Bitcoin and Altcoin Wallets 2.2.2 reference manual

dashed-slug info@dashed-slug.net

Abstract

This is the reference manual for the **Bitcoin and Altcoin Wallets** plugin, version **2.2.2** for WordPress.

Introduction

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

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.

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 on your site if you have not already done so

APIs, APIs everywhere!

The modular philosophy of the Bitcoin and Altcoin Wallets plugin is about **providing programmatic** 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:

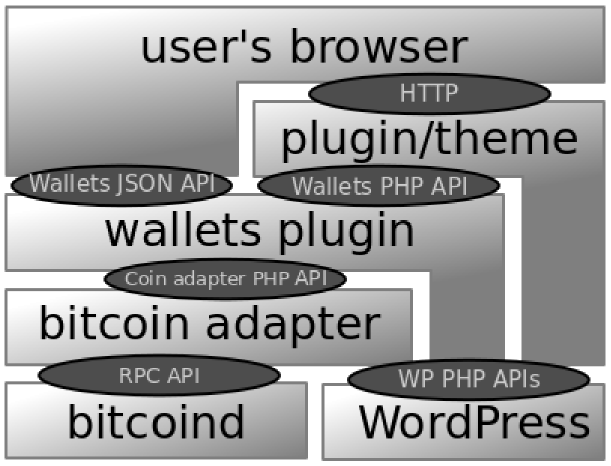


Figure 1: software stack

Features overviw

Deposit crypto

Your users can now deposit and withdraw Bitcoins.

Fees

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

Altcoins

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

User transactions

Users can transfer coins to other users.

Front-end UI

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

Notifications

Users are notified by email whenever they perform transactions. Or bind to the relevant actions and code your own type of notification in PHP.

User accounts

Accounting is taken care of internally.

DB storage

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

Easy backups

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

Pluggable architecture

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!

Capabilities-based access control

Account roles are assigned capabilities which limit the available operations.

Documentation

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

FAQ

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.

Do I really need to run a full node? bitcoind is too resource-hungry for my server.

Running a full node requires the full blockchain to be downloaded.

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 `bitcoinj` and does not store the blockchain locally.

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.

Soon there will be a coin adapter based on the online `blockchain.info` service. This will be another option for running a Bitcoin wallet without the overhead of a local copy of the blockchain. Stay tuned.

How can I integrate it 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.

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

First of all, the forms can be styled with CSS. They have convenient HTML classes that you can use.

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 shortcodes documentation 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.

How can I get support or submit feedback?

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

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

Shortcodes

Wallet shortcodes display forms to the front-end that let the user do basic operations with their cryptocurrency accounts.

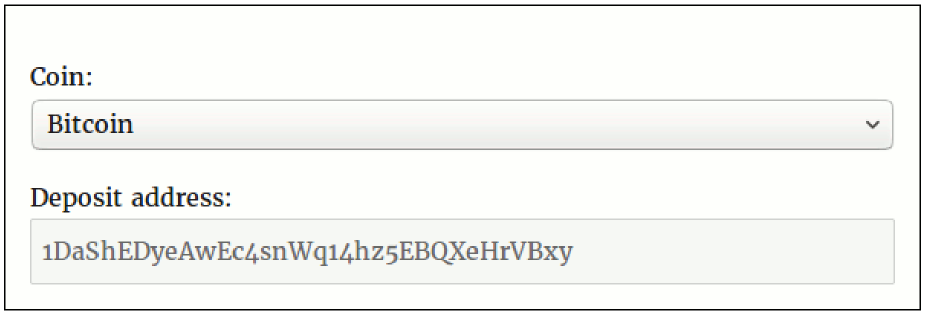
All the operations accessible via shortcodes are only relevant to logged in users. The short-codes will only display if:

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

Shortcode reference

Deposit funds

Required capabilities: `has_wallets`



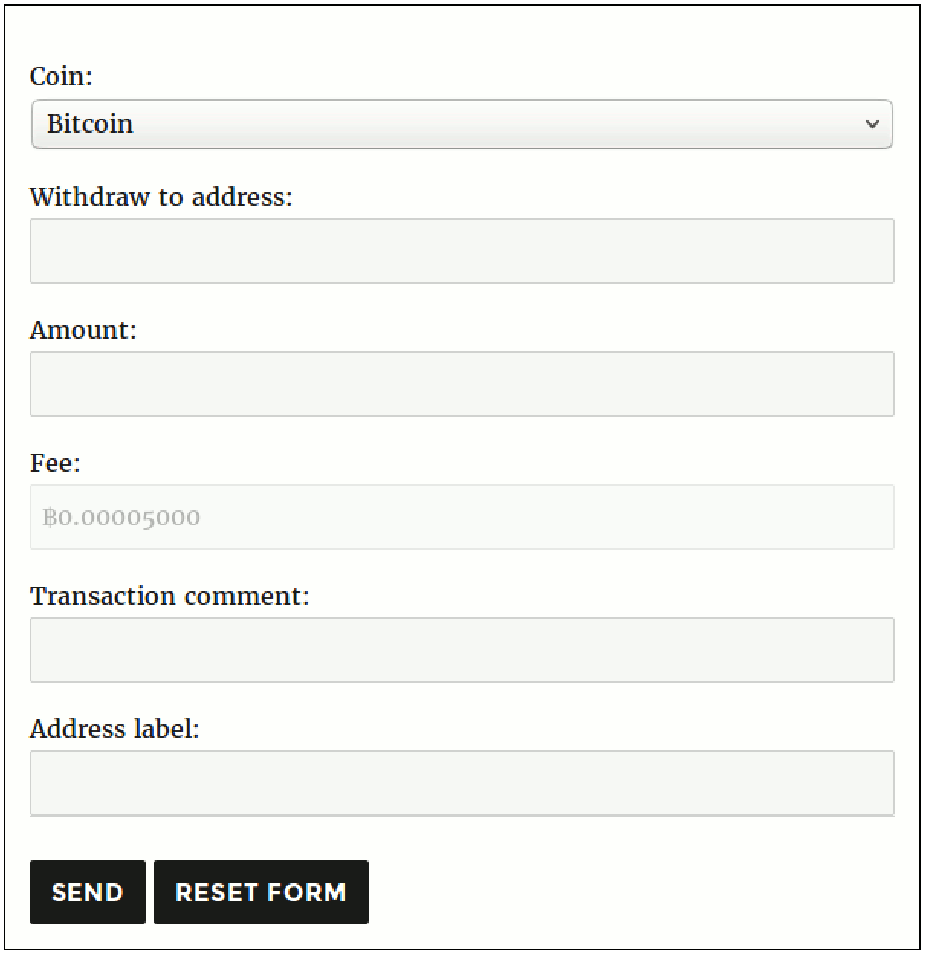
Coin:
Bitcoin

Deposit address:
1DaShEDyeAwEc4snWq14hz5EBQXeHrVBxy

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

Withdraw to address:

Amount:

Fee:

฿0.00005000

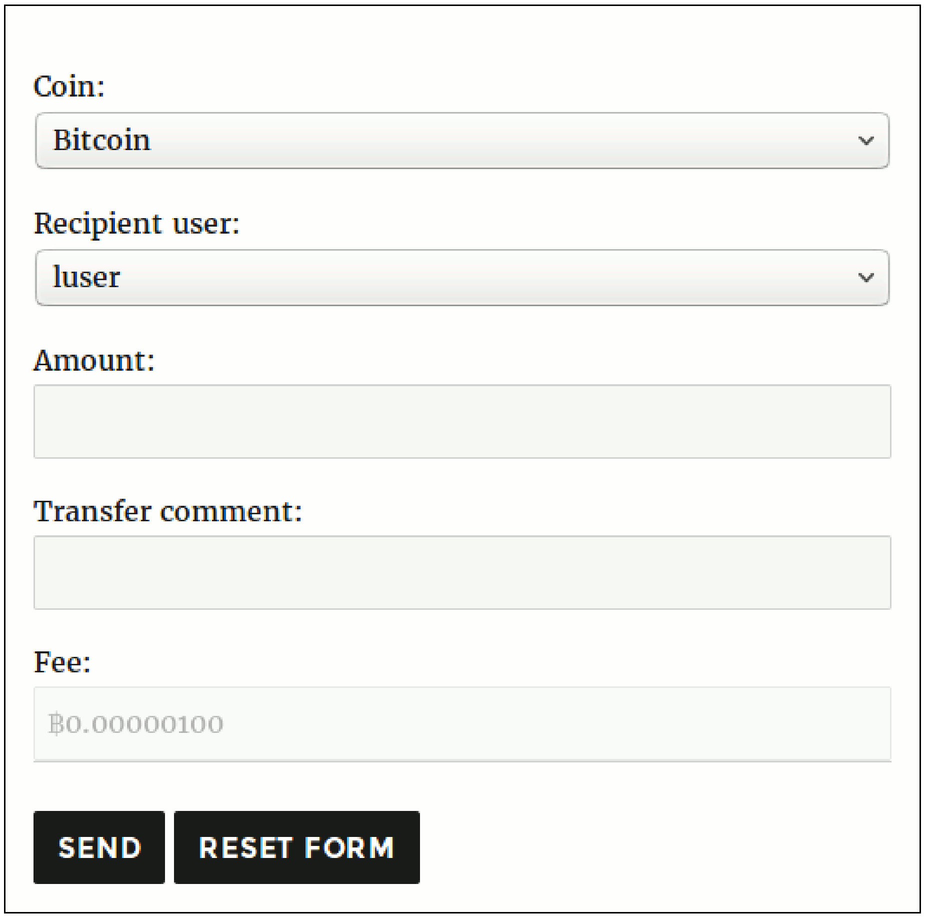
Transaction comment:

Address label:

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

Transfer funds to other users

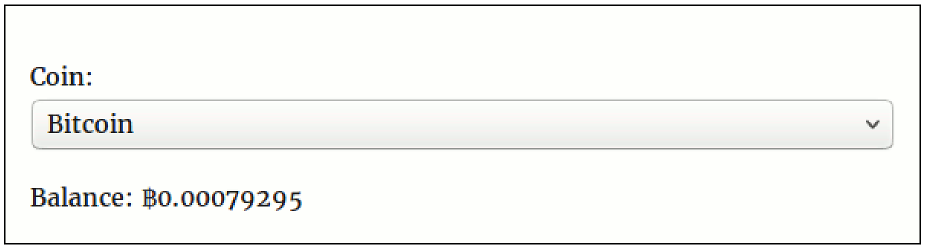
Required capabilities: `has_wallets`, `send_funds_to_user`



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

Display user balance(s)

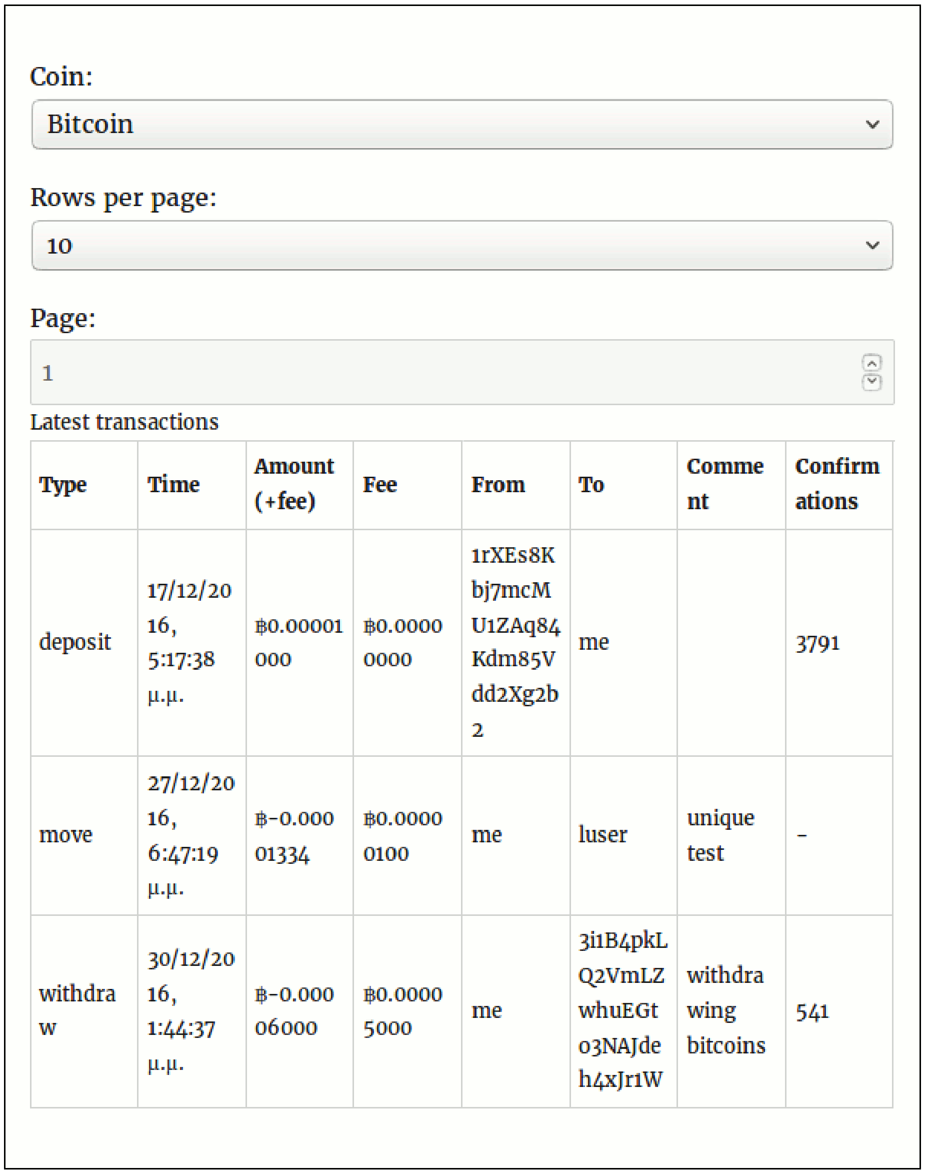
Required capabilities: `has_wallets`



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

View past transactions

Required capabilities: `has_wallets`, `list_wallet_transactions`



Coin:

Bitcoin

Rows per page:

10

Page:

1

Latest transactions

Type	Time	Amount (+fee)	Fee	From	To	Comment	Confirmations
deposit	17/12/2016, 5:17:38 μ.μ.	฿0.00001000	฿0.00000000	1rXEs8Kbj7mcMU1ZAq84Kdm85Vdd2Xg2b2	me		3791
move	27/12/2016, 6:47:19 μ.μ.	฿-0.00001334	฿0.00000100	me	luser	unique test	-
withdraw	30/12/2016, 1:44:37 μ.μ.	฿-0.00006000	฿0.00005000	me	3i1B4pkLQ2VmLZwhuEGto3NAJdeh4xJr1W	withdrawing bitcoins	541

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.

Capturing events on the front-side

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 event names are:

- `wallets_do_withdraw`, and
- `wallets_do_move`.

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:

Observing transfers between users:

```
jQuery('body').on( 'wallets_do_move', function( event, response, symbol, amount, toaccount, co
    if ( typeof(response.result) == 'string') {
        if ( response.result == 'success' ) {
            alert( 'Successfully sent ' + amount + ' ' + symbol );
        } else {
            alert( "Move failed: \n" + response.message );
        }
    } else {
        alert( "Move failed!" );
    }
    event.preventDefault();
});
```

Observing withdrawals:

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

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.

Overriding the default UI

Using 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 { }
```

Using HTML and PHP and JavaScript and CSS and stuff!

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

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

These forms live in the `wp-content/plugins/wallets/includes/views` directory of your WordPress installation:

```
wp-content/plugins/wallets/includes/views/balance.php
wp-content/plugins/wallets/includes/views/move.php
wp-content/plugins/wallets/includes/views/deposit.php
wp-content/plugins/wallets/includes/views/withdraw.php
wp-content/plugins/wallets/includes/views/transactions.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' );
```

E-mail notification settings

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

- You can control which 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
- `###OTHER_ACCOUNT###` — Username 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###` — The date and time of the transaction in ISO-8601 notation. YYYY-MM-DDThh:mm:ssZZZZ
- `###COMMENT###` — For internal fund transfers, the comment attached to the transaction.
- `###ADDRESS###` — For deposits and withdrawals, the external address.

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`).

Transaction API

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

Get users info

Retrieves usernames and user IDs of all users on the site.

Used by the `[wallets_move]` shortcode UI to display a drop-down selection of users to send coins to.

URI path: `/wallets/get_users_info`

Parameters: none

Required capabilities: `has_wallets`, `send_funds_to_user`

Example response:

```
{
   "users":[
      {
         "id":2,
         "name":"luser"
      }
   ],
   "result":"success"
}
```

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.

URI path: `/wallets/get_coins_info`

Parameters: none

Required capabilities: `has_wallets`

Example response:

```
{
   "coins":{
      "LTC":{
         "name":"Litecoin",
         "symbol":"LTC",
      "icon_url":"http:\/\/www.turbox.lan:81\/wp-content\/plugins\/wallets-litecoin\/assets\
         "balance":0.00659988,
         "balance_string":"\u01410.00659988",
         "unconfirmed_balance":0.00659988,
         "unconfirmed_balance_string":"\u01410.00659988",
         "deposit_address":"LPsj8nDiuBjbvRFR8CDoYhSbd4nDekbBQV"
      },
      "BTC":{
         "name":"Bitcoin",
         "symbol":"BTC",
      "icon_url":"http:\/\/www.turbox.lan:81\/wp-content\/plugins\/wallets\/includes\/..\/as
         "balance":0.00091565,
         "balance_string":"\u0e3f0.00091565",
         "unconfirmed_balance":0.00091565,
         "unconfirmed_balance_string":"\u0e3f0.00091565",
         "deposit_address":"mrXEs8Kbj7mcMU1ZAq84Kdm85Vdd2Xg2b2"
      }
   },
   "result":"success"
}
```

Get transactions

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

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

URI path: `/wallets/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",
         "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",
         "other_account_name":null,
         "amount_string":"\u0e3f0.00001000",
         "fee_string":"\u0e3f0.00000000"
      },
      {
         "category":"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",
         "other_account_name":"luser",
         "amount_string":"\u0e3f-0.00001334",
         "fee_string":"\u0e3f0.00000100"
      },
      {
         "category":"withdraw",
         "account":"1",
         "other_account":null,
         "address":"1i1B4pkLQ2VmLZwhuEGto3NAJdeh4xJr1W",
      "txid":"fed08f9a90c526f2bb791059a8718d422b8fdcb55f719bd36b6e3d9717e815e0",
         "symbol":"BTC",
         "amount":"-0.0000600000",
         "fee":"0.0000500000",
         "comment":"withdrawing bitcoins",
         "created_time":"2016-12-30 11:44:37",
         "updated_time":"2016-12-30 12:46:41",
         "confirmations":"12",
         "other_account_name":null,
         "amount_string":"\u0e3f-0.00006000",
         "fee_string":"\u0e3f0.00005000"
      }
   ],
   "result":"success"
}
```

All times are GMT.

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.

URI path: `/?__wallets_action=do_move&__wallets_symbol=SYMBOL&__wallets_move_toaccount=TOACCOUN`

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.

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.

URI path: `/?__wallets_action=do_withdraw&__wallets_symbol=SYMBOL&__wallets_withdraw_address=AD`

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.

Required capabilities: `has_wallets`, `widthdraw_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/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.

URI path: `/wallets/notify/SYMBOL/TYPE/MESSAGE`

Parameters:

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

Example response:

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

PHP API

The Bitcoin and Altcoin Wallets WordPress plugin offers a PHP API.

Purpose

- You can access the API from your theme to perform transactions on behalf of your users.
- You can develop plugins that utlise the PHP API.
- You can install plugins that the Dashed-Slug releases to extend functionality. Many such plugins are on the roadmap.

How to call

First you must call `Dashed_Slug_Wallets::get_instance()` to get an instance of the wallets plugin class. Refer to the auto-generated PHPdoc for a complete reference of what you can do and access.

Security caution

As a side note, this also means that you must double-check what themes and plugins are installed on your site. Every piece of code that your site runs must be trustworthy for security reasons. Do not install themes or plugins from unknown sources without checking the code thoroughly. You should be already aware of this.

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`.

manage_wallets

- Enables admin interface. For administrators only.

has_wallets

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

list_wallet_transactions

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

send_funds_to_user

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

withdraw_funds_from_wallet

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

Adapters

In *Bitcoin and Altcoin Wallets*, connectivity to the various local and cloud wallets is provided by *Coin adapters*.

Coin adapters:

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.

Coin adapter simple example

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 wallet 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

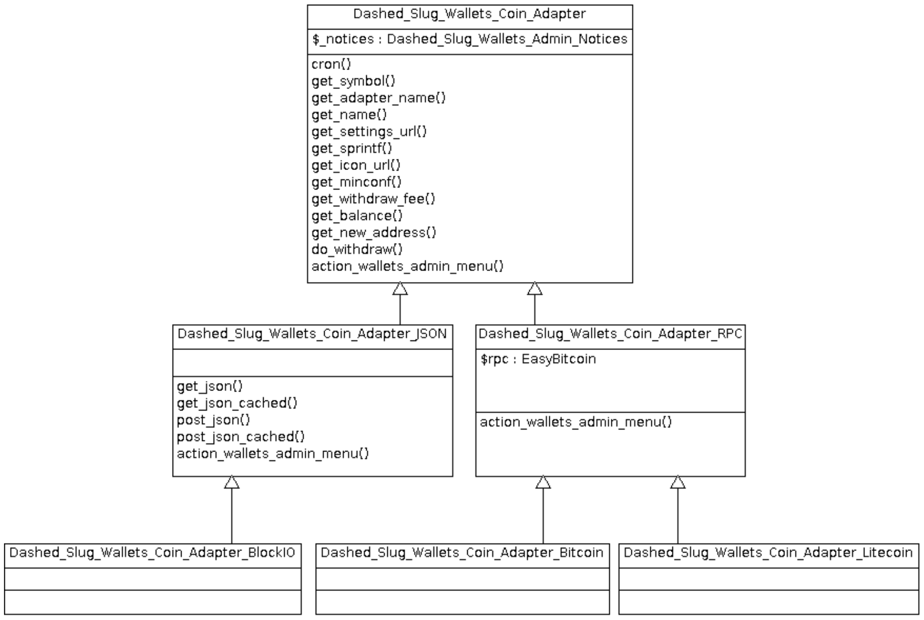

Dashed_Slug_Wallets_Coin_Adapter
$_notices : Dashed_Slug_Wallets_Admin_Notices
cron()
get_symbol()
get_adapter_name()
get_name()
get_settings_url()
get_sprintf()
get_icon_url()
get_minconf()
get_withdraw_fee()
get_balance()
get_new_address()
do_withdraw()
action_wallets_admin_menu()
Dashed_Slug_Wallets_Coin_Adapter_JSON
get_json()
get_json_cached()
post_json()
post_json_cached()
action_wallets_admin_menu()
Dashed_Slug_Wallets_Coin_Adapter_RPC
$rpc : EasyBitcoin
action_wallets_admin_menu()
Dashed_Slug_Wallets_Coin_Adapter_BlockIO
Dashed_Slug_Wallets_Coin_Adapter_Bitcoin
Dashed_Slug_Wallets_Coin_Adapter_Litecoin

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()`.

Download

This plugin is always available at the dashed-slug download area.

Contact and support

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

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