d9717e815e0",
        "symbol":"BTC",
        "amount":"-0.0000600000",
        "fee":"0.0000500000",
        "comment":"withdrawing bitcoins",
        "created_time":"2016-12-30 11:44:37",
        "updated_time":"2016-12-30 12:46:41",
        "confirmations":"12",
        "status":"done",
        "retries":2,
        "admin_confirm":1,
        "user_confirm":1,
        "extra":null,
        "other_account_name":null
      }
   ],
   "result":"success"
}
```


All times are GMT.

Get nonces

Performing `do_move`, `do_withdraw` and `do_new_address` actions requires a nonce code. Get these via this api. Each of the two nonces is returned only if the current user has the capability to perform the action.

This same endpoint is used to retrieve the API key for programmatic access to the *Transactions API*. It endpoint requires that you are logged in. It cannot be accessed using an API key.

Pretty URI path: `/wallets/api3/get_nonces` **Ugly URI path:** `?__wallets_action=get_nonces&__wallets_apive`

Parameters: none

Required capabilities: `has_wallets`

Example response:


```
{
    "nonces":{
        "do_withdraw":"893a9f9dad",
        "do_move":"6d5ebff18a",
        "do_new_address": "27cda821b5",
        "api_key":"5d3919d169f2049b6c2fbd5e2d886abcc1ebd5a12b23bd6dd4eebcbf084f55b1"
    },
    "result":"success"
```

```
}
```

Create new deposit address

Requests that the current user is assigned a new deposit address for the selected coin.

Any previous addresses are retained, so that deposits to any address will be assigned to the user.

Ugly URI path: `/?__wallets_action=do_new_address&__wallets_apiversion=3&__wallets_symbol=SYMBOL&`

Parameters:

- **SYMBOL**: The symbol of the coins to move, e.g. BTC, LTC, etc.
- **WPNONCE**: The `do_new_address` WordPress nonce from the `get_nonces` call.

Required capabilities: `has_wallets`

Example response:

```
{
   "result":"success",
   "new_address": "mpwokEmcuUmvp5TpM83TuPLQyfXHpvgGmF"
}
```

For some coins with extra info (e.g. Monero PaymentID, Ripple Destination Tag, SteemIt Memo, etc.) there will be an additional field `extra` carrying the value of the extra field.

Move funds to another user

Transfers funds to another user. The transfer fee that the administrator has set in the coin adapter is charged to the sender.

Ugly URI path: `/?__wallets_action=do_move&__wallets_apiversion=3&__wallets_symbol=SYMBOL&__walle`

Parameters:

- **SYMBOL**: The symbol of the coins to move, e.g. BTC, LTC, etc.
- **TOACCOUNT**: User ID of the recipient. The IDs are accessible via `get_user_info`.
- **AMOUNT**: The amount of coins to transfer, excluding any transaction fees. This is the amount that the recipient is to receive.
- **COMMENT**: A descriptive string that the sender attaches to the transaction.
- **WPNONCE**: The `do_move` WordPress nonce from the `get_nonces` call.

Required capabilities: `has_wallets`, `send_funds_to_user`

Example response:

```
{
   "result":"success"
}
```

Withdraw funds to an external address

Transfers funds to another address on the coin's network. The withdrawal fee that the administrator has set in the coin adapter is charged to the sender.

Ugly URI path: `/?__wallets_action=do_withdraw&__wallets_apiversion=3&__wallets_symbol=SYMBOL&__w`

Parameters:

- **SYMBOL**: The symbol of the coins to move, e.g. BTC, LTC, etc.
- **ADDRESS**: The external address to send coins to.
- **EXTRA**: Optional extra destination tag supported by some coins (e.g. Monero Payment ID, Ripple Destination Tag, etc). Can be omitted.
- **AMOUNT**: The amount of coins to transfer, excluding any transaction fees. This is the amount that the recipient address is to receive.
- **COMMENT**: A descriptive string that the sender attaches to the withdrawal. Can be omitted.
- **COMMENT_TO**: A descriptive string that the sender attaches to the address. Can be omitted.
- **WPNONCE**: The `do_move` WordPress nonce from the `get_nonces` call.

Required capabilities: `has_wallets`, `widthdraw_funds_from_wallet`

Example response:

```
{
   "result":"success"
}
```

Programmatic access to the Transactions API

To access the *Transactions API* programmatically or from the command line, use version 3 of the API or later.

Getting a key

Only users with the `access_wallets_api` capability can use the JSON API with key-based authentication. Each user is assigned one API key.

The key can be found using:

- The `[wallets_api_key]` shortcode, **while the user is logged in**.
- The *user profile* section in the WordPress admin screens.

To obtain your API key **without logging in via a browser**, you can use curl to simulate a login, then use the `get_nonces` endpoint to retrieve your key:

```
$USER=your wordpress user name
$PASSWORD=your wordpress password
```

```
curl -b cookies.txt -c cookies.txt -F log=$USER -F pwd=$PASSWORD -F testcookie=1 \
  -F wp-submit="Log In" -F redirect_to=https://www.example.com/wp-admin \
  -F submit=login -F rememberme=forever https://www.example.com/wp-login.php
```

The above commands perform a login to your site and store your session cookie in `cookies.txt`. Any subsequent JSON endpoints that you contact using this cookie file will be executed as if you were logged in via your browser:

```
curl \
    -b cookie.txt \
    -c cookie.txt \
    'http://www.example.com/?__wallets_apiversion=3&__wallets_action=get_nonces'
```

The response will contain the `"api_key"` field, a hex string. You can use this key in any subsequent calls to perform authenticated calls to the JSON API, without the need for the cookie file. Use the extra GET parameters. For example, you can do this to get your coin balances and deposit addresses using your key:

```
curl "https://www.example.com/\
?__wallets_apiversion=3\
&__wallets_action=get_coins_info\
&__wallets_api_key=5d3919d169f2049b6c2fbd5e2d886abcc1ebd5a12b23bd6dd4eebcbf084f55b1"
```

It is best not to pass the API key as a GET parameter, since the URLs of requests are often logged.

Instead, you can pass the API key as an HTTP Auth header:

```
curl \
-H "Authorization: Bearer 5d3919d169f2049b6c2fbd5e2d886abcc1ebd5a12b23bd6dd4eebcbf084f55b1"
'https://www.example.com/
?__wallets_apiversion=3
&__wallets_action=get_coins_info
```

Resetting your key

You can reset your API key. This invalidates your previous key and creates a new one.

You can do this using the “Renew” button in the `[wallets_api_key]` shortcode.

To reset the API key, you can use the `do_reset_apikey` endpoint:

Ugly URI path: `/?__wallets_action=do_reset_apikey&__wallets_apiversion=3`

Parameters: none

Required capabilities: `has_wallets`

Example response:

```
{
    "new_key":"c067801dec64449cb76a9e53dce06d502e78087bd2b3d91538404af95c8324b5",
```

```
    "result":"success"
}
```

Notification API

Notify

Notifies the wallets plugin that an event has occurred. The plugin then fires a WordPress action of the form `wallets_notify_TYPE_SYMBOL` with a single `MESSAGE` argument, and adapters can decide how to handle this. The notification API does not require login.

In practice this mechanism is useful for receiving deposits and for updating the confirmation counts of transactions.

For the bitcoin daemon, the `-walletnotify` parameter must be made to call `/wallets/notify/BTC/wallet/TXID` where `TXID` is a transaction ID, and `-blocknotify` must be made to call `/wallets/api3/notify/BTC/block/BLOCKHASH` where `BLOCKHASH` is the hash of the latest block announced on the blockchain.

In general it is the responsibility of coin adapters to inform you of how to set up notification.

Pretty URI path: `/wallets/api3/notify/SYMBOL/TYPE/MESSAGE` **Ugly URI path:** `?__wallets_action=notify&__wallets_symbol=SYMBOL&__wallets_notify_type=TYPE&__wallets_notify`

Parameters:

- **SYMBOL**: The symbol of the coin that this notification is about, e.g. BTC, LTC, etc.
- **TYPE**: The notification type, usually one of `wallet`, `block`, `alert`.
- **MESSAGE**: A string that is the payload of the notification.

Example response:

```
{
   "result":"success"
}
```

Cron trigger

Cron tasks are various periodic tasks that must run to ensure the correct operation of the plugin. This ensures that transaction submission is decoupled from transaction execution. Recurring tasks are then processed in batches.

By default, when cron tasks are scheduled to run, they run on the `shutdown` WordPress action after a user request is handled. This can still slow down frontend performance somewhat.

To trigger the cron tasks manually, first disable WordPress cron tasks using the `WP_DISABLE_CRON` constant in `wp-config.php`. Then set up a system cron task that uses curl to trigger the tasks manually at regular intervals.

When triggering the cron_job via the JSON API the WordPress action `delete_expired_transients` is also invoked. This will force deletion of some transients that should have expired. It is added as a remedy for some setups where transients do not expire properly, causing data to be cached for longer than it should.

URI path: `?__wallets_action=do_cron&__wallets_apiversion=3&__wallets_cron_nonce=NONCE`

Paramters:

- **NONCE**: A random hex string intended to prevent users from DDoSing this endpoint. You can find this nonce in the admin cron settings page.

Example response:

```
{
   "result":"success"
}
```

JSON API filters

The following set of filters modifies coin information sent via the JSON API. Use these to control how some coin information is shown in the frontend.

Coin name

- `wallets_coin_name_XYZ` - Name for coin with symbol `XYZ`.

To override the name for Bitcoin:

```
public function filter_wallets_coin_name_BTC( $pattern ) {
    return 'Bitcoin Core'; // LOL :p
}

add_filter( 'wallets_coin_name_BTC', 'filter_wallets_coin_name_BTC' );
```

Coin icon

- `wallets_coin_icon_url_XYZ` - Image URL for coin with symbol `XYZ`.

To override the URI for the Bitcoin icon:

```
public function filter_wallets_coin_icon_url_BTC( $pattern ) {
    return 'https://upload.wikimedia.org/wikipedia/commons/thumb/' .
        '4/46/Bitcoin.svg/2000px-Bitcoin.svg.png';
}

add_filter( 'wallets_coin_icon_url_BTC', 'filter_wallets_coin_icon_url_BTC' );
```

Amounts pattern

Cryptocurrency amounts in the frontend are passed through an `sprintf()` function. The JavaScript implementation used is https://github.com/alexei/sprintf.js.

- `wallets_sprintf_pattern_XYZ` - Display pattern for amounts of coin with symbol `XYZ`.

To override the frontend display format for Bitcoin amounts:

```
public function filter_wallets_sprintf_pattern_BTC( $pattern ) {
    return 'BTC %01.8f';
}

add_filter( 'wallets_sprintf_pattern_BTC', 'filter_wallets_sprintf_pattern_BTC' );
```

Change the 8 digit to how many decimal digits you want displayed.

Blockexplorer Transaction URI pattern

`wallets_explorer_uri_tx_XYZ` - Use this to select a different blockexplorer for viewing transactions on the blockchain. The string `%s` will be replaced with the transaction ID.

To use chain.so for Bitcoin transactions:

```
public function filter_wallets_explorer_uri_tx_BTC( $pattern ) {
    return 'https://chain.so/tx/BTC/%s';
}

add_filter( 'wallets_explorer_uri_tx_BTC', 'filter_wallets_explorer_uri_tx_BTC' );
```

Blockexplorer Address URI pattern

`wallets_explorer_uri_add_XYZ` - Use this to select a different blockexplorer for viewing addresses on the blockchain. The string `%s` will be replaced with the address.

To use chain.so for Bitcoin addresses:

```
public function filter_wallets_explorer_uri_add_BTC( $pattern ) {
    return 'https://chain.so/address/BTC/%s';
}

add_filter( 'wallets_explorer_uri_add_BTC', 'filter_wallets_explorer_uri_add_BTC' );
```

Wallets PHP API

The Bitcoin and Altcoin Wallets WordPress plugin offers a PHP API that is accessible to other themes and plugins.

Purpose of the PHP API

- You can access the API from your theme to perform transactions on behalf of your users.
- You can develop plugins that utlize the PHP API.
- You can install dashed-slug plugin extensions that use this API.

Online PHPdoc

Consult the auto-generated PHPdoc for full documentation and examples at: http://wallets-phpdoc.dashed-slug.net/

Local copy of PHPdoc

The PHPdoc is also included in the bundle `.zip` file. This is the same download where you found this reference manual. Extract the `.zip` file, then open `api-phpdoc/index.html` with your browser. Finally, navigate to the documentation for the `Dashed_Slug_Wallets_PHP_API` class.

IMPORTANT NOTICE: Every piece of code that your site runs can access this API and perform transactions. Do not install themes or plugins from unknown sources without checking the code thoroughly. You should be already aware of this.

Exchange rates

The plugin can use internet services to pull exchange rates. This helps the plugin and its extensions convert amounts to different currencies.

App extensions to the **Bitcoin and Altcoin Wallets** plugin, such as the **WooCommerce** and **Events Manager payment gateways**, use exchange rates for price calculation.

Calling from PHP

Rates are retrieved with the following API call:

```
Dashed_Slug_Wallets_Rates::get_exchange_rate( $from, $to );
```

Admin settings

You can choose which APIs will be used to pull exchange rates between various cryptocurrencies.

Go to *Wallets → Exchange rates* and select the exchange rate providers to use. You can choose multiple providers.

To convert between fiat and cryptocurrencies, you will need to setup the fixer service. This requires an API KEY that you can get for free from their service.

Set the *Rates cache expiry* to control how often new exchange rates are downloaded.

- New exchange rate data is always loaded after the user request, on PHP shutdown.
- Until new data is downloaded, the old data is used.
- New data is loaded when the time specified by the admin has elapsed.
- For the fixer.io service, data is only loaded once per hour to avoid hitting the limit of 1000 requests per month.

Payment gateways on checkout will only show coins where

- the coin adapter is enabled, and
- the coin is listed on one of the selected rates providers

There are options for tunnelling requests through tor. This is only useful if you are setting up a hidden service. In all other cases, keep this disabled.

Finally there are three views of the the downloaded data. These are there for debugging.

After any change, scroll down and click on *Save Changes*.

User profile settings

Each user can select to override the *Default fiat currency* setting in the admin settings (above).

Each user can navigate from the admin interface to *Users → Bitcoin and Altcoin Wallets → Fiat currency*.

Available options:

- **None.** — Cryptocurrency amounts will not be shown in any equivalent currency amount.
- **Site default** — Cryptocurrency amounts will also be shown as the equivalent in the currency selected by the admin (*Wallets → Exchange Rates → Default fiat currency*).
- **XYZ** — Cryptocurrency amounts will also be shown as the equivalent in the XYZ currency, where XYZ is a fiat currency symbol selected by the user. All known fiat currencies are shown as options here.

Pluggable exchange rate providers

You can also write your own code to add more exchange rates and currencies.

See here for an example: https://gist.github.com/alex-georgiou/492196184f206002c864225180ca8fbb

Cache expiry

If you write code that contacts an external API, make sure to set your cache timeout to the value of

```
Dashed_Slug_Wallets::get_option( 'wallets_rates_cache_expiry', 5 );
```

That is all.

Multi-site (aka network) installations

As of version 2.2.3, the **Bitcoin and Altcoin Wallets** plugin can be installed on multi-site WordPress installations.

As of version 2.4.0, the plugin will behave differently depending on whether it is *network-activated*, or activated on individual blogs.

Activating on individual blogs

The *superadmin* installs the plugin via the network administrative pages and **does not** network-activate it.

Administrators can then activate the plugin on individual blogs and view the *Wallets* menu.

User roles who have the `manage_options` capability (usually administrators) will gain the `manage_wallets` capability and be able to configure settings, setup adapters, view and approve transactions, etc.

All settings can be set to be different between blogs.

Users will maintain separate wallets on each blog. Transactions stored in the database are bound to a blog and only appear on that blog.

Administrators can choose to install *coin adapters* or *feature extensions* on a blog-by-blog basis.

Network-activating

The *superadmin* installs the plugin via the network administrative pages and proceeds to network-activate it.

Site administrators can not activate or deactivate the plugin on individual blogs, nor can they view the *Wallets* menu.

User roles who have the `manage_network` capability (usually super-admins) will gain the `manage_wallets` capability and be able to configure settings, setup adapters, view and approve transactions, etc. Site administrators will not be able to configure wallets settings.

Users will maintain one balance per coin accross the network.

Transactions stored in the database of the network can appear on all blogs in the network.

The network admin can choose to install *coin adapters*.

Blog administrators cannot install *coin adapters*, but they can install *feature extensions* on a blog-by-blog basis.

Troubleshooting

Before contacting support you can have a look at these common pitfalls.

If you do contact support, please go to the Dashboard (top left in the admin screens) and find the plugin's widget. Copy the debug info and paste it into your query. This will let me know what type of system the plugin runs on.

I do not see the UI elements in the frontend.

1. Check that you have added the shortcodes correctly or that you are displaying the widgets in the right widget area.
2. Check that you are logged in.
3. Check that you are logged in as a user with the right capabilities, including `has_wallets`. The needed capabilities for each UI element are listed in the *Shortcodes* section of this document.
4. Check that at least one coin adapter is online. Go to *Wallets* → *Adapters* and check the *status* column of each adapter.
5. Check your browser's JavaScript console for any errors (not warnings). If any script on your page has halted execution of JavaScript scripts then this would cause the UIs to not load.
6. If using the CoinPayments adapter, check to make sure that the API key to your cloud wallet has the correct permissions, as recommended here.

If you have checked all of the above:

5. Contact support. If possible, check your JavaScript console for errors. If you find any, report them when you describe your issue.

I see the UI elements but no coins.

1. Check that there is at least one adapter with status "online".
2. Contact support. If possible, check your JavaScript console for errors. If you find any, report them when you describe your issue.

I am trying to test the plugin but do not want to spend money on transaction fees every time I test deposits or withdrawals.

You can perform most of your tests using testnets. Both Bitcoin and Litecoin have robust testnets that people are always mining.

- For RPC adapters, set `testnet=1` and restart your wallet.
- For the *CoinPayments adapter*, use the LTCT symbol for Litecoin testnet.
- For the *block.io adapter*, set the API keys to the testnet wallets provided by your account.

- For the *bittiraha wallet*, edit `testnet.conf` and set `enabled=1` and `port=18332`. Then restart your wallet and set the *Bitcoin core node* adapter to connect to that port.

Make sure to delete deposit addresses from the DB, as testnet addresses are different from mainnet addresses.

After configuring your adapter to connect to a testnet, use testnet faucets to perform deposits and withdrawals.

Bitcoin testnet faucets:

- https://testnet.manu.backend.hamburg/faucet
- http://bitcoinfaucet.uo1.net/

Litecoin testnet faucet:

- http://testnet.litecointools.com/

I am testing the plugin in my development environment and I am not getting any email notifications.

First make sure that notifications are enabled in *Wallets* → *Notifications*. They should be enabled by default.

This is probably not an issue with the plugin, but with your OS setup. Unless you have set up sendmail on your system, emails will not work.

Since you are testing in a development environment, it makes sense that sendmail would not be set up, while on an actual live server provided by a web host, emails would work.

The easiest workaround would be to forward emails through a gmail account. You can setup a plugin such WP Mail SMTP.

Another option would be to test with *notifications* sent as BuddyPress private messages. However you would not be able to get *confirmation emails*, as this would not make sense from a security standpoint. If you choose to test the plugin using BuddyPress for this reason, you would have to disable confirmations. See the glossary section for the difference between notifications and confirmations.

I can't find the *Wallets* menu in the admin screens even though the plugin is listed as installed and activated.

There's only a few reasons why the wallet menu would not show up:

On a multisite install if the plugin is network-activated, then the *Wallets* menu will be in the network admin pages, not at the blog level admin pages. If this is not what you wanted, first

network-deactivate the plugin, and then activate the plugin on each blog where you want to use it.

Secondly, the menu appears only to uses who have the `manage_wallets` capability. The plugin normally sets this capability to the administrator on activation. You can check if your user has this capability with this plugin: Capability Manager

Assuming the above is not the case, this would only happen if the plugin was not installed properly for whatever reason.

Maybe files are missing, or maybe something is very wrong with the database. In any case there must be errors in the logs.

I am trying to change the UIs by editing my shortcodes or my Widget settings, but the frontend remains the same no matter what.

If you have installed a caching plugin such as *WP Super Cache*, try clearing your server cache.

I cannot connect to a wallet running on my home computer

Unless your home internet connection has a static IP and you have opened the correct ports on your router/firewall you will not be able to use your home computer as a wallet backend. In fact, this is not a very good idea. Ideally you need a dedicated server to run a wallet with the availability required by a site that serves multiple users. Virtual private servers (VPSs) should be OK as the wallets do not max-out CPU usage under normal operation, after the blockchain finishes syncing. Shared/managed hosting plans are not appropriate for running wallet daemons. If you have a shared/managed hosting plan (i.e. no ssh access), you are stuck with using web wallets.

Check with your hosting plan for disk space and memory requirements against the requirements of the wallet you wish to run. For Bitcoin core, click here.

Under *Wallets* → *Adapters*, the status of my RPC coin adapters is *not responding*.

- The error message includes *403 - forbidden*: The port you are trying to connect to is not open for your WordPress machine. This would indicate that your `rpcallowip` setting is not correct. Check your IP and port settings, your firewalls and lastly your host's firewalls. The unix commands `telnet` and `nc` are your friends in debugging this.
- The error message includes *301 - unauthorized*. The username and password that you provided does not match your `rpcuser` and `rpcpassword` settings, or your `rpcauth` setting. Remember that you are to EITHER set `rpcuser` and `rpcpassword`, OR `rpcauth` but not both.
- If there is another error message, check the error message to get a hint as to what is going wrong. Contact support if needed.

I have setup an RPC adapter correctly (status=responding). When I perform a deposit, the deposit never shows up in the plugin, even after waiting for a few minutes.

Deposits are discovered via two ways, polling on cron, and via curl using the wallet's notification mechanism. It is unlikely that both of these mechanisms don't work. Check to see first if the wallet has downloaded the entire blockchain. If it is stuck or is simply still downloading, then it would not know about any recent transactions. For Bitcoin, you can check the block count with `bitcoin-cli getblockchaininfo`. The value of `blocks` should match the current block height for the network. Check https://blockchain.info/latestblock for Bitcoin, or a similar block explorer service for any other coin.

There are withdrawal transactions entered in the system but they do not get processed.

If the withdrawals are in the *unconfirmed* state: The transactions may need to be confirmed either by an admin, or by the user, or both. Read the *Confirmations* section of this document, in the *Transaction* chapter.

If the withdrawals are in the *pending* state: Check that the adapter is unlocked. Read the *Withdrawals and wallet locks* section of this document, in the *Transaction* chapter.

I get the following error in the frontend: `Could not contact server. / Status: parseerror / Error: SyntaxError: JSON.parse: unexpected character at line 1 column 1 of the JSON data`

You would normally get this if:

1. you have enabled logging in `wp-config.php` (`WP_DEBUG` is `true`), and
2. you have not set `WP_DEBUG_DISPLAY` to `false`) and
3. any other plugin or theme on your installation is writing any text to the logs for any reason.

The plugin's frontend performs a number of XHR requests to the WordPress server, and the responses should be valid JSON. If any errors are written to the output, that would cause the responses to not be valid JSON. Set `WP_DEBUG_DISPLAY` to `false` in your `wp-config.php`.

I got an email from fixer.io with the message "Please be aware that you have reached at least 75% of your available API Request volume."

The exchange rates API uses the fixer.io service to determine exchange rates to fiat currencies. You would get this message typically if you have connected your website to a fixer account with a free plan.

The free plan for fixer.io gives you 1000 API calls per month. The plugin pulls data from fixer once per hour. For a 30 day month this comes out to be 30×24, or 720 calls per month. The plugin will do these calls whether the site is being used or not, as long as it is online. If you are using the plugin only on one site, the limit will not be exceeded. You can always check your fixer account to see how much of the limit you have been using. Simply login to the service and go to your dashboard to check: https://fixer.io/dashboard

If you are using the same API key with multiple sites then you might have to consider upgrading to another plan.

When attempting a withdrawal via a Full Node wallet (Bitcoin core or other coin's full node wallet), I get a failure with "This transaction requires a transaction fee of at least X".

It is possible that your wallet's default transaction fees are currently not set to a value that is reasonable for the blockchain network that you are using. Please consult your wallet's documentation to set the appropriate transaction fees. For Bitcoin, the relevant settings would include `paytxfee`, `mintxfee`, and `fallbackfee`. Alternatively you can use `txconfirmtarget`. After setting the fees correctly, make sure that the withdrawal fee you have set on your coin adapter's settings is the same or higher than what you set to your wallet.

When I activate the plugin, my WordPress becomes somewhat slow but is still usable.

If you can still use WordPress, go to *Wallets → Exchange rates* and set the *Rates provider* to *disabled*. If this makes WordPress fast again, this means that the delay was coming from the exchange rates API. Try experimenting with other rates providers.

You can also go to *Wallets → Cron job* and set the *Run every* interval to a higher value. The plugin and its extensions perform miscellaneous maintenance tasks on every cron heartbeat. You can control how often these tasks are performed.

I have finished testing the plugin and am now ready to use it in production. I would like to clear any deposit addresses and transactions that I used in testing.

You should simply empty your transactions table and your deposit addresses table. Assuming your DB prefix is `wp_`, you would do the following in your SQL console:

```
DELETE FROM wp_wallets_txs;
DELETE FROM wp_wallets_adds;
```

When I activate the plugin, my WordPress becomes extremely slow and is unusable, or I get a white screen of death, or I get a 5xx HTTP error.

In the event that WordPress has become unusable right after installing this plugin, you can regain control of your WordPress if you simply delete the plugin.

- You will have to delete the `wp-contents/plugins/wallets` directory.
- You will not lose your settings unless you run the plugin's uninstall script.
- You will not lose any transaction data or deposit addresses, even if you run the uninstall script.
- You can reinstall the plugin after diagnosing the problem.

The plugin contacts a number of third-party endpoints for its normal operation. If these connection attempts timeout (instead of succeeding or failing) then this will typically incur a slowdown of about 30 seconds per each connection attempt, making the plugin and WordPress unusable. Connection timeouts are usually caused due to firewalls.

- Make sure that you have only enabled tor settings if your know that your site is a hidden site running on tor.
- If you are connecting to an RPC wallet situated on a different machine than your WordPress server, make sure that your webhost allows outgoing connections to TCP ports other than 80 and 443.
- If you are the administrator of the WordPress machine, check your firewalls for any rules that may interfere with outgoing connections. This can include hardware firewalls, software system firewalls and any WordPress security plugins.

After minifying the HTML output of my webpage, the UI no longer works

The UIs use the `knockout.js` library, and its *containerless control flow syntax*. This means that there are some HTML comments in the markup that have semantics necessary for the operation of the UI. These are comments of the form `<!-- ko foo: bar -->` and must be preserved.

The frontend balances are not updating unless I refresh the cache, even though polling is enabled.

Another plugin could be causing the output of the JSON API to be cached. Super Progressive Web Apps has been known to cause problems.

Can I continue using a coin that has transactions on it using a different wallet backend?

It is possible to mix-and-match any adapters, as long as you do not use two adapters for the same coin. So, for example you could have BTC and LTC on block.io, and ETH on CoinPay-

ments.

If you have balances on one wallet and you need to transfer the balances to another, do this:

1. Deactivate the plugin.
2. Transfer the entire balance from that coin to the other wallet. Do not use a user deposit address. Use another address generated by the receiving wallet.
3. Activate the plugin.
4. Connect the new coin adapter for that coin.
5. Go to *Wallets → Adapters* and click on *Renew deposit addresses* under the new adapter name. New deposit addresses will be generated for your users on demand.

You should be good to go!

Something else is wrong. I would like to check the logs for any errors that might give me clues as to what's going on.

Here are instructions on how you enable logs: Debugging in WordPress

Make sure that you have `define( 'WP_DEBUG', true );` and `define( 'WP_DEBUG_LOG', true );`. This will write to `wp-content/debug.log`. You will want to set `define( 'WP_DEBUG_DISPLAY', false );`. Otherwise any error or warning will cause the JSON API to not parse.

The plugin always writes to the logs on activation. It would write something like: `Upgrading wallets schema from X to Y.` and `Finished upgrading wallets schema from X to Y.`. If it does not write to that file while you are activating the plugin, then your logging is not working properly. Check your `wp-config.php` and the filesystem owner/permissions on the `wp-content` directory. It must be writable by the user of your webserver daemon.

Coin Adapter development

In *Bitcoin and Altcoin Wallets*, connectivity to the various local wallets and cloud wallets is provided by *Coin adapters*. These are WordPress plugins that extend Bitcoin and Altcoin Wallets.

Coin adapters do the following:

1. Perform the on-chain transfers (withdrawals).
2. Notify the plugin for any incoming deposits.
3. Provide information about a coin and the wallet service.
4. Provide settings that control communication with the wallet.

Coin adapters do not worry about accounting or internal fund transfers. This is done by the main plugin.

Instructions for Bitcoin-like coins

For Bitcoin-like coins, previously the recommended method was to implement a full node adapter by modifying the Litecoin full node adapter.

With the release of the *Full Node Multi Coin Adapter extension* this task becomes easier:

1. Install the *Full Node Multi Coin Adapter extension*. This coin adapter loads data for a number of coins from a `coins.csv` file.
2. Use a WordPress filter anywhere in your code (in your theme's `functions.php` file, or anywhere else) to append to that data. For example, here is how you would add support for Litecoin (you do not have to add Litecoin because it is already listed in the `coins.csv` file):

```
function wallets_multiadapter_coins_filter( $coins ) {
    $coins['LTC'] = array(
        // Coin symbol
        'symbol' => 'LTC',

        // Coin name
        'name' => 'Litecoin',

        // Default withdrawal fee (coin adapter settings override this)
        'wd fee' => '0.005',

        // Default internal transaction fee (coin adapter settings override this)
        'move fee' => '0.0005',

        // Default min confirmation count required for deposits (coin adapter settings
        'confirms' => '12',

        // Default RPC port (coin adapter settings override this)
```

```
            'port number' => '9332',

            // Whether the wallet supports -walletnotify
            'tx notify' => '1',

            // Whether the wallet supports -blocknotify
            'block notify' => '1',

            // Whether the wallet supports -alertnotify (some wallets have deprecated this)
            'alert notify' => '0',

            // Comma separated list of hex bytes, needed for frontend validation of withdra
            'versions' => '0x30',

            // An sprintf() pattern for deposit address QR Code URI. If unsure, set to '%s'
            'qr pattern' => 'litecoin:%s',

            // An sprintf() pattern for displaying amounts. If unsure, leave to '%01.8f'.
            'amount pattern' => '%01.8f',

            // Default sprintf() pattern for URI to block explorer transaction page. Can be
            'explorer tx uri' => 'https://live.blockcypher.com/ltc/tx/%s/',

            // Default sprintf() pattern for URI to block explorer address page. Can be ove
            'explorer address uri' => 'https://live.blockcypher.com/ltc/address/%s/',

            // URL to an 64x64 icon for the coin. Or leave empty to pull the icon from 'ass
            'icon url' => '',
        );
        return $coins;
    }

    add_filter( 'wallets_multiadapter_coins', 'wallets_multiadapter_coins_filter' );
```

3. Edit the coin adapter settings via your admin screens and make sure they match with the settings you entered in your full node wallet's `.conf` file.

Coin adapter simple example

You would only need to develop your own coin adapter in case the *Full Node Multi Coin Adapter extension* is not suitable for your coin.

When the `wallets_declare_adapters` action is triggered, include a file that will contain the adapter.

```
function myplugin_wallets_declare_adapters() {
```

```
    include 'myplugin-coin-adapter-class.php';
}

add_action( 'wallets_declare_adapters', 'myplugin_wallets_declare_adapters' );
```

The adapter itself must be a class that derives from `Dashed_Slug_Wallets_Coin_Adapter`.

The class must have a no-argument constructor. You do not instantiate your class. *Bitcoin and Altcoin Wallets* will create a single instance of each class you declare.

Simply study the `Dashed_Slug_Wallets_Coin_Adapter` class file and override all methods needed in your implementation.

For example, a Litecoin adapter might begin as follows:

```
class MyPlugin_Adapter extends Dashed_Slug_Wallets_Coin_Adapter {

    public function get_symbol() {
        return 'LTC';
    }

    public function get_name() {
        return 'Litecoin';
    }

    // etc...

}
```

The adapter's submenu normally appears under the wallets menu.

On the action `admin_menu`, *Bitcoin and Altcoin Wallets* triggers the `wallets_admin_menu` action.

Adapters normally bind the `action_wallets_admin_menu()` function to this action upon construction. This is the function that binds the adapter settings.

You might want to override it to provide additional settings:

```
public function action_wallets_admin_menu() {
    parent::action_wallets_admin_menu();

    // bind additional settings...
}
```

Or to hide the settings altogether and possibly provide your own page.

```
public function action_wallets_admin_menu() {   }
```

Coin adapters class hierarchy

Figure 2: Coin adapters class hierarchy

You do not have to extend directly from `Dashed_Slug_Wallets_Coin_Adapter`.

`Dashed_Slug_Wallets_Coin_Adapter_RPC`

If your wallet is a Bitcoin-like wallet daemon with an RPC API, you might want to derive from `Dashed_Slug_Wallets_Coin_Adapter_RPC`. This is an abstract class that uses EasyBitcoin-PHP to communicate with a wallet.

`Dashed_Slug_Wallets_Coin_Adapter_JSON`

If you need to do GET or POST requests to an API that returns JSON, consider deriving from `Dashed_Slug_Wallets_Coin_Adapter_JSON`. This provides the following helpers: `get_json()`, `get_json_cached()`, `post_json()`, `post_json_cached()`.

Locked/unlocked state

Coin Adapters can be locked, meaning that the wallet cannot process withdrawals. The user can then unlock the adapter, usually with a passphrase that enables withdrawals. Implement `is_unlocked()` so it returns true only when the wallet can process withdrawals. You do not need to do this if you derive from `Dashed_Slug_Wallets_Coin_Adapter_RPC`, as the mechanism is already implemented in the abstract class.

Withdraw address validators

You can bind JavaScript validator functions to coins. Whenever a user enters a withdrawal address into the `[wallets_withdraw]` UI, your function can test that address for consistency and warn the user if the address does not pass validation. For example, here's how to hook a validation function for Litecoin:

```
$.fn.walletsBindWithdrawAddressValidator(
    'LTC',
    function ( val ) {
        var bytes;

        try {
            bytes = bs58check.decode( val );
        } catch ( e ) {
            return false;
        }

        if ( bytes.length != 21 )
            return false;

        var version = bytes[0];

        return 0x30 == version;
    }
);
```

The above code binds a validator function to the `LTC` symbol. The code does a bs58check and a version check to validate a Litecoin address. The `bs58check` object from bitcoinjs is always available.

To make sure that your code runs after the wallets code, enqueue it with these dependencies:

```
function action_wp_enqueue_scripts() {
    wp_enqueue_script(
        'wallets_litecoin_validator',
        plugins_url( 'wallets-litecoin-validator.js',
            "wallets-litecoin/assets/scripts/wallets-litecoin-validator.js" ),
        array( 'wallets_ko', 'bs58check' ),
```

```
            false,
            true
        );
    }
    add_action( 'wp_enqueue_scripts', 'action_wp_enqueue_scripts' );
```

This ensures that your script only runs after `$.fn.walletsBindWithdrawAddressValidator` and `bs58check` are loaded.

Glossary

Here are some terms related to *Bitcoin and Altcoin Wallets*.

> *PRO TIP:* When contacting the *dashed-slug* for support or other inquiries, it is useful to have this glossary handy until you are fluent with the language of the dashed-slug ;-)

Accepted by admin — A type of transaction *confirmation*. For either *withdrawals* or *internal transfers* the admin can be required to provide *confirmation*. In this case, the admin has to go to the *Wallets&* → Transactions* admin panel and click next to a transaction to let it proceed.

Adapter — See *Coin Adapter*.

App extension — An *extension* to *Bitcoin and Altcoin Wallets* that provides useful functionality. Contrast with *Coin adapter extension*.

Capabilities — 1. In the general context of WordPress, see the Codex definition of Roles and Capabilities 2. In the context of the *Bitcoin and Altcoin Wallets* plugin, the WordPress capabilities that begin with the slug `wallets_` and control access to wallet actions and management.

Cloud wallet adapter — A *coin adapter* that uses an online wallet service rather than a standalone wallet node. Offers ease of use and minimal maintenance. The *CoinPayments Adapter* and the *block.io Adapter* extension are examples of *cloud wallet adapters*.

Cold storage — A wallet that holds a part of the website's funds and is offline. That is, there is no way for the system where WordPress runs to withdraw funds from the cold storage wallet even if the system is compromised. Only the site operator should be able to transfer funds to and from cold storage.

Confirmation — 1. In the general context of cryptocurrencies, see the Bitcoin wiki page for Confirmation 2. In the context of the *Bitcoin and Altcoin Wallets* plugin, the transaction attributes *Accepted by admin* and *Verified by user*.

Coin adapter — Communicates with a cryptocurrency wallet to provide deposit and withdraw support for a coin to the *Bitcoin and Altcoin Wallets* plugin.

Coin adapter extension — An *extension* to *Bitcoin and Altcoin Wallets* that provides one or more *Coin Adapters*.

Cron — A mechanism of *Bitcoin and Altcoin Wallets* where on every heartbeat of the plugin, a number of transactions are executed and *coin adapters* can perform their own actions such as detecting transactions or other checks. Normally *cron* is triggered by the wp cron mechanism, which in turn can be triggered by a Linux cron scheduler if needed. However the *Bitcoin and Altcoin Wallets* cron is guaranteed to run eventually even in installations where the other mechanisms are absent.

dashed-slug — Software house that churns out cryptocurrency plugins for WordPress at an alarming rate!

Deposit — A *transaction* whereby a user sends money from some other wallet on the cryptocurrency network to their *deposit address* on your website.

Deposit address — A cryptocurrency address where a user can send coins to perform a *deposit*. Every user has one *deposit address* for each *coin adapter* and a new address is generated on the wallet by the *coin adapter*, if one does not exist.

Extension — A plugin that extends *Bitcoin and Altcoin Wallets*. See *App extension* and *Coin adapter extension*.

Fees — Fees are subtracted from user accounts and remain with the wallet. For every coin, there is one wallet that is split between all users with the has_wallets capability. In the case of withdrawals, the fees are first used to pay for the blockchain's network fees, and any remaining fees stay with your site's wallet. In the case of internal transfers between users, the entirety of the paid fees is subtracted from the sender's balance and remains with your wallet, since there is no blockchain transaction. When you visit the *coin adapters* screen in your admin area, you can see the total of user balances for each coin, and the balance of the wallet. These amounts do not have to be the same. As users perform transactions and pay fees, these fees will be subtracted from the user balances, but not from the wallet balance.

Unavailable balance — The part of a user's balance that is locked in either pending *transactions* or in Exchange market orders that have not yet been filled. This balance cannot be withdrawn or traded until the transaction or order is executed/cancelled/failed. Internally, *app extensions* can use the `wallets_api_unavailable_balance` filter of the *PHP API* to add to this unavailable balance.

Hot wallet balance — The total amount of coins available to the site for withdrawals. This does not need to be the same as the total sum of *user balances*.

Internal transfer — A transfer between two users in the same WordPress installation, either in the same blog or accross blogs in a multisite install. The transfer can cost any amount of fees that the administrator chooses, including zero fees. The transaction is only stored in the local database and is not broadcast to the network. In fact, the *coin adapter* is not notified about internal transfers at all. Users can initiate internal transfers using the `[wallets_move]` shortcode. *App extensions* such as the *WooCommerce Payment Gateway extension* use internal transfers to perform transactions.

IPN — Stands for Instant Payment Notification. In the context of the CoinPayments *multi-adapter*, the mechanism that lets the CoinPayments platform to inform *Bitcoin and Altcoin Wallets* of incoming user deposits.

JSON API — An HTTP-based API that provides an interface between the frontend JavaScript code and the plugin's backend. The JSON API exposes information provided by the PHP API to logged in users.

Move — See *Internal transfer*. The term *move* is used throughout the sourcecode for brevity.

Multi-adapter — A *coin adapter extension* that provides more than one *coin adapter*.

Notification — A message sent to a user to inform them of some event that affects their balance in some way, such as a *deposit*, *withdrawal*, or *internal transfer*. The message reports whether the transaction succeeded or failed, and includes an error message in case of failure. It can be sent over e-mail or, since version 2.8.3 via the BuddyPress Private Messaging component.

PHP API — Collection of WordPress actions and filters that can be used to interact with the wallets via PHP code. The PHP API is documented using phpDoc. The *JSON API* utilizes this PHP API.

Symbol — A short, usually three-letter string of letters that identifies a currency. e.g. “BTC” for Bitcoin.

Transaction — A *deposit*, *withdrawal*, or *internal transfer* that affects user balances (i.e. excluding *cold storage* deposits and withdrawals.

Transaction fees — These are the fees a user pays when they send funds to other users.

Transaction status — One of the following: *Unaccepted* is a *transaction* that has not been *Accepted by admin* or *Verified by user*. *Pending* is a *transaction* that has been, and is ready to be executed. *Done* is a transaction that has been performed successfully. *Failed* means the transaction did not succeed due to some error. *Cancelled* is a transaction that was cancelled manually by an admin with the `manage_wallets` capability.

Unavailable balance — The portion of the *Hot wallet balance* that is currently unavailable for withdrawals. Reasons for it being unavailable are: unconfirmed transactions, PoS staking, newly minted coins, immature balance, etc.

User balance — The total sum of all *transactions* that a user has performed. *Deposits* and received *internal transfers* are positive, while sent *internal transfers* and *withdrawals* are negative. Thus the total sum is equal to the user’s funds.

Verified by user — A type of transaction *confirmation*. For either *withdrawals* or *internal transfers* the user that originated the transaction can be required to provide *confirmation*. This is done via a nonce that is sent to the account owner over e-mail for added security.

Withdrawal — A *transaction* whereby a user sends money from your website to another address on the cryptocurrency network. If that address is a deposit address of some other user on the same website, an *internal transfer* is done instead.

Withdrawal fees — This the amount that is subtracted from a user’s account when they withdraw funds to an external blockchain address. The amount should be larger than any network fees that the wallet will pay for a typical transaction, because the network fees are paid from these withdrawal fees.

Contact and support

Please use the support forum on WordPress.org for all issues and inquiries regarding the plugin.

Please use the appropriate support forum for the plugin's extensions.

You are welcome to send in any problems, questions, suggestions, thoughts, etc.

For all other communication, please contact info@dashed-slug.net.

When requesting support, it is helpful if you include some info about the problem you are facing and your system. You can go to the WordPress Admin Dashboard, and in the widget for *Bitcoin and Altcoin Wallets* you will find information about your setup. You can copy this information and paste it into your support request.