

Bitcoin and Altcoin Wallets 3.5.4 reference manual

dashed-slug info@dashed-slug.net

Abstract

Integrating cryptocurrency wallets into your WordPress site, using the **Bitcoin and Altcoin Wallets** plugin, version **3.5.4**.

At a glance

Turn your blog into a bank: Let your users deposit, withdraw, and transfer bitcoins and altcoins on your site.

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 this manual to familiarize yourself with some basic concepts.

Read the readme to understand the tradeoff between setting up a full node or using the cloud wallets.

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

A full node is harder to setup and maintain, but gives you performance and freedom to control network fee settings.

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 then the installation instructions are on the coin adapter page. If you prefer to install the block.io coin adapter then the installation instructions are on that other coin adapter page. You will also find a troubleshooting section on these pages, and the support forums are [here](#) and [here](#).

The troubleshooting section for the main plugin is also found in this document. The support forum for the main plugin is at [WordPress.org](#).

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 strings 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.

How to install with a Bitcoin full node

*The installation for the plugin itself is the same as for any WordPress plugin. Additionally, **you will have to install and maintain a Bitcoin daemon on your server**. This will typically require SSH access and some basic knowledge of UNIX/Linux.*

To Install the plugin and connect it to a Bitcoin full node using the built-in Bitcoin adapter:

1. Make sure that you have **the latest WordPress version** installed, and that you are running on **at least PHP 5.6**. Even though the plugin has been tested on WordPress 4.0 and PHP 5.3, for security reasons you are **strongly** recommended to use the latest version of WordPress and a supported version of PHP. Check to see [here](#) if your PHP version is currently supported for security issues. As of 2017, anything below 5.6 has reached its end-of-life and is no longer supported.
2. **Install and activate the Wallets plugin**. For general information on installing WordPress plugins, you can consult the relevant WordPress documentation.
3. **Install a Bitcoin full node** on your server. Detailed instructions are available [here](#). Read and follow the instructions carefully.

Take note of the memory, disk, and bandwidth requirements and check against the resources available on your server. If you find that running a full node is too heavy on your server's resources, or if you do not have the technical expertise to work with Linux and the command line, or if you only have access to shared hosting, please see the FAQ section below for alternative options.

4. **Configure the bitcoin adapter on your WordPress installation**. Navigate to *Wallets* → *Bitcoin (BTC)* in your WordPress admin area.

At a minimum you need to enable the adapter and enter the location and credentials to your *Bitcoin daemon RPC API*.

You will need to set the following: IP, Port, User, Password, Path.

5. **Configure the bitcoin daemon on your server**. You will need to edit your `~/.bitcoin/bitcoin.conf` file and make the configuration match what you entered above. The plugin will give you the exact configuration arguments that you need to start the daemon with.

For more information on the bitcoin daemon configuration, consult the relevant [wiki page](#).

6. **Check that the adapter works**. Navigate to the *Wallets* menu in the admin area. If the *Bitcoin Adapter Status* reads *Responding*, then you're good to go.

Note that for a new bitcoind installation, you might have to wait until the entire blockchain downloads first. This can take a few hours. Again, skip to the FAQ section for other alternatives.

Frequently Asked Questions

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.

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.

Is it secure?

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 am encountering some problem with the Bitcoin and Altcoin Wallets plugin

You should first have a look at 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 use the appropriate forum and post a new thread for each distinct issue.

Do I really need to run a full node? 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.

Cloud wallets

Instead, you can choose to install one of the available coin adapters that are backed by cloud wallet services. These currently are:

- The CoinPayments Adapter extension
- The block.io Cloud Wallet Adapter extension

Study the services and their terms of service including what fees they charge before choosing to use them.

bittiraha

From version 1.1.0 onward, this plugin is compatible with the bittiraha-walletd wallet. From the project's description on GitHub:

Lightweight Bitcoin RPC compatible HD wallet This project is meant as a drop-in replacement for bitcoind for use in lightweight servers.

This is a wallet based on bitcoind and does not store the blockchain locally. You will have to install this on a VPS or other server via the shell.

A downside is that the walletnotify mechanism and the listtransactions command are not implemented. **This means that there is no easy way for the plugin to be notified of deposits.** Deposits will not be recorded in the transactions table. Users will not be emailed when they perform deposits and they will not be able to see their deposits in the [wallets_transactions] UI. Deposits will correctly affect users' balances. You have been warned.

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.

Keep in mind that no software is set-and-forget. Once you install 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 should not be running a wallet with people's money on it.

Remember that you have two options: stand-alone wallets or web 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 web wallet.

How can I integrate the plugin with my site?

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

- **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.

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.
2. If you wish to translate the form texts to a different language, see the *Localization* section of this manual.
3. If you wish to change the texts to something else, you may use the `wallets_ui_text_*` WordPress filters.
4. If you wish to create forms with completely different markup, you can provide your own views for these shortcodes. Use the `wallets_views_dir` filter to override the directory where the views are stored (the default is `wallets/includes/views`). Most people will not need to do this.

Read the *Frontend* → *Alternative knockout templates* section of the user manual for more details.

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.

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 can no longer cater to requests to add new coin adapters. I can only provide assistance by answering specific questions to coin adapter developers.

If your coin's wallet has a standard RPC API that is a direct fork of Bitcoin core, then you should be able to use the Full Node Multi Coin Adapter extension.

If your coin is an ERC-20 token then there is no support for that at the moment.

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.

Can you add fiat currency deposits?

I do not have plans to add fiat currency deposits. That is not to say that someone cannot develop an extension to do this.

Can you build an exchange on top of the plugin?

Yes this is in the future plans, but it is a huge undertaking! For now you can use the ShapeShift app extension that lets you display a front-end UI to the ShapeShift.io service.

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

I plan to build a plugin extension that will allow you to pay for membership via cryptocurrencies. When ready, this extension will also be made available as a dashed-slug premium extension. In the meantime, you may contact me directly at info@dashed-slug.net if you wish to send a Bitcoin payment and I will activate your membership manually.

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.

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 API.* 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`.

Deposit funds

Required capabilities: `has_wallets`

Coin:

Ripple



Deposit address:

rCoinaUERUrXb1aA7dJu8qRcmvPNiKS3d

Destination Tag (optional):


1762055488

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

Withdraw funds

Required capabilities: `has_wallets`, `withdraw_funds_from_wallets`

Coin:

Bitcoin

Recipient user:

user2

Amount:

0.1

USD 906.74

Fee (deducted from amount):

฿0.000000100

USD 0.01

Comment:

sending 0.1 Bitcoin to user2

Send

Reset form

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`

Coin:

Bitcoin

Recipient user:

user2

Amount:

0.1

USD 906.74

Fee (deducted from amount):

฿0.00000100

USD 0.01

Comment:

sending 0.1 Bitcoin to user2

Send

Reset form

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

Coin:

Litecoin Testnet



Balance: LTCT 0.07461744

USD 10.97

Use the `[wallets_balance]` shortcode to show the current user's balances.

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	฿-0.000001000	฿0.000000001
deposit	airdrop ad-5ae329b9134d0	4/27/2018, 7:47:08 PM	฿1.000000000	฿0.000000000
transfer	send wallets-faucet payout	4/26/2018, 4:46:27 PM	฿-0.000000100	฿0.000000000
transfer	send wallets-faucet payout	4/26/2018, 4:44:01 PM	฿-0.000000100	฿0.000000000
transfer	send move	11/21/2017, 1:18:45 PM	฿-0.00200000	฿0.000000100
transfer	send move	11/21/2017, 1:16:38 PM	฿-0.00100000	฿0.000000100

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.

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.

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, to_address ) {
    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, to_address ) {
    if ( 'string' === typeof response.result ) {
        if ( response.result == 'success' ) {
            alert( 'Successfully withdrew ' + amount + ' ' + symbol + ' to ' + 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.

Frontend UI WordPress filters

All the UI elements are implemented as knockout.js forms that talk to the provided JSON API.

The templates for these forms live in the `wp-content/plugins/wallets/includes/views` directory of your WordPress installation:

- `wp-content/plugins/wallets/includes/views/balance/default.php`
- `wp-content/plugins/wallets/includes/views/move/default.php`
- `wp-content/plugins/wallets/includes/views/deposit/default.php`
- `wp-content/plugins/wallets/includes/views/withdraw/default.php`
- `wp-content/plugins/wallets/includes/views/transactions/default.php`
- `wp-content/plugins/wallets/includes/views/account_value/default.php`

If you wish to provide translations to these view 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_amountplusfee
- wallets_ui_text_balance
- wallets_ui_text_coin
- wallets_ui_text_comment
- wallets_ui_text_confirmations
- wallets_ui_text_depositaddress
- wallets_ui_text_fee
- wallets_ui_text_feedeductedfromamount
- wallets_ui_text_from
- wallets_ui_text_me
- wallets_ui_text_page
- wallets_ui_text_recipientuser
- wallets_ui_text_resetform
- wallets_ui_text_retriesleft
- wallets_ui_text_rowsperpage
- wallets_ui_text_send
- wallets_ui_text_status
- wallets_ui_text_tags
- wallets_ui_text_time
- wallets_ui_text_to
- wallets_ui_text_txid
- wallets_ui_text_type
- 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.

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.

Mouseover text

- wallets_ui_text_copy_to_clipboard Shown when hovering mouse over copy to clipboard button.

Amounts

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.

For example, 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.

Frontend UI WordPress actions

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`

Frontend UI styling with CSS

All the UI elements have the `dashed-slug-wallets` class to help you with CSS styling:

```
.dashed-slug-wallets.move { }  
.dashed-slug-wallets.withdraw { }  
.dashed-slug-wallets.balance { }  
.dashed-slug-wallets.deposit { }
```

```
.dashed-slug-wallets.transactions { }  
.dashed-slug-wallets.account-value { }
```

Alternative knockout templates

The basic views can be rendered in a number of different ways.

If you are a developer, you can override the default forms provided to do much more than just change the CSS styling.

All the UI elements are implemented as knockout.js forms that talk to the provided JSON API.

The templates for these forms live in the `wp-content/plugins/wallets/includes/views` directory of your WordPress installation:

- `wp-content/plugins/wallets/includes/views/balance/default.php`
- `wp-content/plugins/wallets/includes/views/move/default.php`
- `wp-content/plugins/wallets/includes/views/deposit/default.php`
- `wp-content/plugins/wallets/includes/views/withdraw/default.php`
- `wp-content/plugins/wallets/includes/views/transactions/default.php`
- `wp-content/plugins/wallets/includes/views/account_value/default.php`

You can copy these files to another directory of your choosing. Make sure to override the views directory using the `wallets_views_dir` filter, to let the plugin know where to look for the templates. Here's sample code for how to do this:

```
function my_wallets_views_dir_override( $dir ) {  
    return '/path/to/the/directory/where/the/new/templates/are/copied';  
}  
add_filter( 'wallets_views_dir', 'my_wallets_views_dir_override' );
```

As of version 3.2.0 you can also specify the directory via a shortcode attribute.







```
[wallets_balance views_dir="/path/to/the/directory/where/the/new/templates/are/copied"]
```

The above attribute takes precedence over any directory specified via the `wallets_views_dir` filter. In this case the template rendered would be the file `/path/to/the/directory/where/the/new/templates/are/co`

Finally, you can specify a different template other than the default one. Here's how you would display deposit addresses as a list:

```
[wallets_deposit template="list"]
```

This would render the `wp-content/plugins/wallets/includes/views/deposit/list.php` template, rather than the default one (`default.php`):

Coin	Deposit address
 Ripple (XRP)	rCoinaUERUrXb1aA7dJu8qRc 1762055488
 Bitcoin Cash (BCH)	mtzHegdEJSQK8PaqMRu3kF
 Ether (ETH)	0x17ca27ad0e435a04b6a4c
 Litecoin Testnet (LTCT)	mqqumC3xUV3NadEezk3KT
 Litecoin (LTC)	LLycx1sbsuPVak61V9SoUYE
 Bitcoin (BTC)	muX7EfQ1YKcsfowYRdmSsH

You can combine the `views_dir` and `template` attributes to provide a library of your own shortcode templates anywhere on your filesystem.

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_withdraw()`, `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`.

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.)
- **###AMOUNT###** — The amount transacted.
- **###FEE###** — For withdrawals and transfers, the fees paid to the site.
- **###SYMBOL###** — The coin symbol for this transaction (e.g. “BTC” for Bitcoin)
- **###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*.

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:

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

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

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.

This release features an admin page that makes it easy to split funds into a live and cold wallet: For every currency you have installed, the page will tell you

- how many coins you have in your online wallet,
- how much is the total of all your users' balances, and
- how much you need to deposit or withdraw to reach your target level.

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!

The modular philosophy of the Bitcoin and Altcoin Wallets plugin is about **providing program-matic** access to an **abstraction layer** that can then talk to a **bitcoin** wallet or to other cryptocurrencies.

In simple English: You and others can build and use applications that do financial transactions, without worrying about the implementation details of each coin, or about UI issues, accounting, etc.

For site administrators

- Install wallet daemons
- Install this plugin and configure it to use your wallets.
- Insert the simple shortcodes into your site's content.

For developers

If you're a developer, by now you will probably be asking:

Can I see the software stack diagram?

OK, here it is:

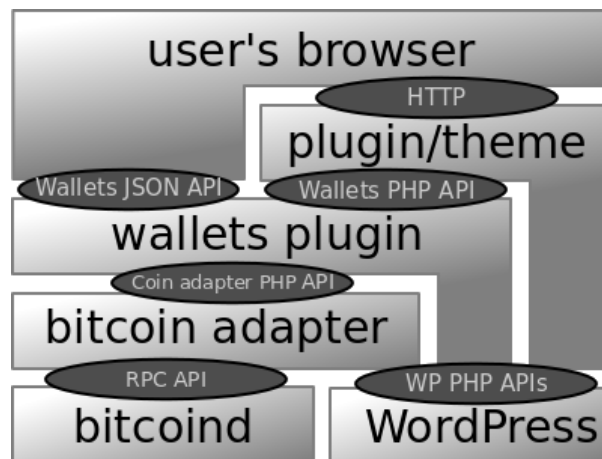


Figure 1: software stack

Wallets JSON API

The Bitcoin and Altcoin Wallets WordPress plugin exposes two JSON APIs:

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 2**.

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/api2/get_coins_info` **Ugly URI path:** `?__wallets_action=get_coins_info&__wallets`

Parameters: none

Required capabilities: `has_wallets`

Example response:

```
{
  "coins":{
    "LTC":{
      "name":"Litecoin",
      "symbol":"LTC",
      "icon_url":"http://example.com/wp-content/plugins/wallets-litecoin/assets/icon.png",
      "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",
        "deposit_address_qrcode_uri":"litecoin:LPsj8nDiuBjbvRFR8CDoYhSbd4nDekbBQV"
    },
    "BTC":{
        "name":"Bitcoin",
        "symbol":"BTC",
        "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",
        "deposit_address_qrcode_uri":"bitcoin: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/api2/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": "fed08f9a90c526f2bb791059a8718d422b8fdbcb55f719bd36b6e3d9717e815e0",
        "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` and `do_withdraw` 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.

Pretty URI path: `/wallets/api2/get_nonces` **Ugly URI path:** `?__wallets_action=get_nonces&__wallets_apiver=2`

Parameters: none

Required capabilities: `has_wallets`

Example response:

```

{
    "nonces": {
        "do_withdraw": "893a9f9dad",
        "do_move": "6d5ebff18a"
    },
    "result": "success"
}

```

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=2&__wallets_symbol=SYMBOL&__wallets_toaccount=TOACCOUNT&__wallets_amount=AMOUNT&__wallets_comment=COMMENT&__wallets_wpnonce=WPNONCE`

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=2&__wallets_symbol=SYMBOL&__wallets_address=ADDRESS&__wallets_amount=AMOUNT&__wallets_comment=COMMENT&__wallets_comment_to=COMMENT_TO&__wallets_wpnonce=WPNONCE`

Parameters:

- **SYMBOL:** The symbol of the coins to move, e.g. BTC, LTC, etc.
- **ADDRESS:** The external address to send coins to.
- **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.
- **COMMENT_TO:** A descriptive string that the sender attaches to the address.
- **WPNONCE:** The `do_move` WordPress nonce from the `get_nonces` call.

Required capabilities: `has_wallets, withdraw_funds_from_wallet`

Example response:

```
{
  "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/api2/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/api2/notify/SYMBOL/TYPE/MESSAGE` **Ugly URI path:**

`?__wallets_action=notify&__wallets_symbol=SYMBOL&__wallets_notify_type=TYPE&__wallets_notify_message=MESSAGE`

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"
}
```

Accessing the JSON API from cURL

The JSON API is available to logged in users only. Here's how to access the API programmatically.

One must first perform a login to the WordPress site:

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

Substitute `www.example.com` with your domain, and `USER` and `PASSWORD` with your user credentials.

Then, perform a request to the JSON API, making sure to pass the same cookies that were returned after successful login:

```
curl -b cookie.txt -c cookie.txt \
http://www.example.com/wallets/api2/get_coins_info
```

That is all.

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 utilize 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 1. the coin adapter is enabled, and 2. the coin is listed on the selected exchange

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*.

Pluggable exchanges

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.

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://bitcoinaucet.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 am getting a warning about cron “The wp_cron job has not run in a while and might be disabled.”

You are seeing this error message either because your site has low traffic or because your cron job is not running for some reason. If the cause is low traffic then you can safely ignore this message. If something else is wrong and no transactions are being executed, read on:

1. Manually visit the following link in your browser, replacing *example.com* with the domain name for your site. This should trigger cron once and execute a single batch of transactions.

`http://example.com/wp-cron.php?doing_wp_cron`

2. Setup a unix cron job to trigger that URL via curl at regular intervals. For example this will trigger cron every five minutes:

```
* /5 * * * * curl http://example.com/wp-cron.php?doing_wp_cron  
> /dev/null 2>&1
```

3. Alternatively, you can setup the above curl command in CPanel if you prefer. See here for details.

I am getting a warning about cron “WordPress cron is disabled.”

1. Check if the constant `DISABLE_WP_CRON` is set to false in your `wp-config.php` file. If it is, delete the line and cron should be enabled again.
2. If you cannot find the constant in `wp-config.php`, some other plugin might be setting the constant. Search the code of the other plugins or try disabling them to figure out which plugin is causing this.

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 users 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 `rpccallowip` 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 getinfo`. 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 you have enabled logging in `wp-config.php` (`WP_DEBUG` is `true`), and you have not set `WP_DEBUG_DISPLAY` to `false`) and 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`.

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.

When I activate the plugin, my WordPress becomes extremely slow and is unusable.

If WordPress has become so slow as to be completely unusable, you can regain control of your WordPress if you simply delete the plugin.

- You will have to delete the `wp-content/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 you 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.

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 withdrawal
        '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 overriden
        'explorer tx uri' => 'https://live.blockcypher.com/ltc/tx/%s/',

        // Default sprintf() pattern for URI to block explorer address page. Can be overriden
        '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 'assets'
        '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

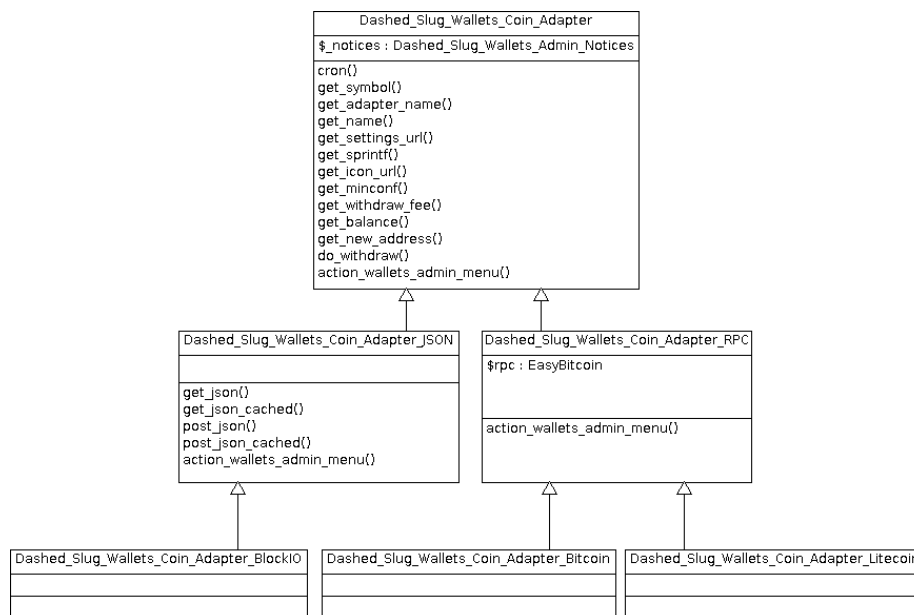


Figure 2: Coin adapters class hierarchy

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.

Internal transfer — A transfer between two users in the same WordPress installation, either in the same blog or across 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.

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.

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.

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.

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*.

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.