
Beam Documentation

Release December 2018

Beam Team

Jun 04, 2019

1	Announcement	3
2	What is Beam?	5
3	Getting Started	7
4	Important differences from other cryptocurrencies	9
5	Reporting Issues and Getting Support	11
5.1	Desktop Wallet User Guide	11
5.2	Desktop Wallet troubleshooting	59
5.3	Command Line Wallet User Guide	60
5.4	Beam Node User Guide	65
5.5	Backup and Restore	68
5.6	Mining Beam	69
5.7	Blockchain Explorer	77
5.8	Supported Platforms	77
5.9	Files and Locations	78
5.10	Reporting Issues and Getting Support	79
5.11	Troubleshooting	79
5.12	Resources	80
5.13	Glossary	80
5.14	Building Beam	81
5.15	Understanding Beam logs	83
5.16	Consensus Rules	84
5.17	Local Setup	87



CHAPTER 1

Announcement

Note: Beam is currently in Mainnet.

Rules Signature: [ed91a717313c6eb0](#)

Download binaries from: [Beam Downloads Page](#)

Source code: [Beam Github](#)

TLDR;

I want to Mine Beam

I want to learn how to use Beam Desktop Wallet

I want to learn how to use Beam Command Line Wallet

What is Beam?

Beam is a next generation scalable, confidential cryptocurrency based on an elegant and innovative [Mimblewimble protocol](#).

Things that make BEAM special include:

- Users have complete control over privacy - a user decides which information will be available and to which parties, having complete control over his personal data in accordance to his will and applicable laws.
- Confidentiality without penalty - in BEAM confidential transactions do not cause bloating of the blockchain, avoiding excessive computational overhead or penalty on performance or scalability while completely concealing the transaction value.
- No trusted setup required.
- Blocks are mined using Equihash Proof-of-Work algorithm.
- Limited emission using periodic halving.
- No addresses are stored in the blockchain - no information whatsoever about either the sender or the receiver of a transaction is stored in the blockchain.
- Superior scalability through compact blockchain size using the “cut-through” feature of Mimblewimble.
- No premine. No ICO. Backed by a treasury, emitted from every block during the first five years.
- Implemented from scratch in C++.

CHAPTER 3

Getting Started

The simplest way to get started with Beam is by visiting the [Beam website](#), reading and understanding the materials posted there and joining Beam Community on Telegram (<https://t.me/BeamPrivacy>) for updates and discussions.

Danger: Beam is extremely new and experimental technology. No guarantees can be provided by anyone whatsoever. Use it at your own risk. Make sure you know what you are doing, especially if there is money involved.

Just like any other cryptocurrency, using Beam requires learning and understanding what this all is about. If real money is involved, it also requires concern with security of the process.

Hint: That said, you can always safely play with Beam by connecting to the permanent Testnet.

To learn more about how cryptocurrencies work in general and Beam in particular please visit our [Resources](#) page

Once you familiarized yourself with key ideas and concepts, it is recommended to start from connecting to our Testnet. The simplest way to do that is by downloading and installing our Desktop Wallet and following instructions in [Desktop Wallet User Guide](#).

Important differences from other cryptocurrencies

Mimblewimble has several important differences from most other existing cryptocurrencies which are very important to understand. Please review the following information carefully.

The concept of *Address*** is completely different**

In most cryptocurrencies Address is a hashed public key for which the owner of that Address knows the corresponding private key. In order to transfer funds, the Sender should only know the Address of the Receiver in order to create a unilateral transaction. *The Sender is not aware of whether the Receiver is online or not or whether it even exists.* Once transaction to an Address is complete and added to the blockchain, Receiver that can prove knowledge of the private key corresponding the Address can control this UTXO (short for Unspent Transaction Output).

In Mimblewimble there are no addresses at all and transaction are created **interactively** by both Sender and Receiver wallets. This means that in order to create a transaction, both wallets have to participate in the creation process and eventually co-sign the transaction before it is sent to the blockchain.

<p>Attention: In Beam it is not possible to create a transaction unilaterally. Both Sender and Receiver have participate in transaction creation.</p>
--

To allow Sender and Receiver wallets to create transactions without having to be online at the same time and directly connected to each other, Beam added a module called *SBBS* that allows wallets to securely communicate using encrypted messages to create a transaction. SBBS Addresses are merely private / public key pairs used to encrypt and decrypt these messages.

Important: SBBS Addresses are not recorded in the blockchain and are not used to control funds

You are encouraged to create a new SBBS Address for each transaction.

Wallet and Node concepts are slightly different

Beam documentation mentions terms Wallet and Node quite a lot and it sometimes causes confusion with users of other cryptocurrencies.

Beam Wallet is a *light client* which stores information about UTXO that belong to it and has an ability to create new transactions by connecting to other wallets via *SBBS*. It does not store or verify the entire blockchain and can thus only work if connected to a Node.

Beam Node, is a *full node* that downloads, validates and updates the entire blockchain state.

Note: Beam Desktop Wallet, provides options to run both as just the Wallet (connected to a remote node) and as a full node.

<p>Attention: It is always recommended to run a full node</p>
--

Information that can be restored from the blockchain is completely different

In most blockchains, information about current UTXOs and the transaction history can be recovered from the blockchain using only the *Seed Phrase*.

In Beam, only UTXOs can be recovered from the blockchain. All other information, including transaction history and any other meta data are only stored locally in the Beam Wallet database and encrypted by *Wallet Password*.

This means that if you run Beam Wallet on two different machines, transaction history **WILL NOT** be synchronized.

This also means, that to preserve transaction history, or any additional meta data, it is necessary to regularly backup Beam Wallet database file.

For more information about backup and restore procedure see *Backup and Restore*

Reporting Issues and Getting Support

To report issues and get support please perform the following steps:

1. Gather all relevant information including:
 - Detailed description of the problem you have encountered and steps to reproduce it
 - Version of the binaries you are running
 - Logs (see :ref: *log locations* for information where to find the logs files)
 - Relevant configuration files (please check for private information before sending)
 - Your system configuration
 - Screen shots or any additional information you think is relevant
2. Send an email to support@beam.mw (or testnet@beam.mw for testnet related issues).

You can also open an issue in github and follow the provided template.

Attention: Providing all the information described above will allow us to quickly and efficiently analyze and resolve the issue for you and everyone else.

Warning: The following document is still under construction and is subject to changes.

5.1 Desktop Wallet User Guide

Using Beam Desktop Wallet is the simplest way to start using Beam. It is available for Linux, Mac and Windows platforms (see *Supported Platforms* for details).

Attention: Beam blockchain does not store transaction history and SBBS addresses. These are only stored in local database inside the wallet data folder.

Please follow the guidelines below to avoid problems with sending or receiving Beam transactions.

1. Do not copy the wallet.db to another machine and run another wallet simultaneously using the same wallet database
2. Do not run two different wallets with the same seed at the same time
3. SBBS messages sent between wallets expire after 12 hours. You have to connect within 12 hours of the transaction initiation to receive or send the funds.
4. SBBS Addresses by default expire after 24 hours. Always use ‘never’ expiring addresses with pools and exchanges to make sure you receive payments.

5.1.1 Downloading binaries

Start downloading here: <https://beam.mw/downloads>

Ensure that the communication between your browser and Beam’s official website is encrypted by verifying that the padlock icon is displayed in its locked state on the URL bar.



Download the Beam wallet app for your platform (Mac, Linux or Windows):

	Apple	Linux	Windows
Wallet desktop app (includes CPU miner node)	Beam-Wallet-1.0.3976.dmg SHA256: 8ee0e04d0f01c91e9662bd9aa5f3fa59082a1d9096a9efcb7944b70d0bc76d	Beam-Wallet-1.0.3976.deb SHA256: ac5fee08d0a21e3d3d5a8366374be969ceea66bb685a224385fb44dc	Beam-Wallet-1.0.3976.exe SHA256: 5342b41c21131019e170d557caaf643aa0c0c5dbccca49160ea0462639092
[BETA] Wallet desktop app (includes CPU/GPU miner node)	N/A	Beam-Wallet-1.0.4028.deb SHA256: 40c11f0d4dc8eb54cc0e999986fee4d530d2eaa98a644cbca65de0d7fcd26a	Beam-Wallet-1.0.4028.exe SHA256: b0d648e8a6052c7d70bb98aceeb78208369cea31a263cb83dfcc38ad6d
CLI wallet	beam-wallet-cli-1.0.3976.tar.gz SHA256: 62b0ef63639c97878a602bd5842d1b26089eac4c8260970d7591acc0508148	beam-wallet-cli-1.0.3976.tar.gz SHA256: b0be699584b4e93eaa3d017fa0d5523d7087abb0973f0b584c56bffa1b2a	beam-wallet-cli-1.0.3976.zip SHA256: 82c0643043942e7932161444b3bae52f9dc79340ca3c5e45600471e804
Node	beam-node-1.0.3976.tar.gz SHA256: 80b088e6f98b08099c119d1ed3cd9765b0677dabcb5e65b47dfe37447c42	beam-node-1.0.3976.tar.gz SHA256: d9bb7006677d8862d34a40501d267eeb1c3ae4399075de4a9c033ac3b77e1	beam-node-1.0.3976.zip SHA256: 5d4e42c175fcd647ac16130a1cb51410a66d7aacc40e46d50fee3fde3d72cb7
Miner (OpenCL)	N/A	openc1-miner-1.0.52.tar.gz SHA256: 654fb4a40be294328727ce6aded9b240400081e9e94059c9889613a5de13e6	openc1-miner-1.0.52.zip SHA256: d539b15447b7d7a8b0c1388a9e4631cb6d06bc072042579a442a48deb62c5
Miner (CUDA)	N/A	cuda-miner.tar.gz SHA256: 7e7806d0d0016362c07b57b86b71813c95f788e51ada187515ba0f6b6cfd55	cuda-miner.zip SHA256: 7eb47bc266f9ae853ee77e82b51920abc1e4c405b891982488f6f9974

- On Mac open terminal and run: `shasum -a 256 <file>`

- On Linux run: `sha256sum <file>`
- On Windows, open Power Shell and run: `Get-FileHash 'C:\Users\\Downloads\beam-node-0.3860.zip'`

Substitute your own path instead of the one in the example above.

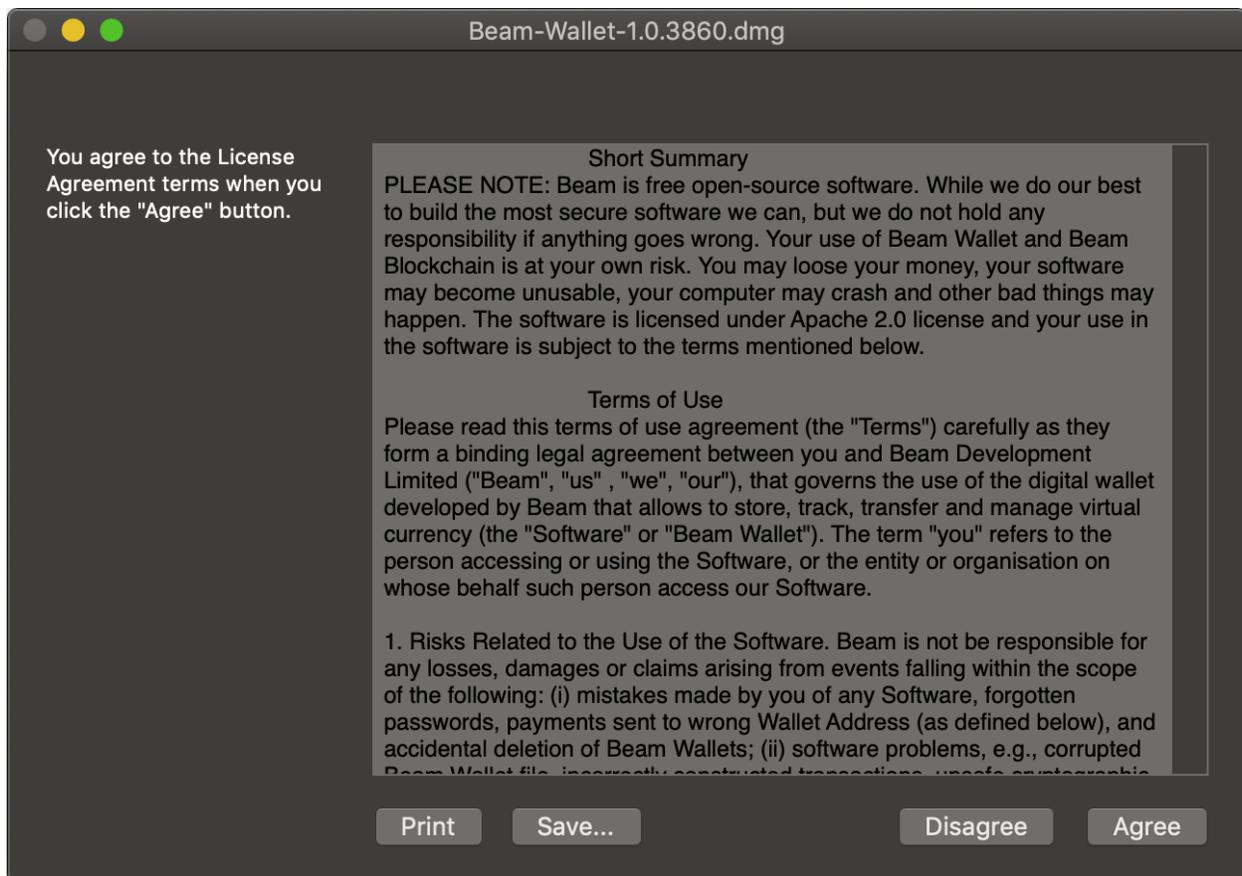
Note: As the wallet will be continuously updated, the actual version numbers and SHA values might be different from the screenshot at the moment of reading.

5.1.2 Installing the desktop app

Once the application image is downloaded, double-click the icon to start the installation.

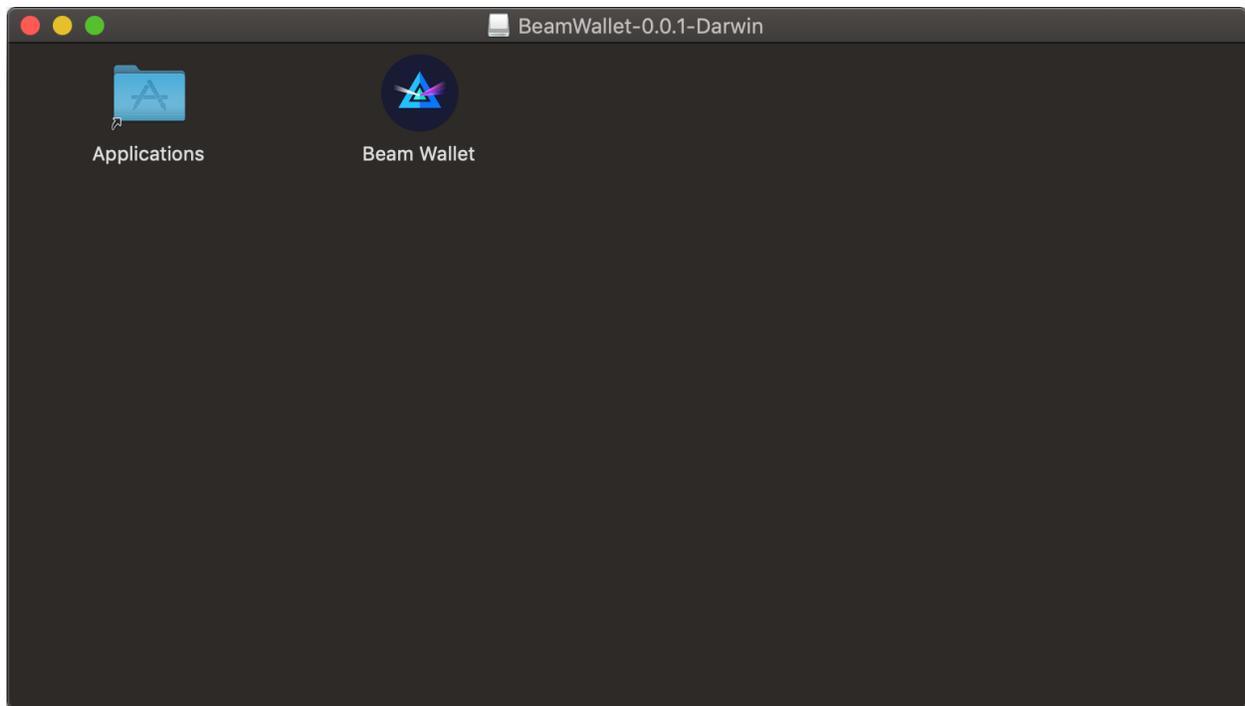
5.1.3 Mac

When you click on the .dmg file you will see a screen with Disclaimer and End User License Agreement. Please read carefully and click 'Agree'.



On the next screen, drag the Beam Wallet icon into the Applications folder to install.

When you will try to open the wallet for the first time, you will receive a security warning stating that Beam Wallet was created by an unrecognized developer. We are working to eliminate the warning, meanwhile follow the steps below to launch Beam Wallet app on your Mac.



Open 'System Preferences'.

Locate and click 'Security & Privacy' settings.

Click the lock icon in the bottom left corner of the dialog to unlock.

Change the 'Allow apps downloaded from' setting from 'App Store' to 'App Store and identified developers'. Click the lock icon again when done making changes.

Now, launch the Beam Wallet from the Applications folder. When notification appears, click 'Open'.

5.1.4 Where are the files?

Once Beam Wallet desktop app is installed, the wallet data files are stored separately from the binaries.

The locations of all the files are described here: [Files and Locations](#)

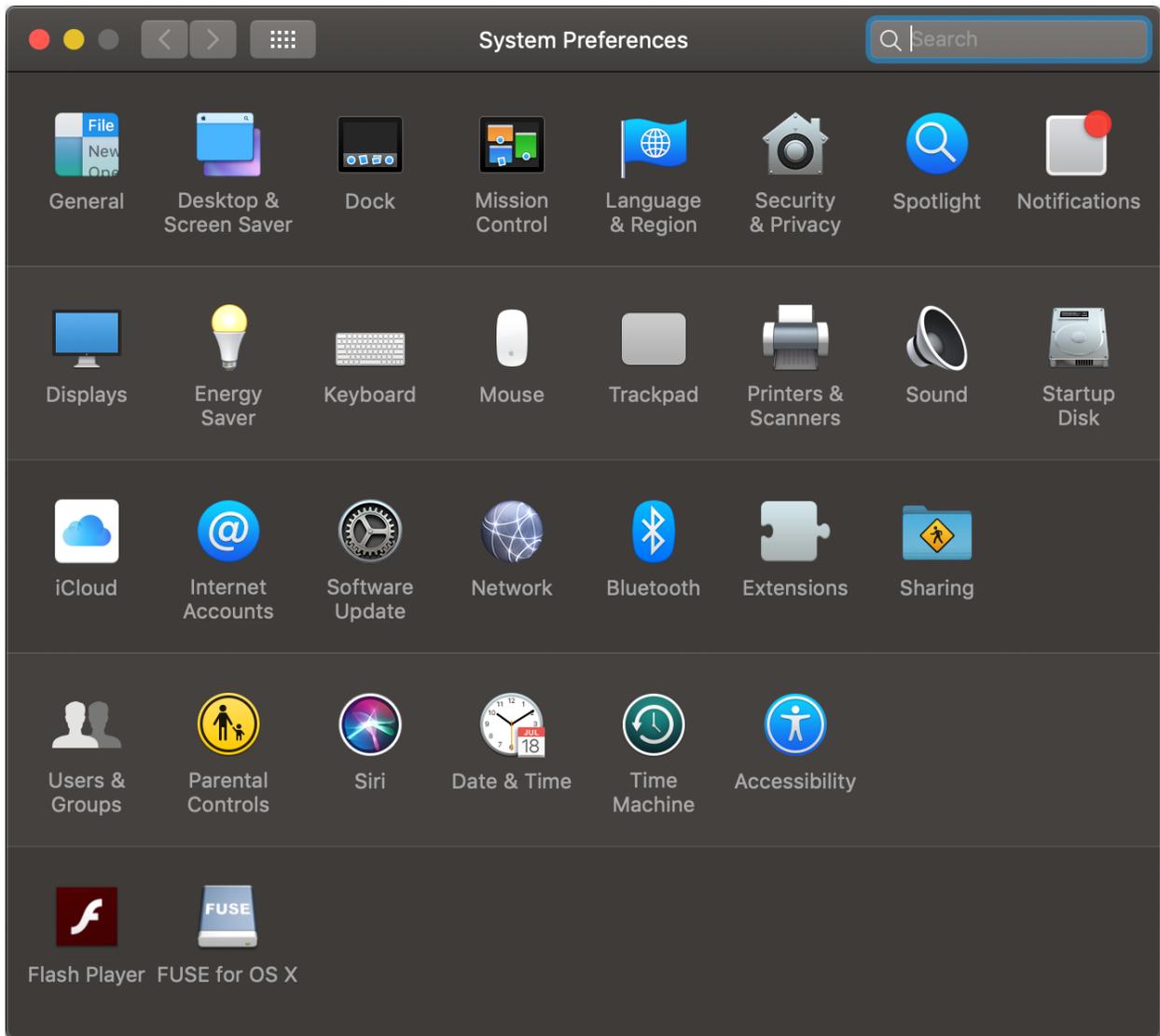
..note::When the Beam wallet app is running, right click on it and select *Options | Keep in dock* for easy access of the app in the future.

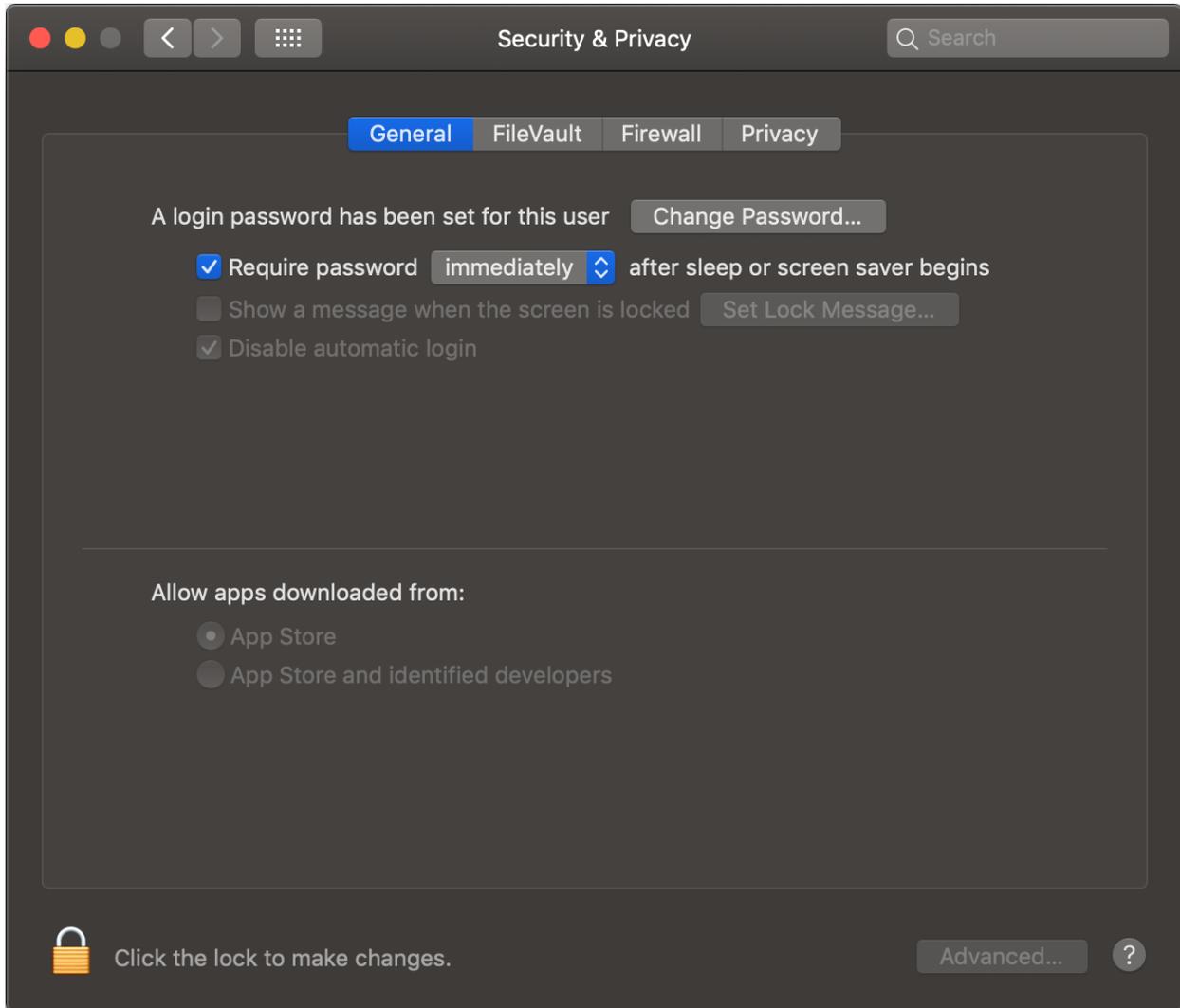
5.1.5 Creating new wallet

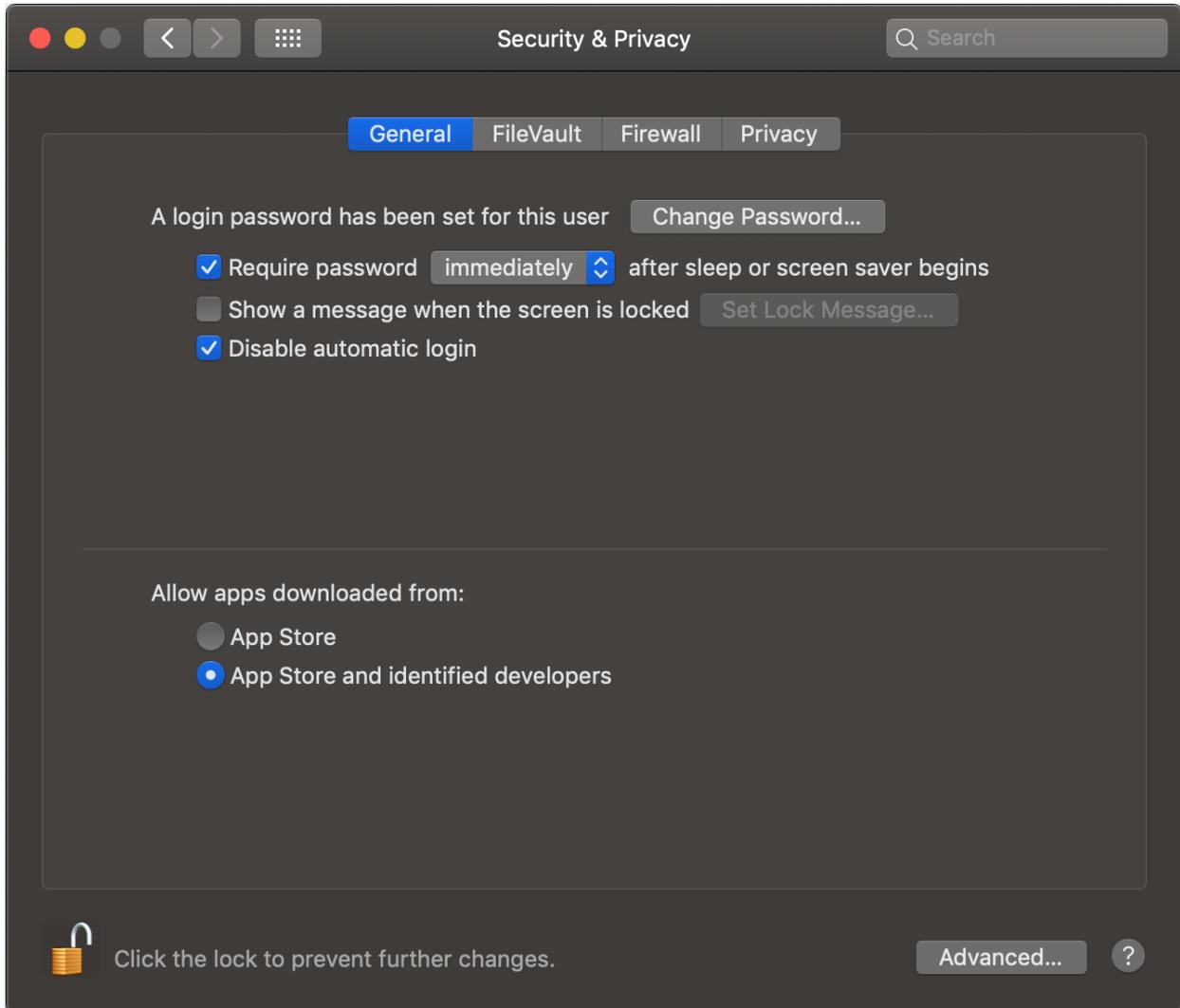
Once you launch the wallet for the first time, click 'Create new wallet'

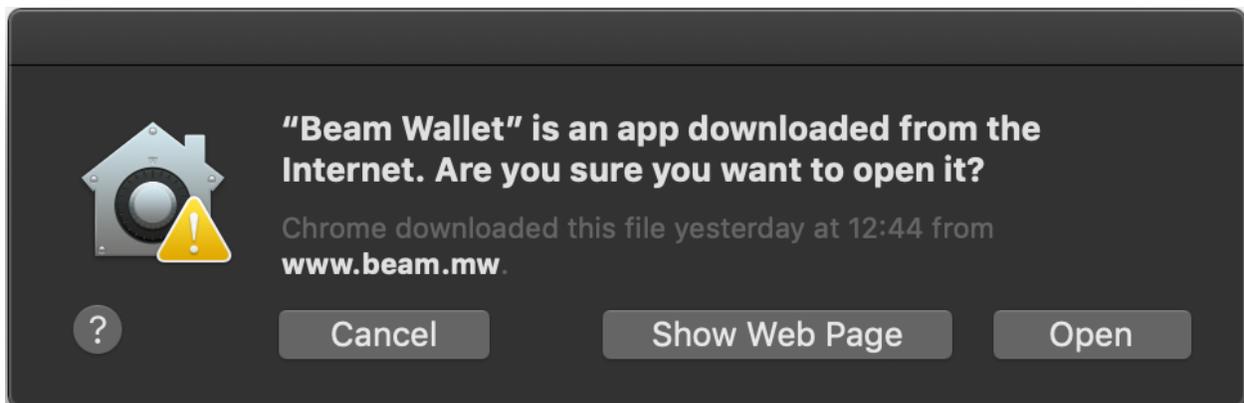
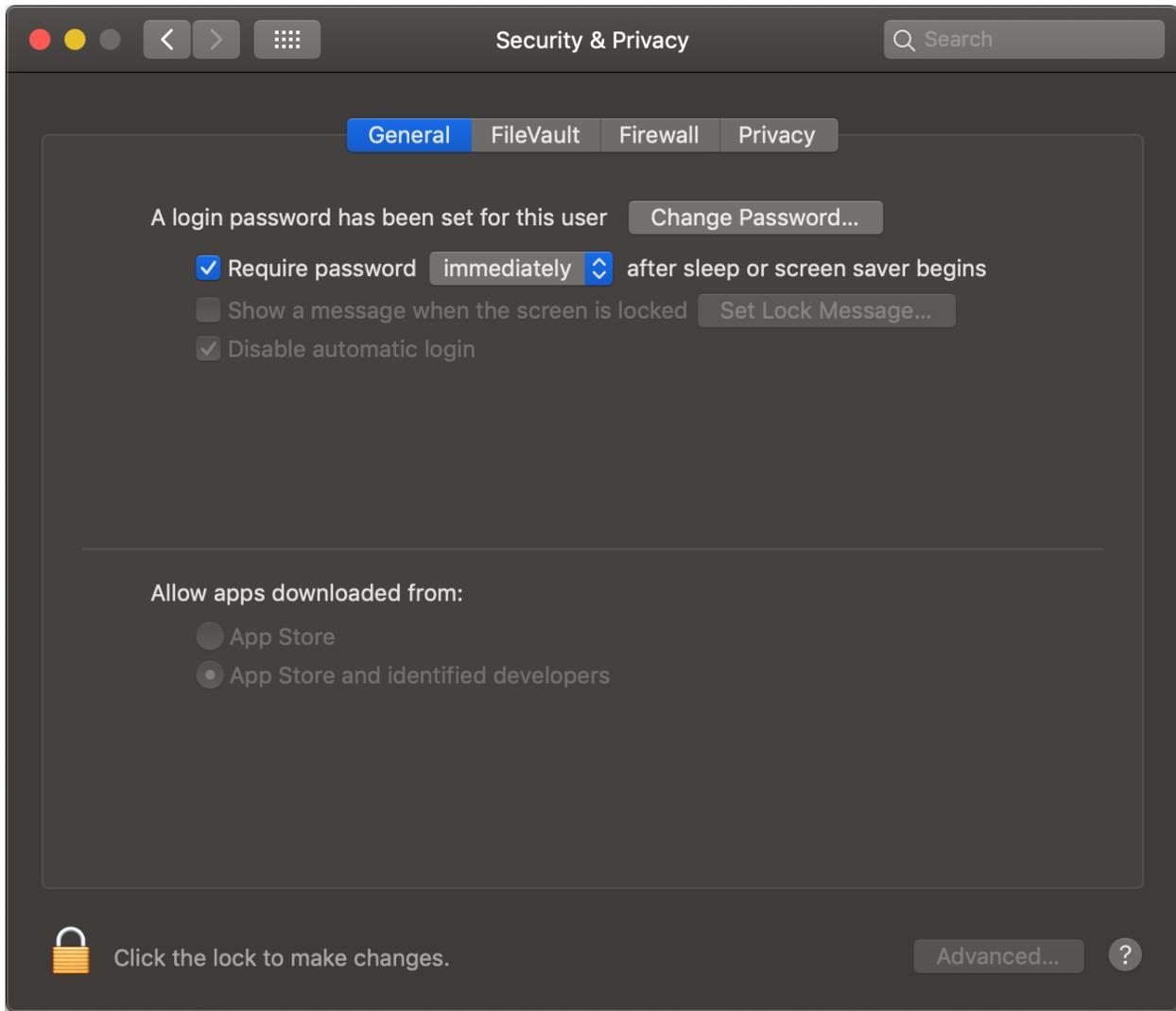
5.1.6 Generating seed phrase

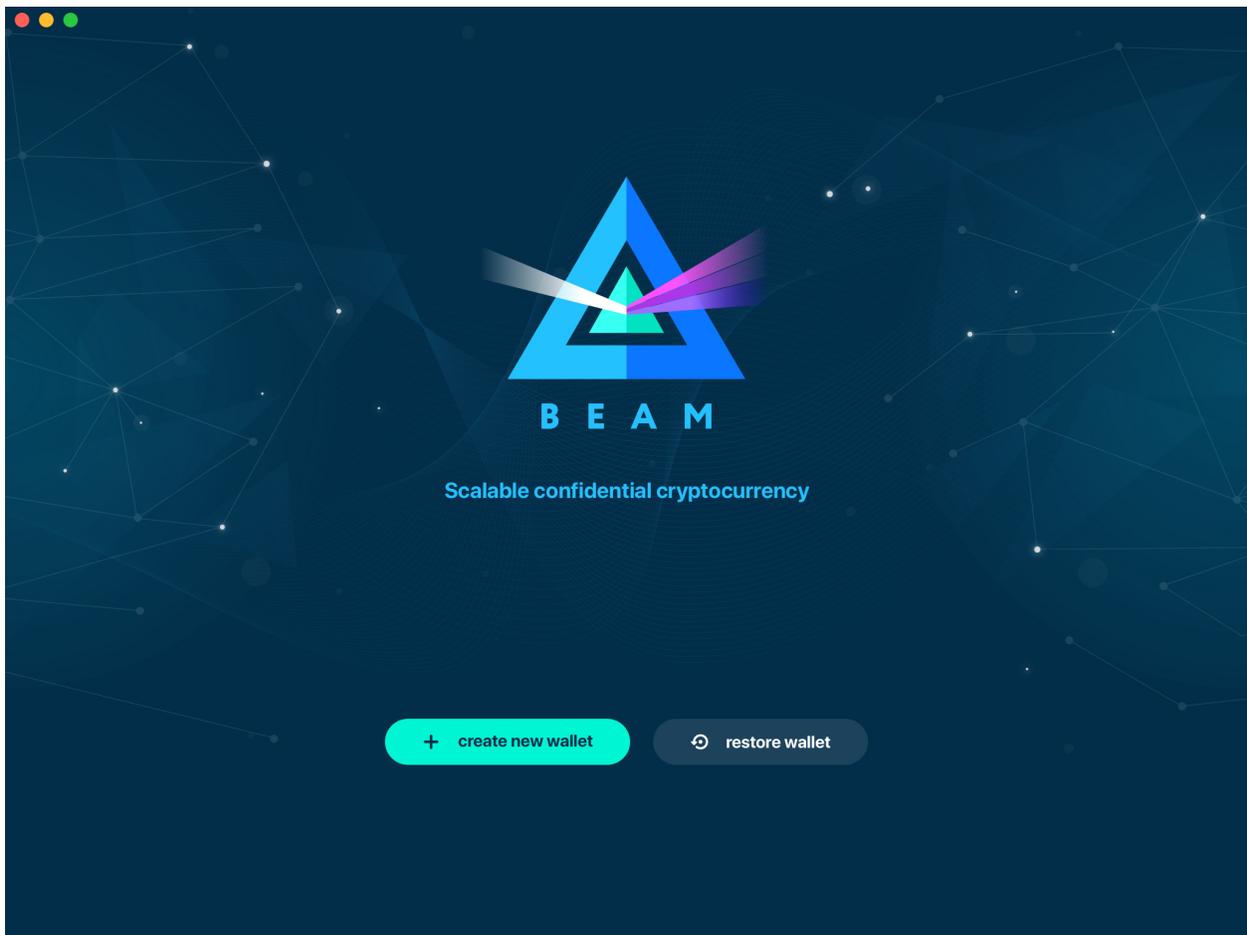
As a part of creating a new wallet, a new seed phrase will be generated for you.



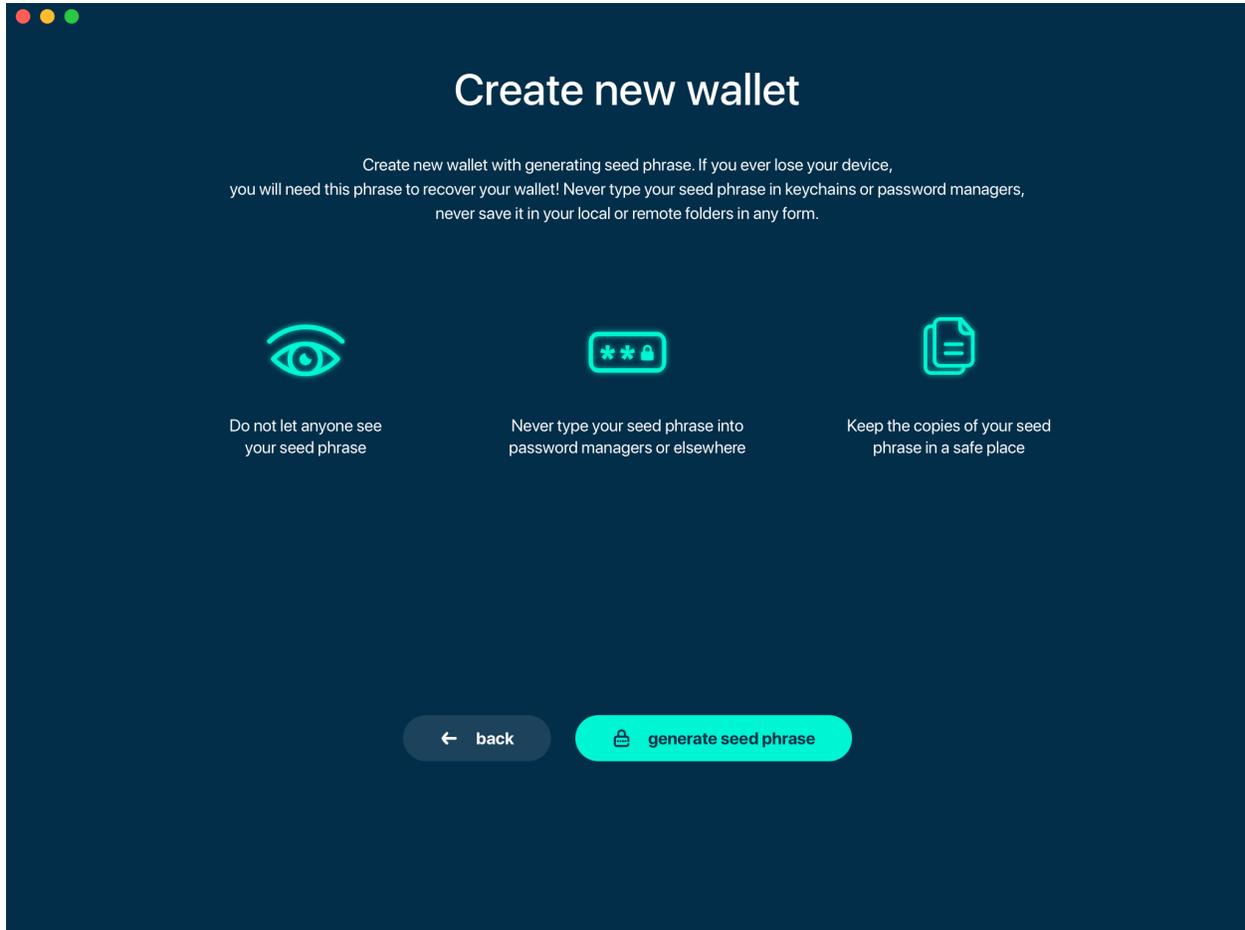








Attention: Seed phrase is the **most important secret you have to keep**. Knowing the seed phrase enables you (or anyone else) to access all your funds.



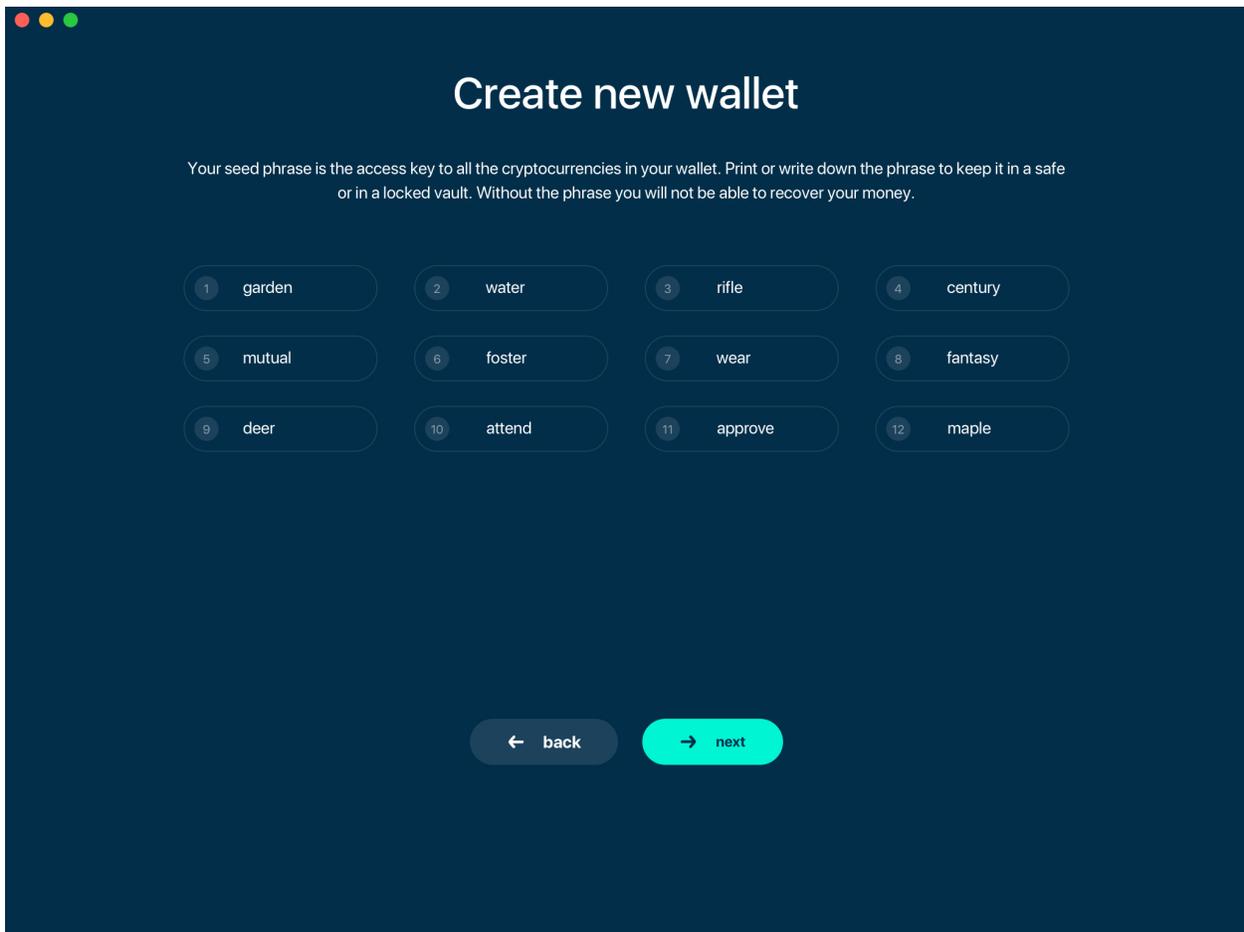
Attention: Seed phrase in the Beam wallet is *not* linked to email, phone number or any other identifier. You will need this phrase to restore your wallet when you lose or reformat your device, or want to access your funds from another device (your mobile phone or another desktop / laptop).

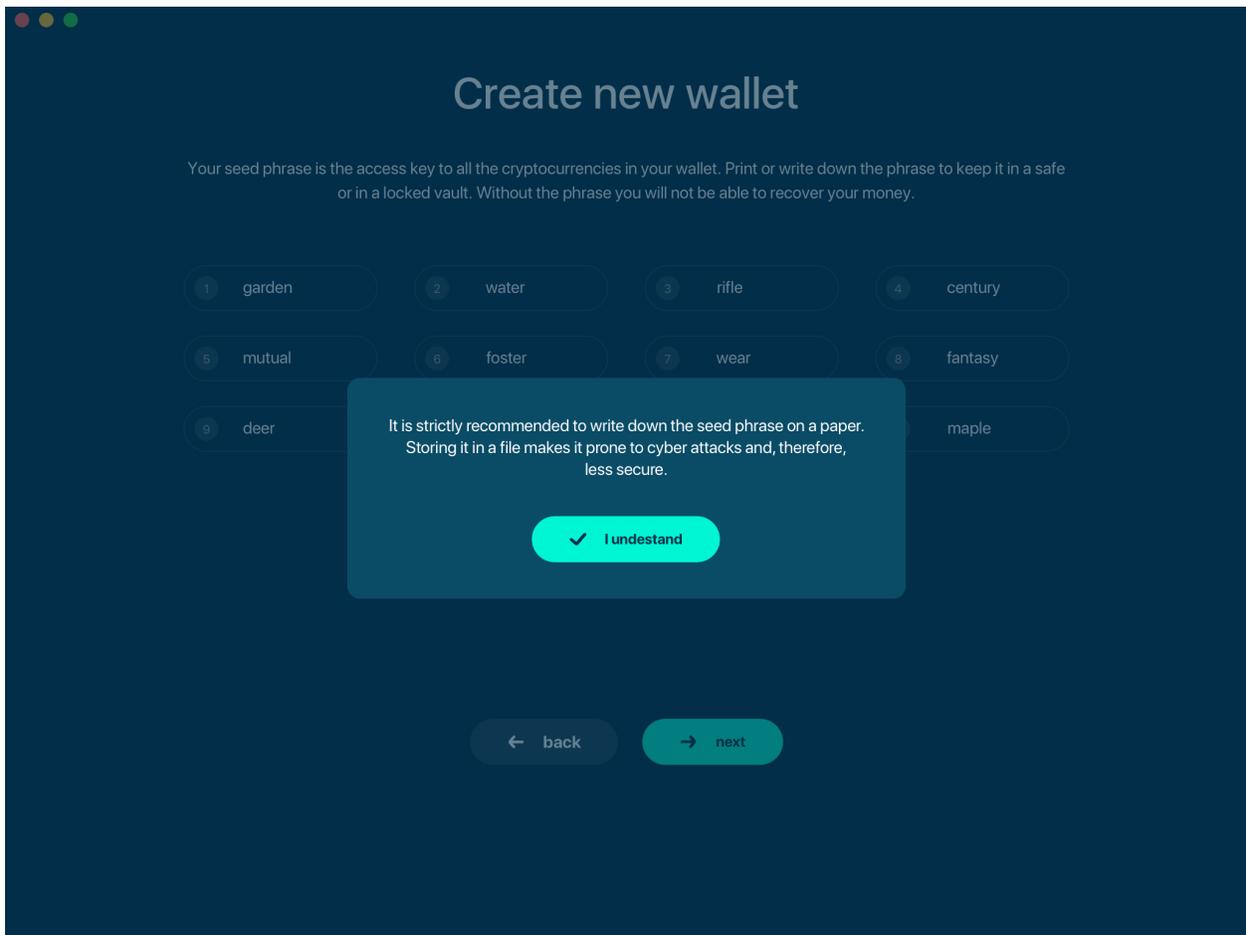
Attention: The seed phrase is **for your eyes only!** Make sure no one is looking over your shoulder. For the best security always do it on a clean air-gapped machine.

Did you write down your seed phrase correctly? Triple-check your handwriting again. The difference between `_F_unnel` and `_T_unnel` can be crucial when trying to restore a wallet with valuable funds in the far or near future.

Did you verify your handwriting? Now go find a safe space for the paper!

Important: Storing the seed phrase on your computer makes your funds prone to cyber attacks (read: much *less* secure). ‘Creative’ approaches like saving a screenshot of the wallet or your handwriting on your computer or in the cloud *may* sound like a good idea, but it is absolutely **not recommended**. If hackers get the access to your computer,

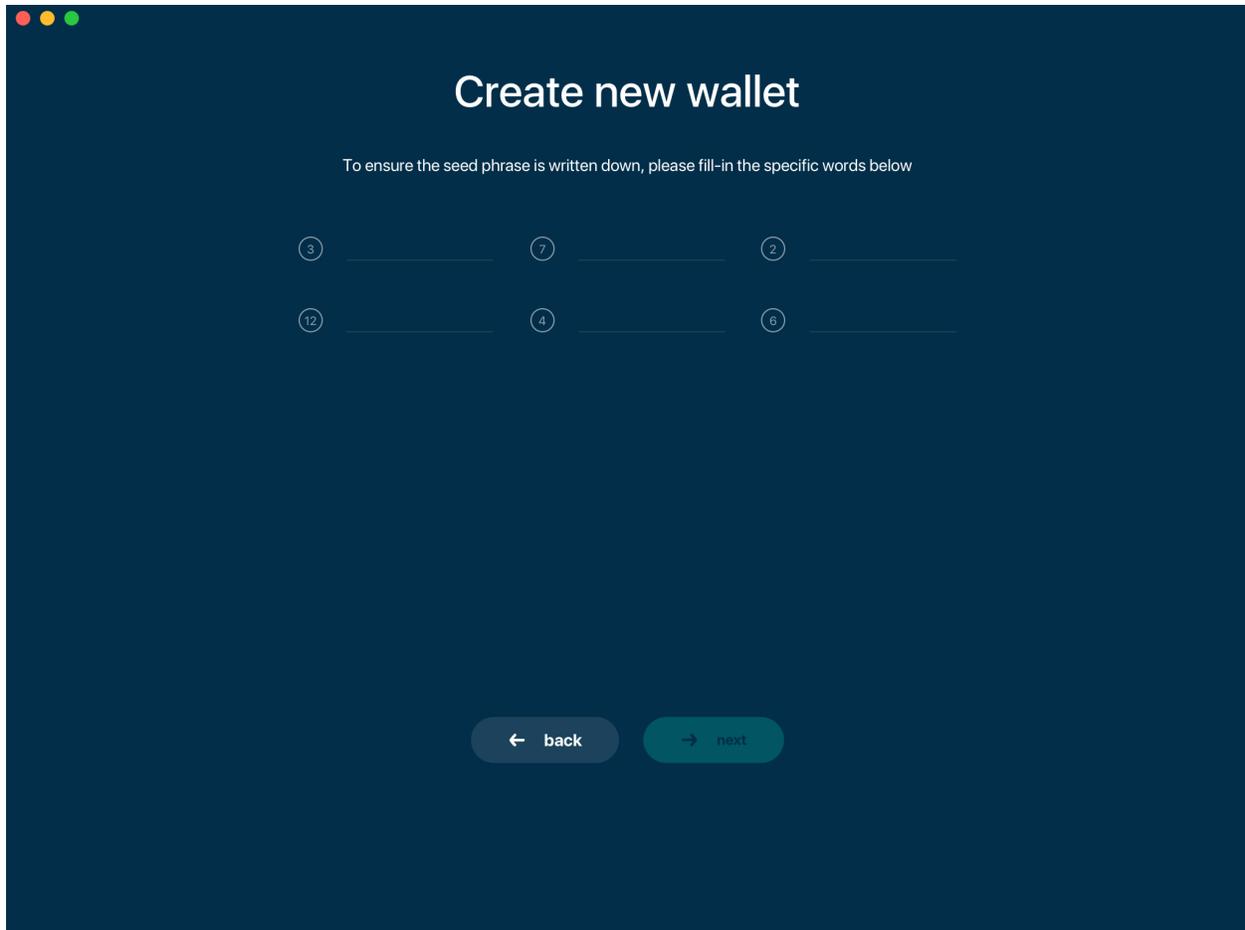




network drive etc., they can potentially steal your seed phrase by using OCR programs (which can scan pictures and transform them into plain text) and, therefore, get access to your funds.

Attention: Always store your seed phrase in a safe and secure location (and better more than one in different geo locations). Write it on a piece of paper. Do not store electronically neither as plain text nor in any other form!

In order to ensure that you have really written down your seed phrase, you will be asked to fill in the specific words from your seed phrase in random order.

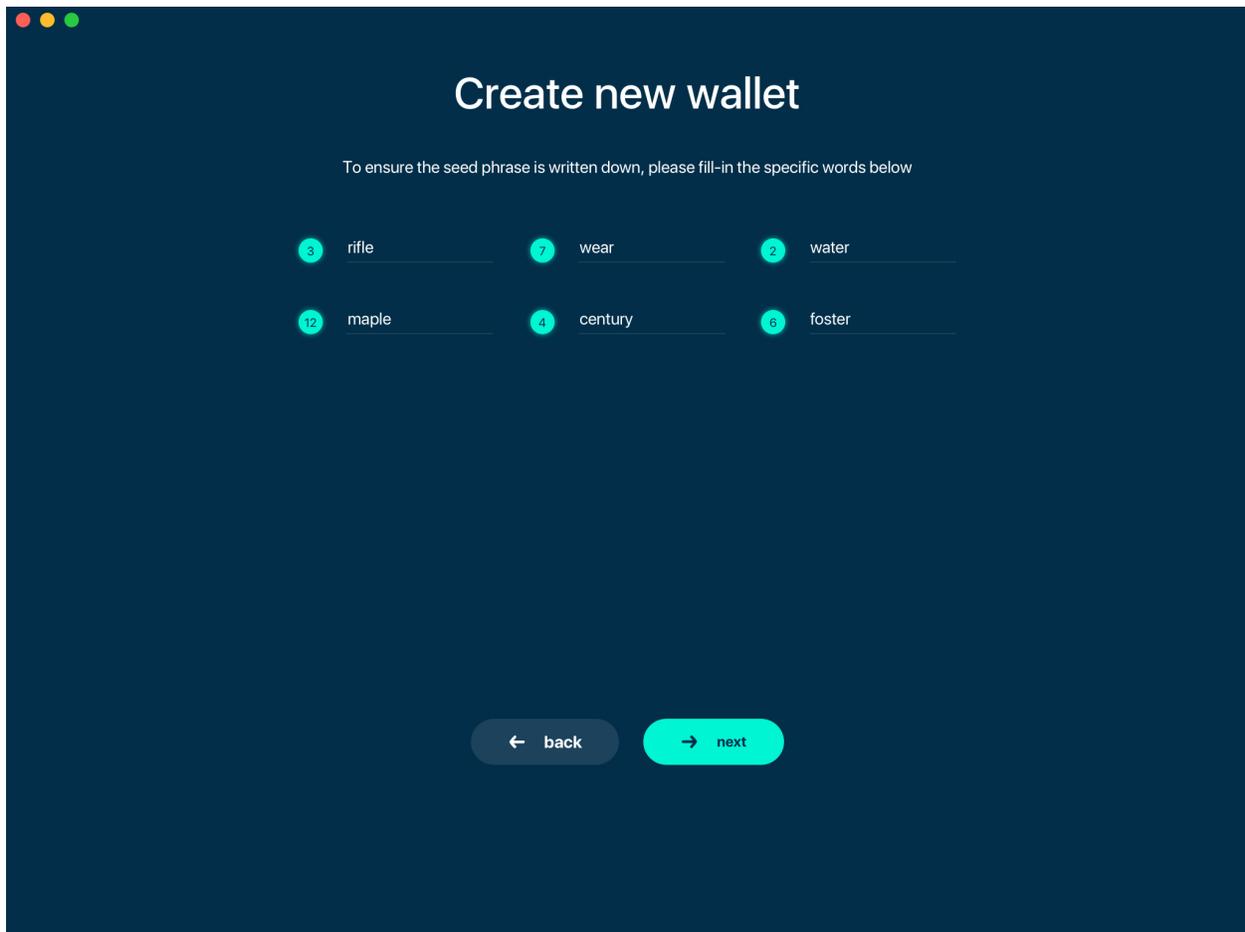


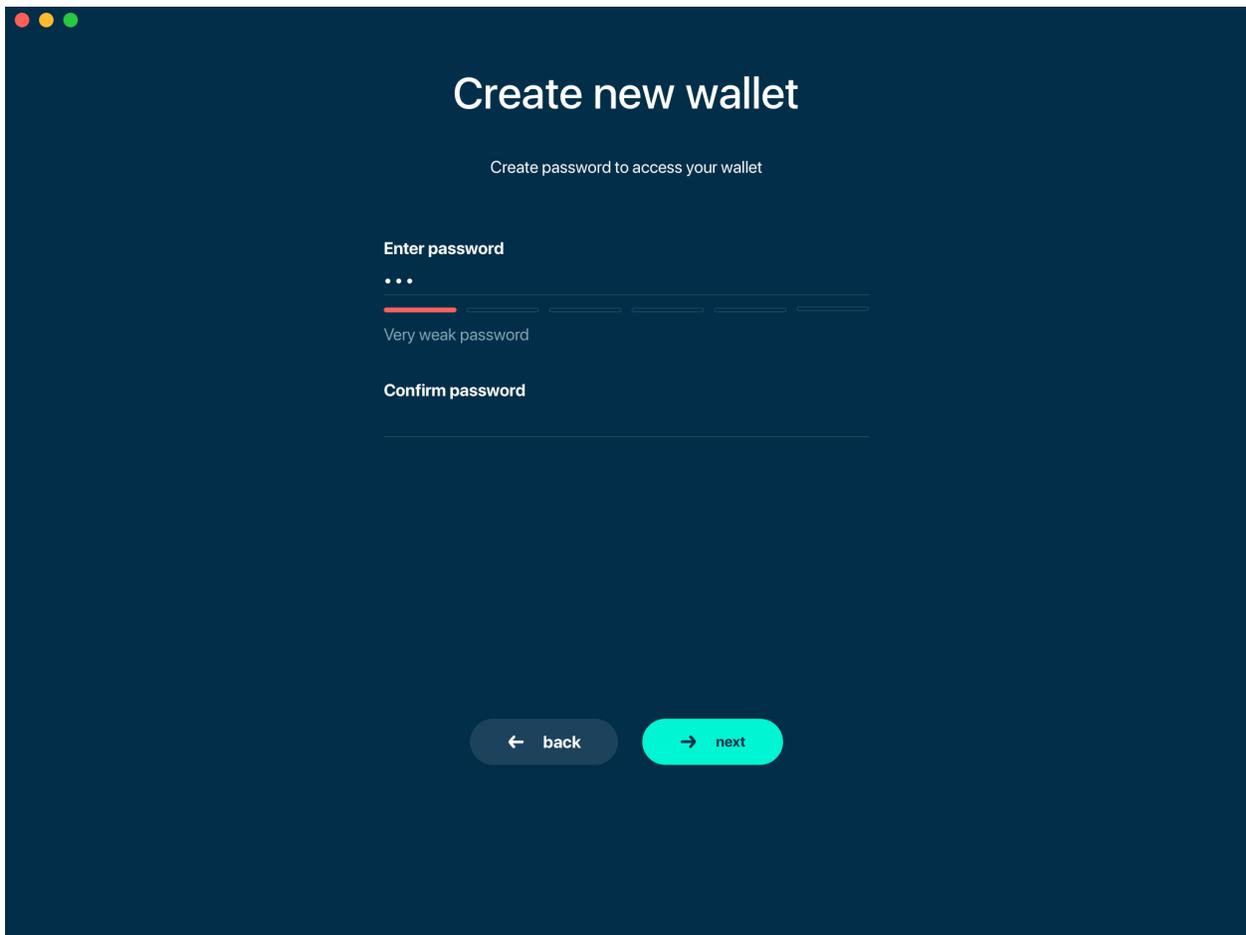
Only when you typed all the selected words correctly, you will be allowed to proceed to the next step.

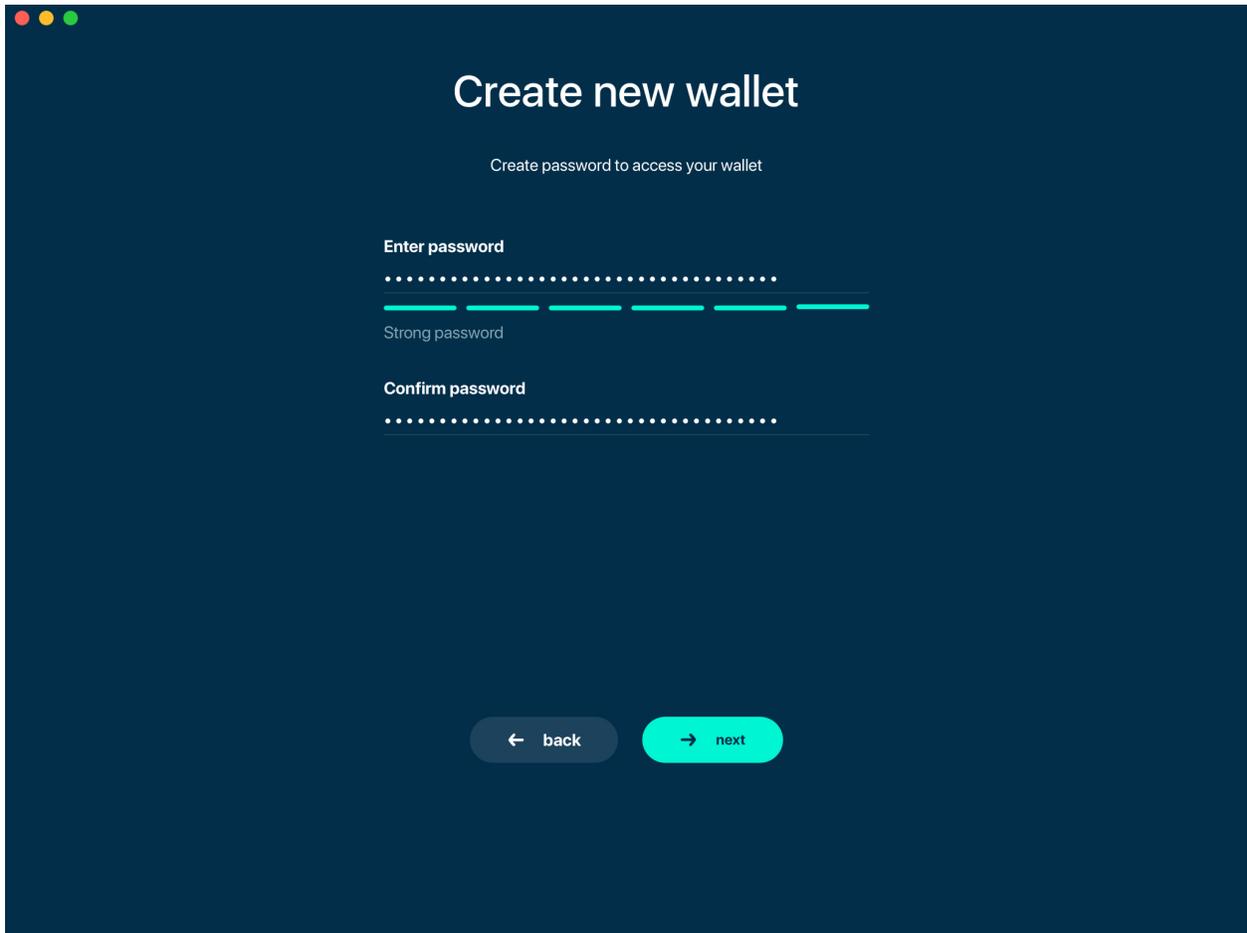
5.1.7 Setting wallet password

To access your Beam Wallet, you will need to create a password. This password is not the same as the seed phrase. Seed phrase identifies a wallet and enables access to all the funds stored in it from any device. Your wallet password provides with a second security layer in case someone gains access to your computer or has stolen your wallet database file. It is important to choose a strong password.

The wallet will provide some indication of password strength for your convenience. Do not count on it, however. Choose a password that is at least 8 characters long with a combination of letters, numbers, and symbols.







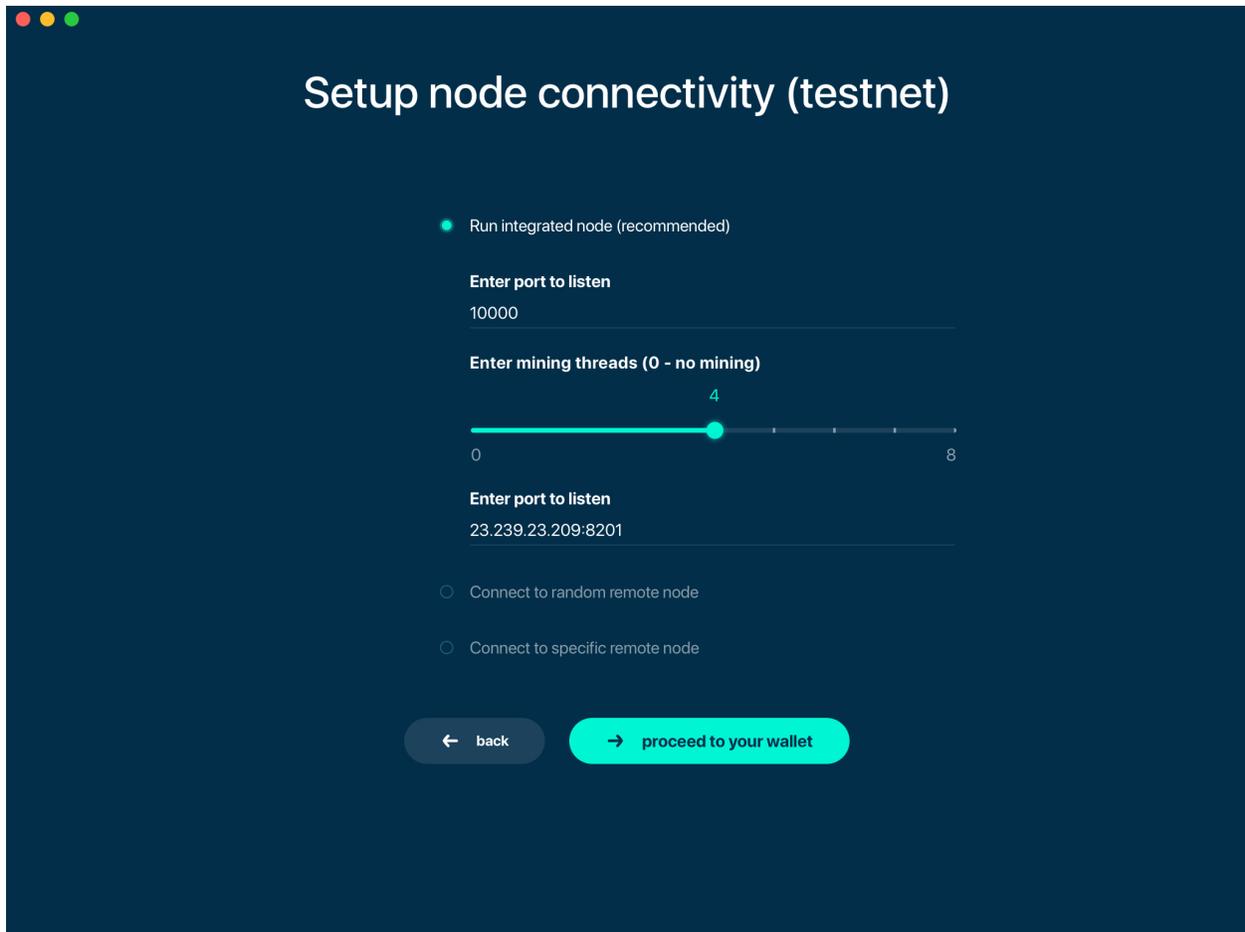
5.1.8 Choosing the node connectivity mode

Beam Desktop Wallet can connect to the network through:

- Integrated node
- Random remote node
- Specific remote node

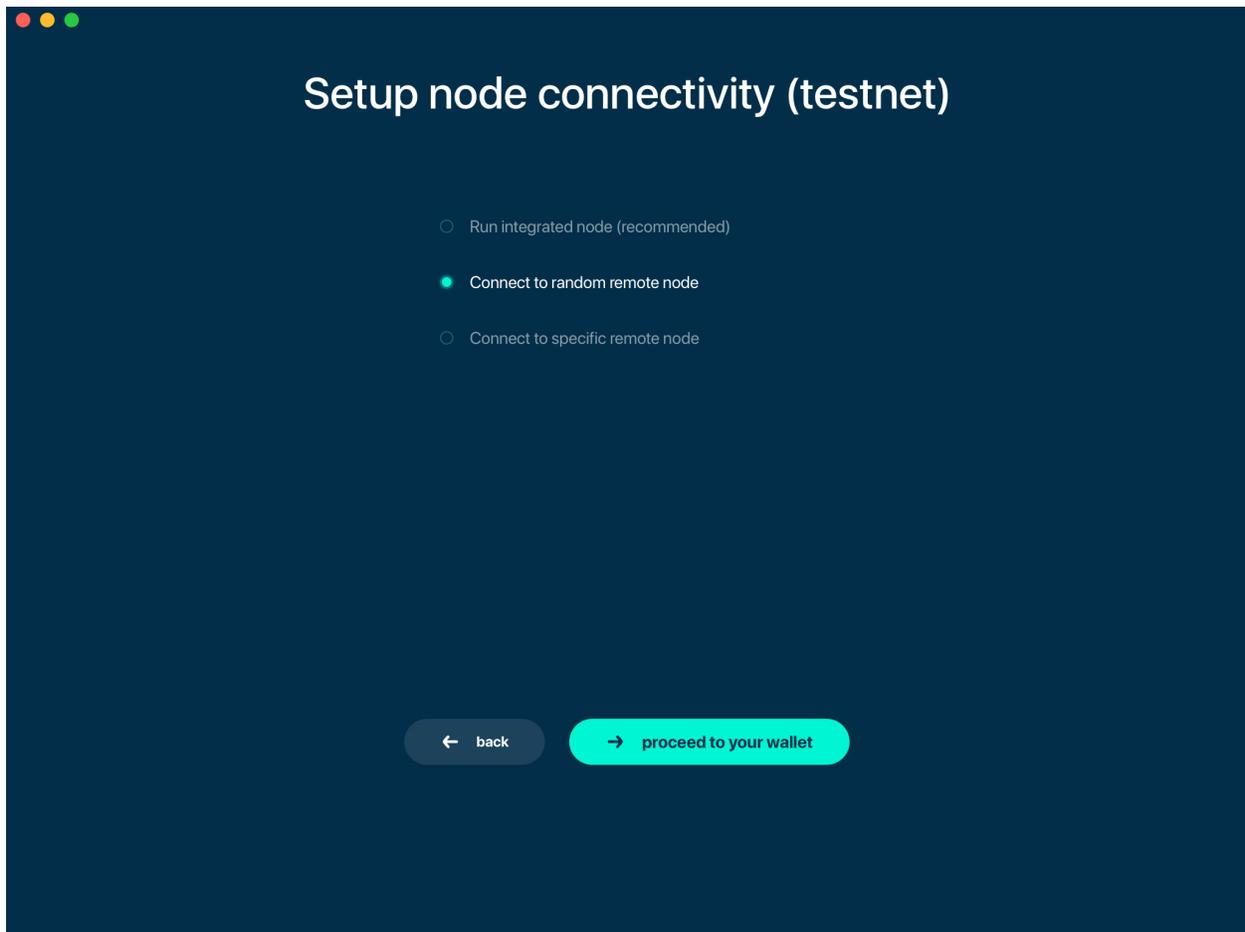
If you choose to run an integrated node from within the wallet, the trusted node will automatically verify the blockchain. This means you will automatically be connected to a node when you open your wallet.

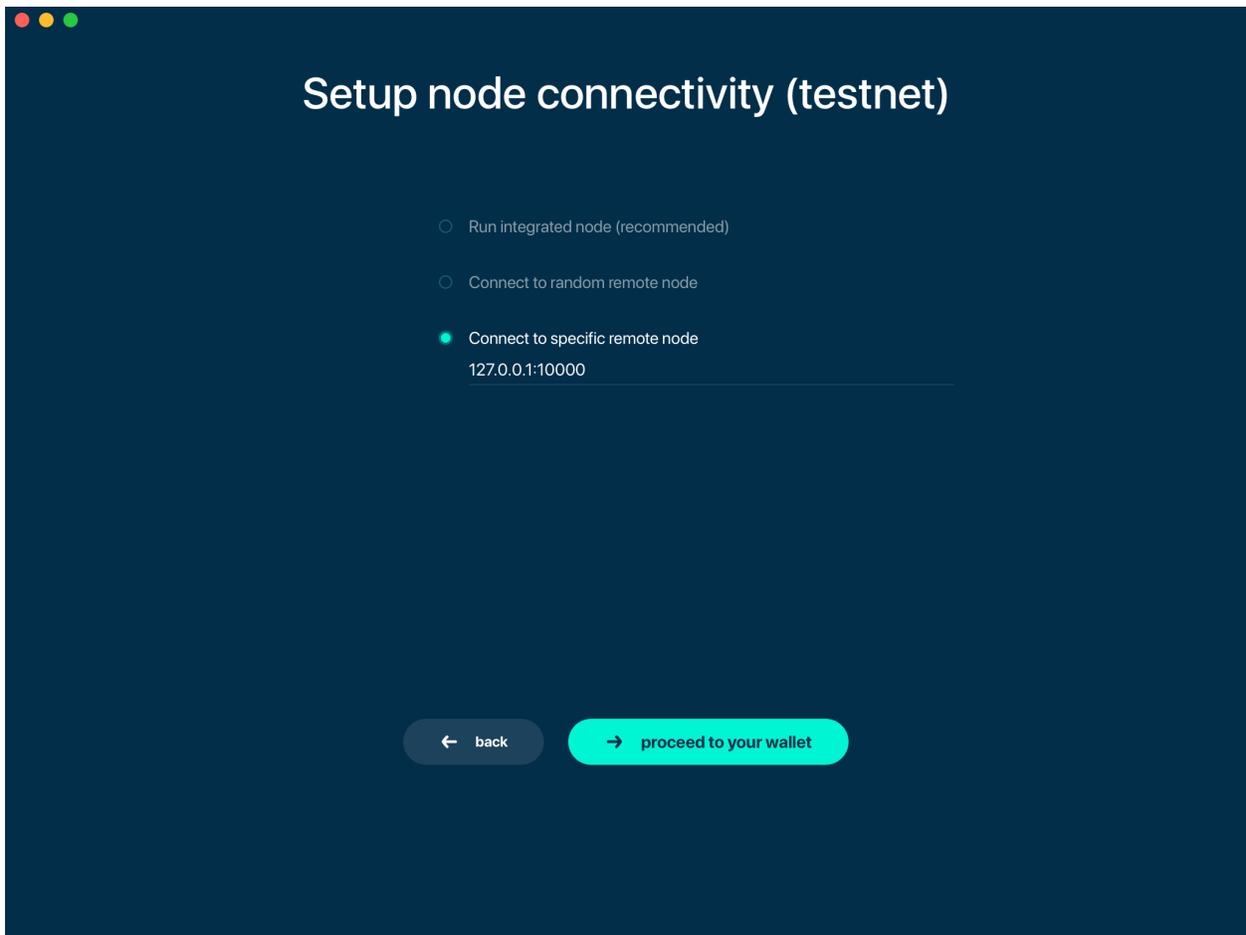
Note: When running behind a firewall you can change the default port the node will be listening on, and in case of CPU mining, set the amount of mining threads. You will be probably provided at least one default peer to connect to but you can always add more peers on the Settings screen. The recommended peers are published in the list of bootstrap nodes in the [downloads page on Beam official website](#).



Random mode allows you to automatically connect to random bootstrap node. In this mode Beam Wallet acts like a 'light client', it will create transactions but will have to trust the remote node for blockchain verification. It is recommended for lighter devices with limited CPU power and/or RAM memory.

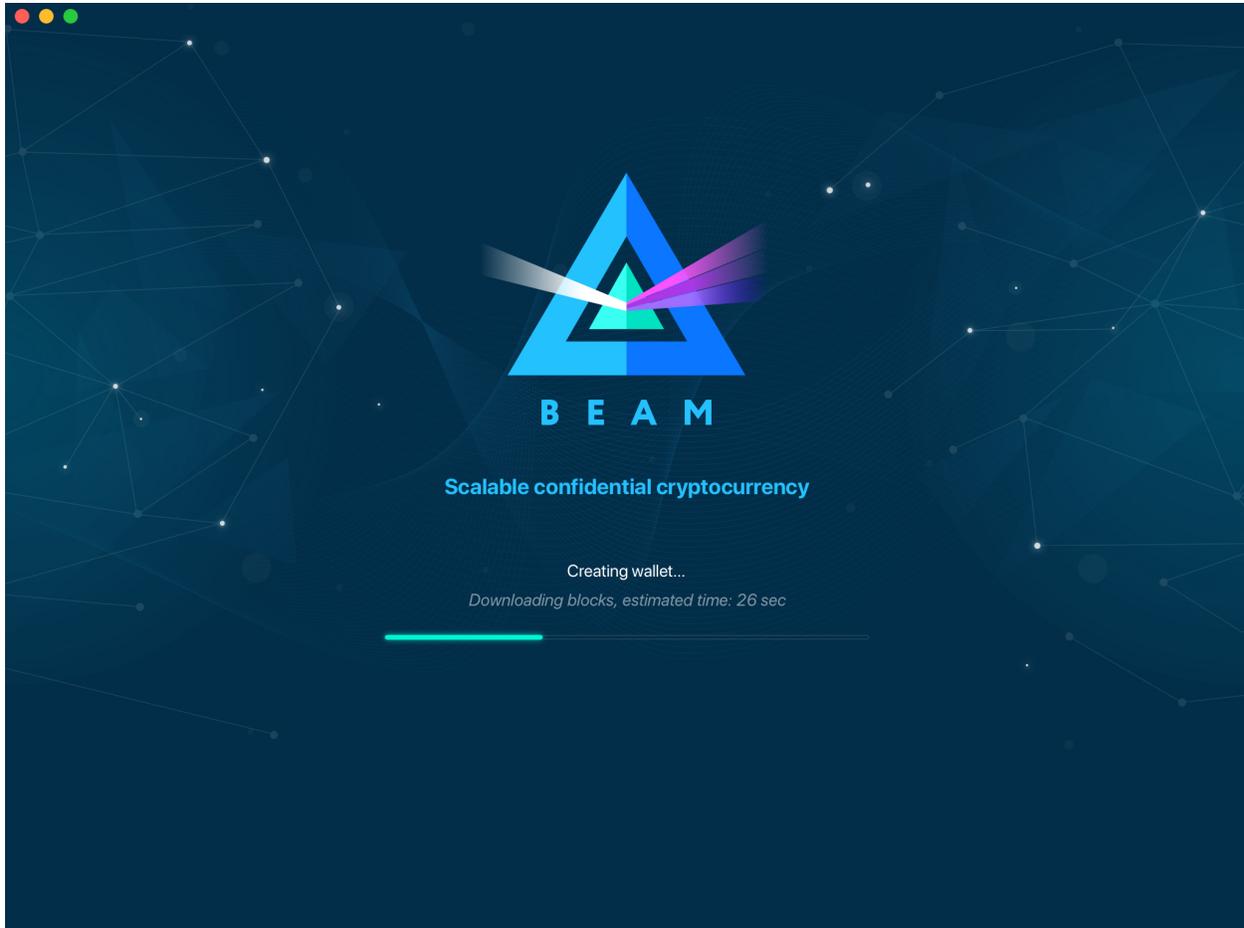
If you are running your own node somewhere (or maybe even more than one) and want to connect specifically to it, use the third option by providing the IP and port the node is listening on.





5.1.9 Synchronizing the wallet

Once the Wallet is connected, it synchronizes with the current blockchain state from the network. Be patient when running with the integrated node: the data downloading process might take some time. The wallet will first download and validate the latest 'macroblock' and then all the rest of the blockchain.



5.1.10 Main Screen

Once your wallet is created, the main screen will show up. In the future, the screen will pop-up automatically after you open the application and type in your password.

5.1.11 Wallet status indicator

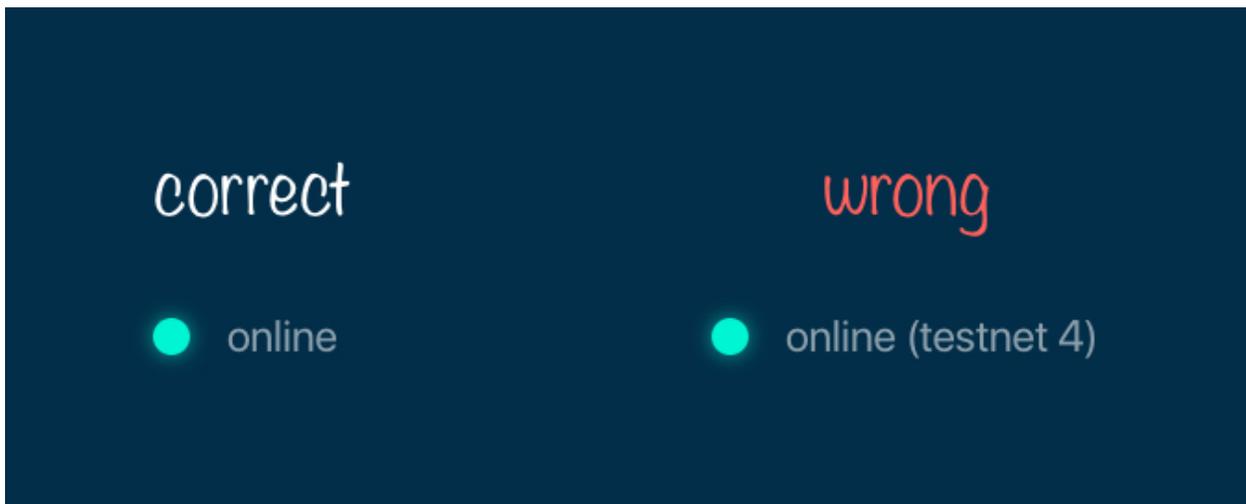
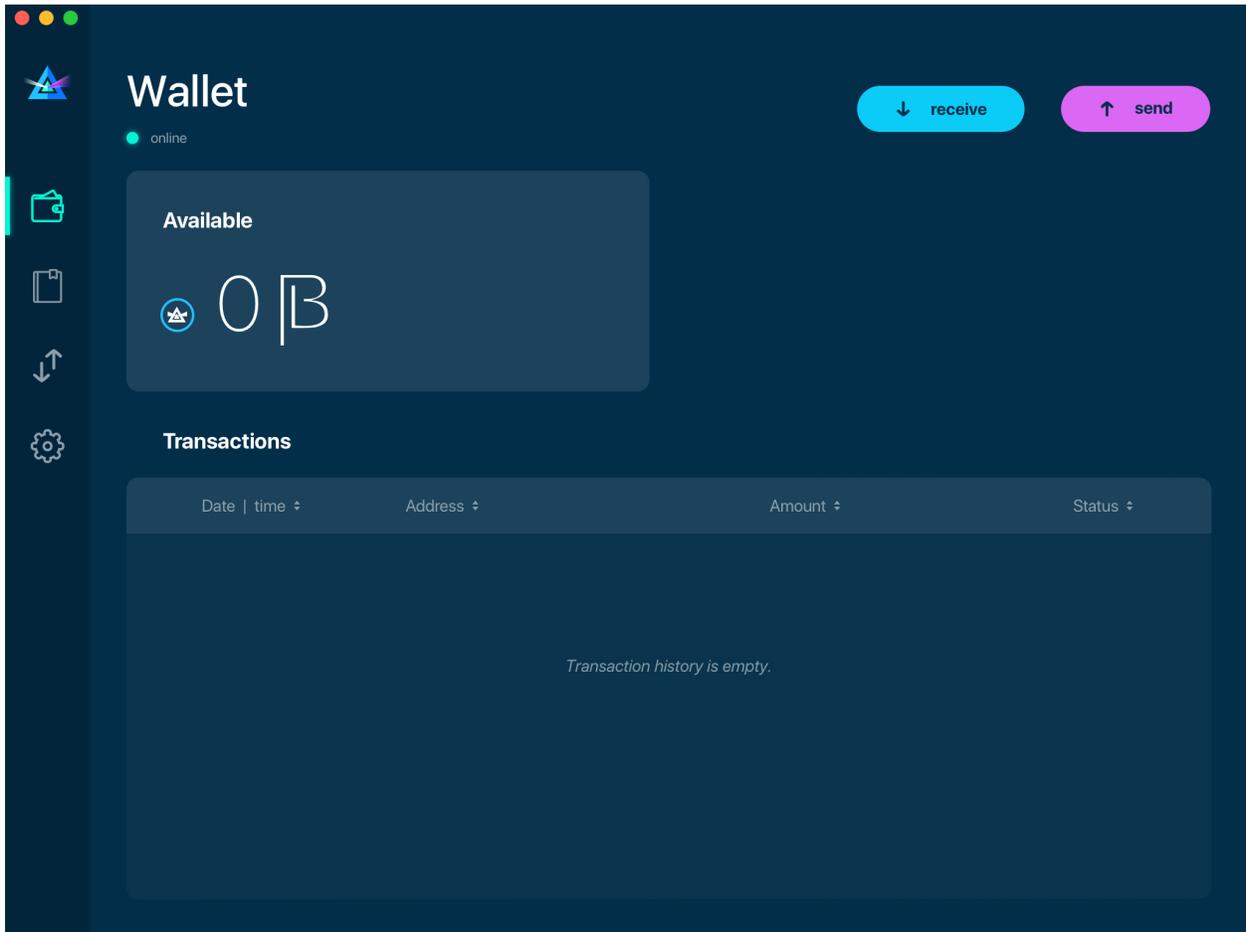
The main screen of the wallet shows the current balance as well as the transaction history and statuses. On the left, under the Beam logo, there is a toolbar that provides navigation between different wallet screens such as *Main Screen*, *Addresses Screen*, *UTXO Screen* and *Settings Screen*.

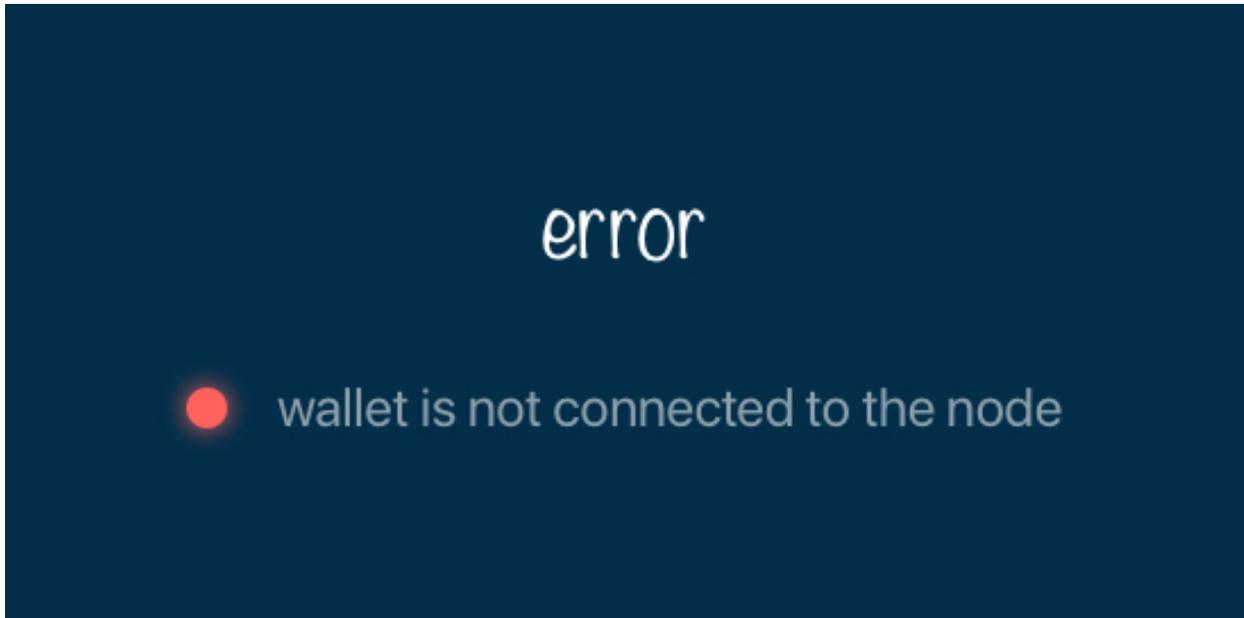
In the top left corner of the Main Screen, under 'Wallet', you see connection status indicator.

Green indicator means that the node is up and running connected to peers.

Red means it is either unable to connect to the node or the node has a problem you can help to solve.

In brackets, the indicator also specifies the network to which the wallet is connected:





- **Mainnet:** the network with real money and actual transactions.
- **Testnet<number>** (such as Testnet3, Testnet4 etc) : staging environments for trying new features in our wallet, node and miner software. For advanced users only.
- **Masternet:** new features under development, if you see this name in your wallet it means that you are very early adopter or Beam code contributor otherwise you've probably arrived to the wrong place.

Attention: Since Mainnet is the default network for the vast majority of Beam users, nothing is written by the online status.

Finally, node connectivity node is displayed (ie. integrated, random remote node or specific node).

5.1.12 Financial transactions

'Send' and 'Receive' buttons at the top right corner help us to *Sending BEAM* and *Receiving BEAM*. Let's start using the wallet!

5.1.13 Receiving BEAM

Before starting to receive BEAM for the first time, please read first about what `:ref:'address'` is.

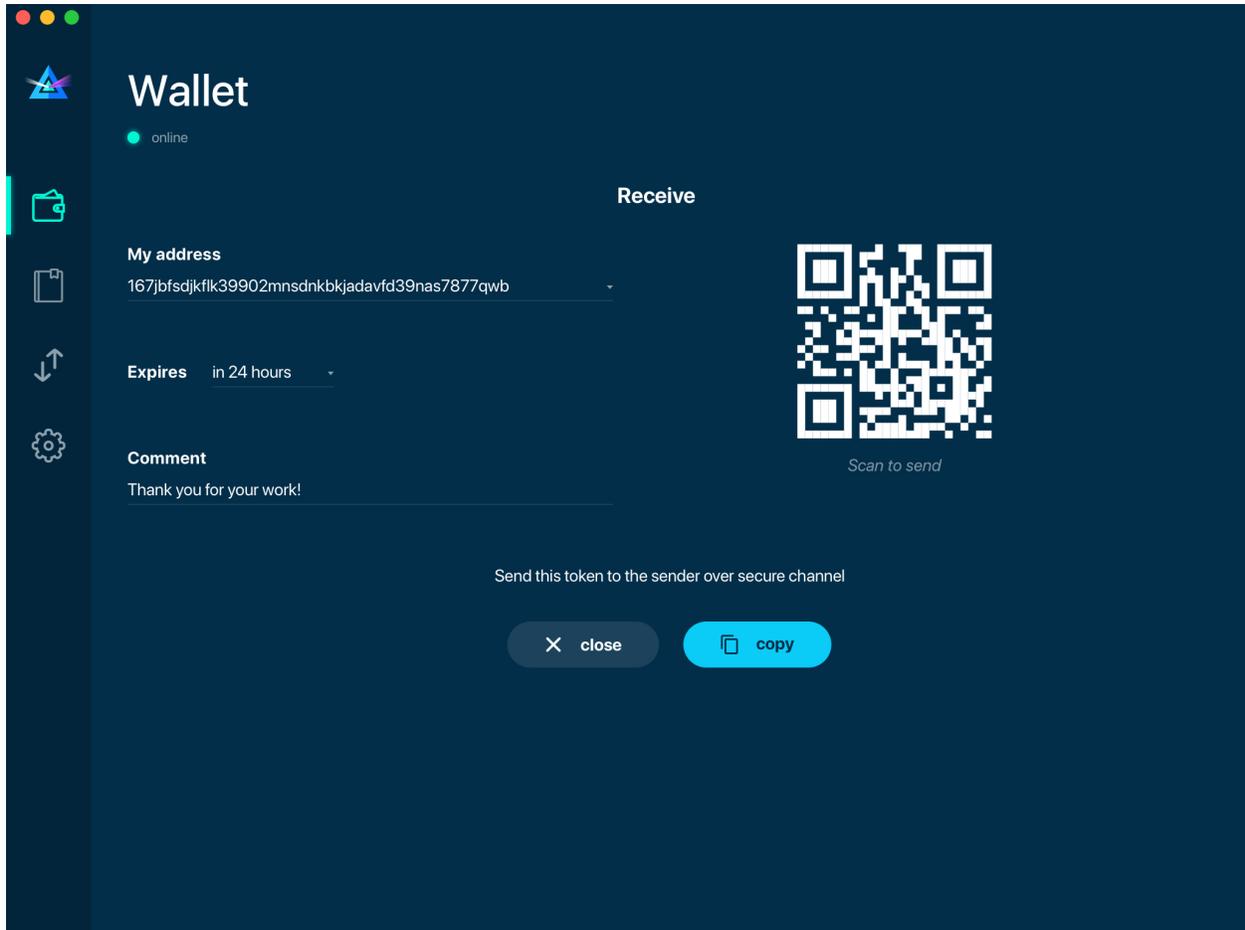
Here is how the process of receiving BEAM looks like from a Receiver's perspective:

- Generate an address
- Send your address to the Sender person **over a secure communication channel**
- Both Sender and Receiver's Wallet must be online at the same time to complete a transaction.

It's possible to reuse an address that already exists, more on that later.

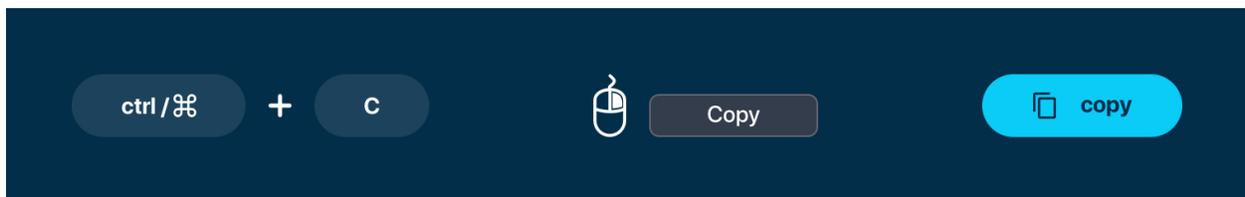
5.1.14 Generate an address

Proceed to the main screen and click the blue ‘Receive’ button at the top right corner. This will open the receive screen.



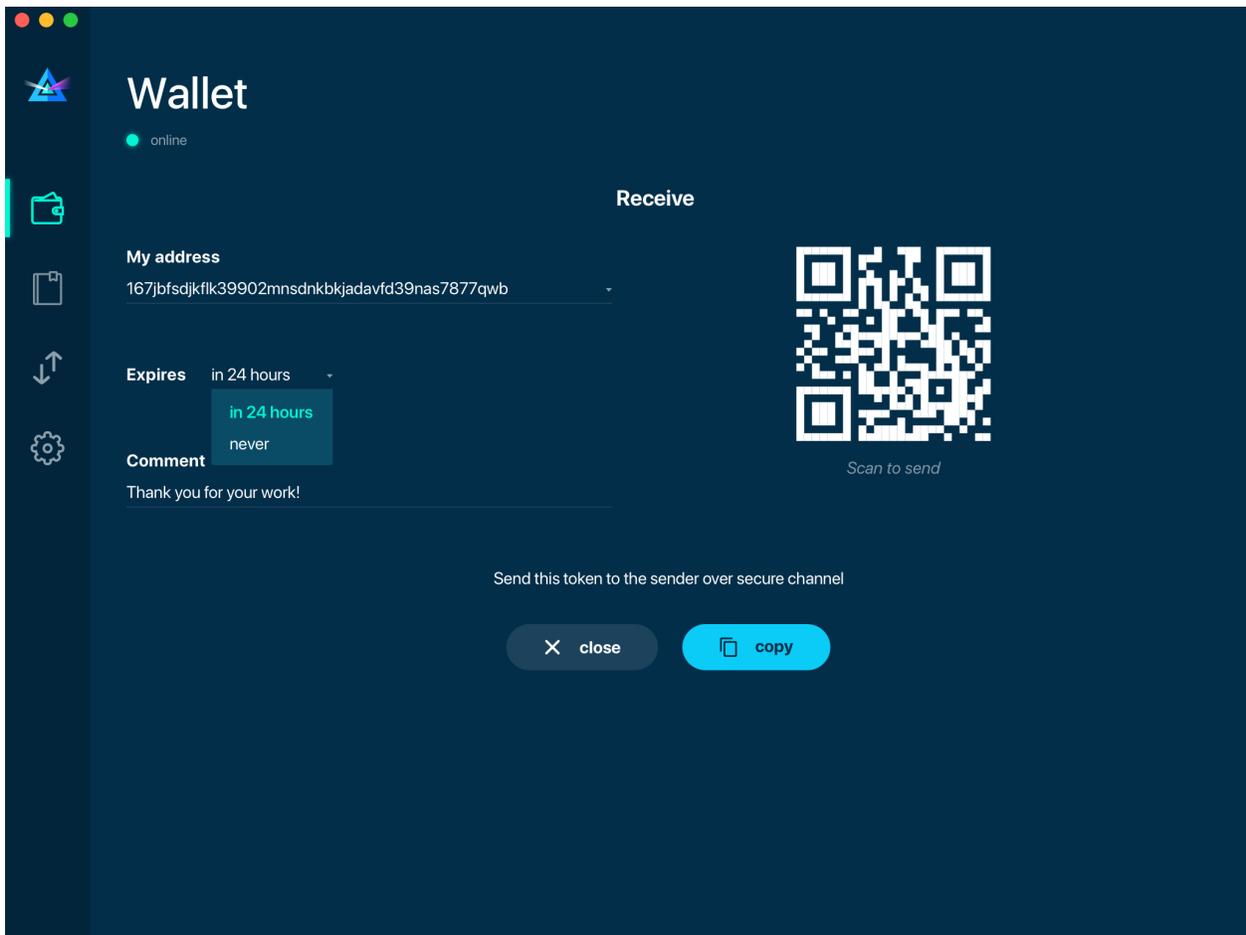
Copy and paste the newly generated Beam address to send to Sender over a **secure communication channel**. There are three ways to do it:

- By selecting the address and clicking Command-C or Ctrl-C (depending on your platform)
- By right-click on the address and choosing ‘Copy’ from the drop-down menu
- By clicking the ‘Copy’ button



A new Beam address is generated every time the ‘Receive’ screen is opened. By default, the address is valid for 24 hours. It is best to give the Sender your address closest to the time they will be sending BEAM so the address does not expire.

You can set the expiration time to ‘Never’ for this address only by selecting the value in the ‘Expires’ drop down.

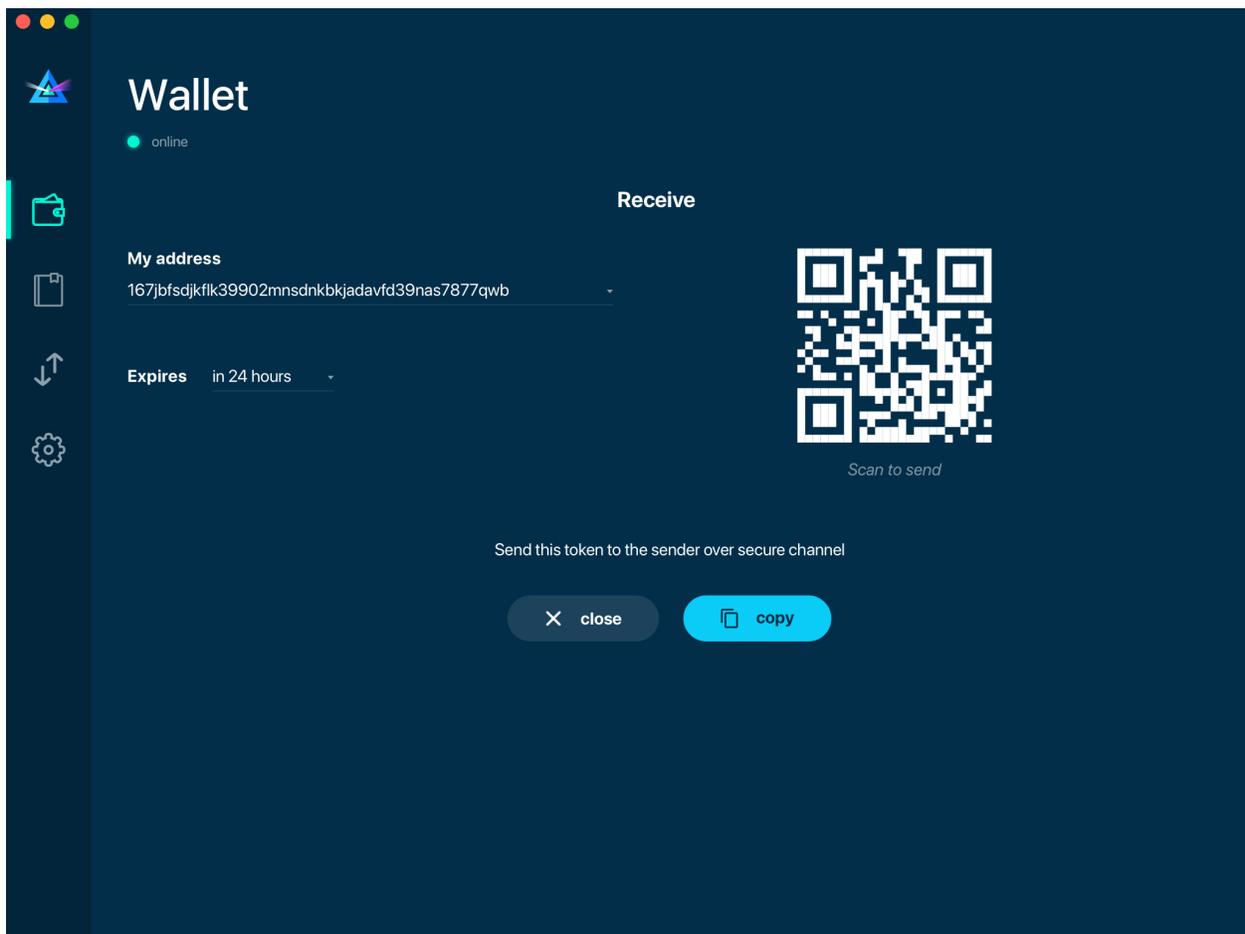


Warning: The permanent addresses that never expire are only useful in limited special cases, like working with mining pools or exchanges. For ultimate privacy, do not use this option for regular transactions, rather always create a new address for each transaction.

Attention: You have to click ‘Close’ button in the screen for the address to become active.

5.1.15 QR code

If the Sender uses a mobile app, he can quickly scan the QR code instead of receiving, copying and pasting the alphanumeric address. This feature will be available soon in the upcoming Beam Wallet mobile app.



5.1.16 Comment

You can add a comment when creating the receiving address. The comment is never sent to the network, it is only visible inside your wallet and is used for internal bookkeeping only.

The comment can be seen on the ':ref:’Address screen’ and in the extended transaction view.

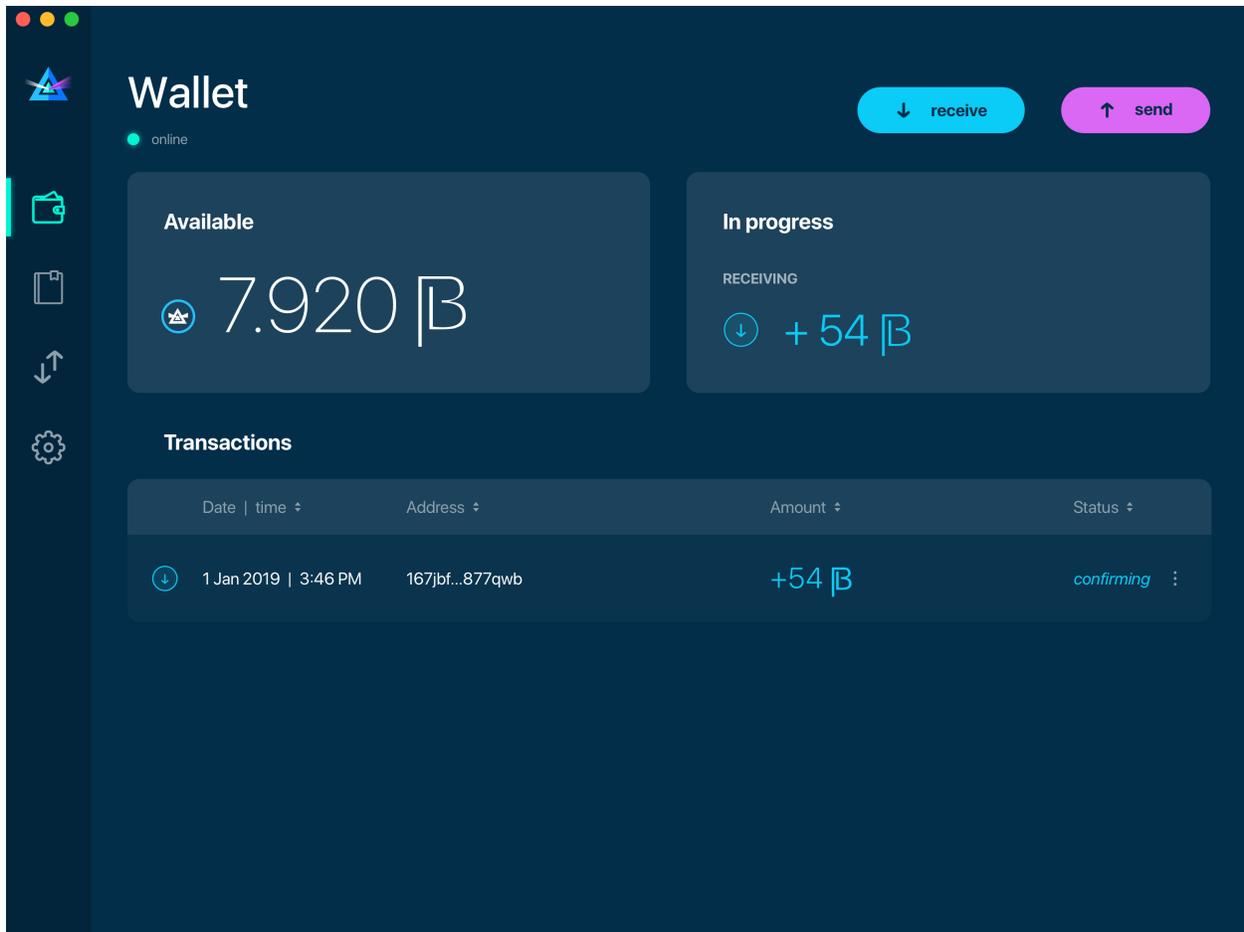
5.1.17 Sending the address

Attention: When sending the address make sure you use a secure communication channel.

Attention: Make sure the entire address is sent to the Sender as it's longer than it appears on the screen. Don't forget to double check the value in whichever messenger app of your choice because viruses and malware on your computer may change your address while it's in the clipboard.

5.1.18 Completing the transaction

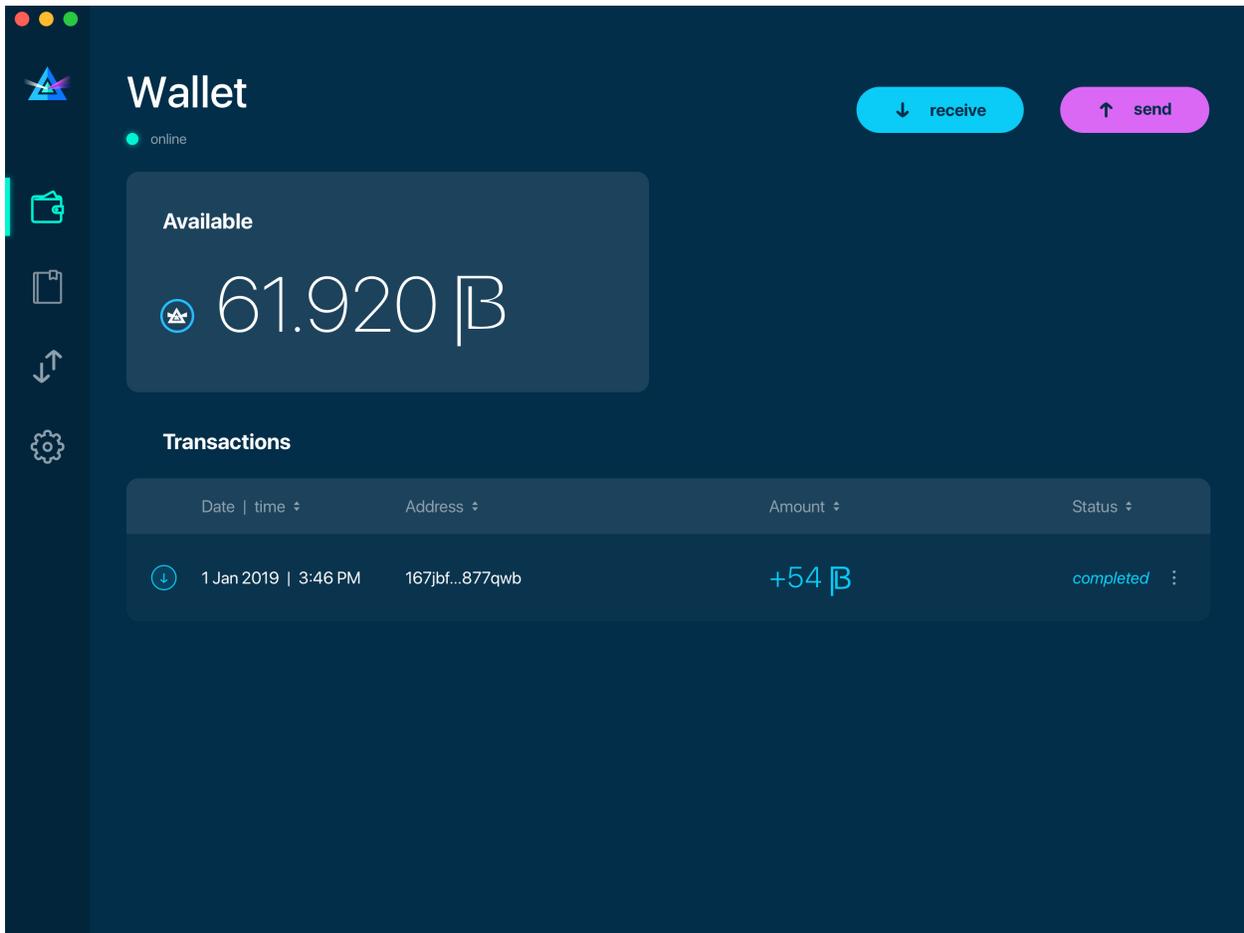
Once Sender initiates the transaction, you will see on the new transaction appear in the transaction list on the main screen. The amount sent will also appear on the 'In progress' box.



Normally, a transaction will pass through the following stages:

- **In Progress** - the phase when the transaction is being created by the Sending and Receiving wallets.
- **Confirming** - the phase after the transaction was sent to the nodes, but before it is mined.
- **Completed** - the phase after the transaction is fully mined and confirmed.

Once transaction is complete, the available balance will be updated and the 'In Progress' box will disappear.



5.1.19 Sending BEAM

Before starting to send BEAM, please read first about what `:ref:Address` is.

Here is how the process of sending BEAM looks like from a Sender's perspective:

- Receive the address the funds should be sent to
- Send BEAM to Receiver
- Stay online until Receiver confirms the transaction

5.1.20 Receiving the address

Attention: Make sure that the address is received untampered by using a **secure communication channel**.

Attention: When copying the address to the Beam Wallet app please verify visually that the address in the wallet looks exactly like the address in the secure messaging app, because viruses and malware on your computer may change your address while it's in the clipboard.

5.1.21 Sending funds

In order to send BEAM, you will need to click the magenta 'Send' button at the top right corner. This will open the Send screen.

Make sure you have the correct address and paste the Receiver's Beam address in the 'Send To' field.

To help to identify the transaction, you may also choose to fill in the optional Comment field. The comment will remind you what or who the transaction is for. The comment is stored locally, thus it will only be visible in your wallet for bookkeeping purposes.

The comment can be seen on the `:ref:Address` screen:

The comment is also displayed in the extended transaction view:

Select the transaction amount in BEAM you want to send. Transaction amount is in BEAM and may contain fractional values such as 1.25 BEAM or 11.3 BEAM and the like. Keep in mind you also have to pay a transaction fee, hence the amount to send plus the fee must be equal to or less than the available balance.

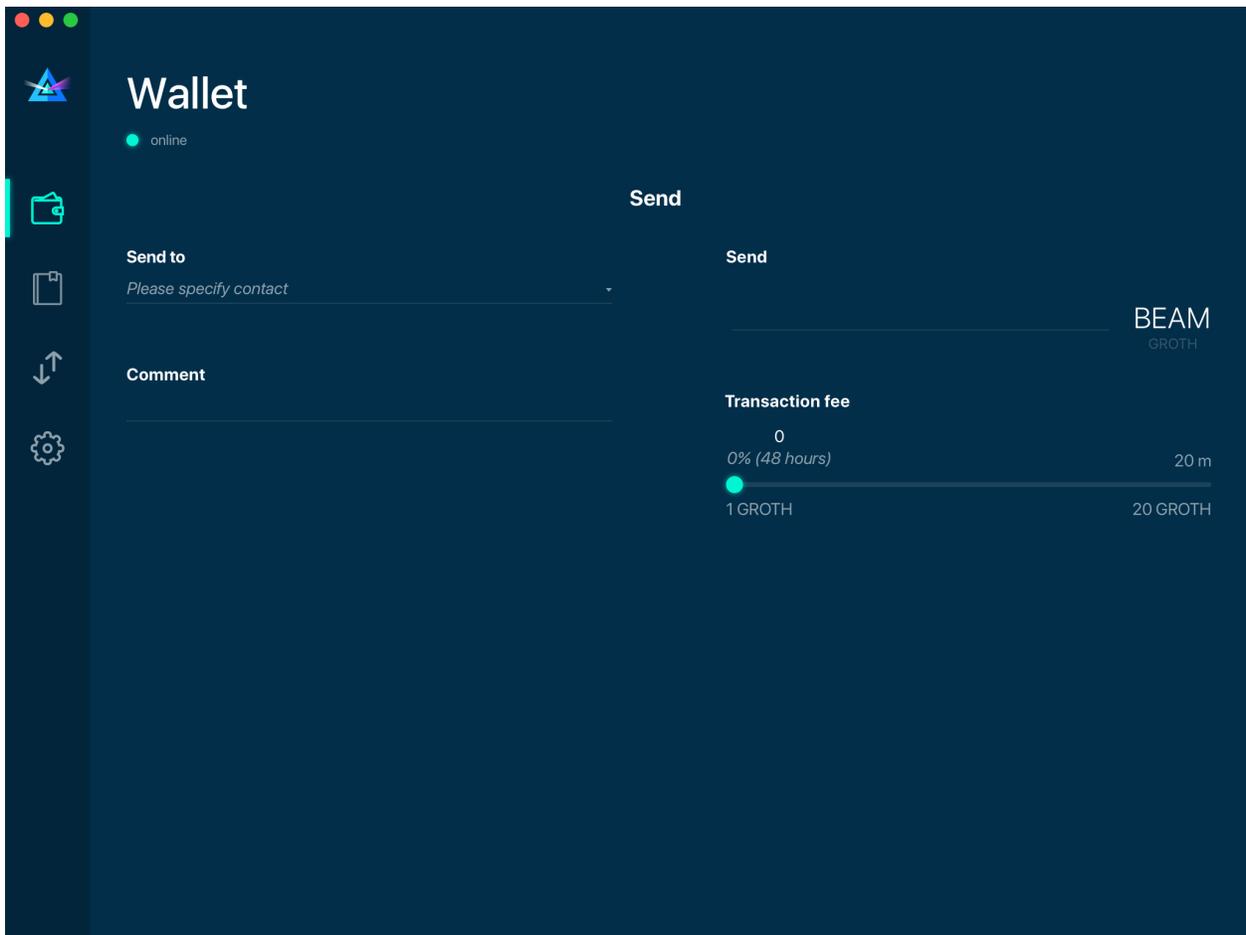
Transaction fees are specified in GROTH (100 millionths of BEAM). Amount of fees you need to pay depends on the current status of the network and average fee sizes. Simply said, the higher transaction fee will help miners to prioritize your transaction. To determine the current average fee size use [Beam Blockchain Explorer](#).

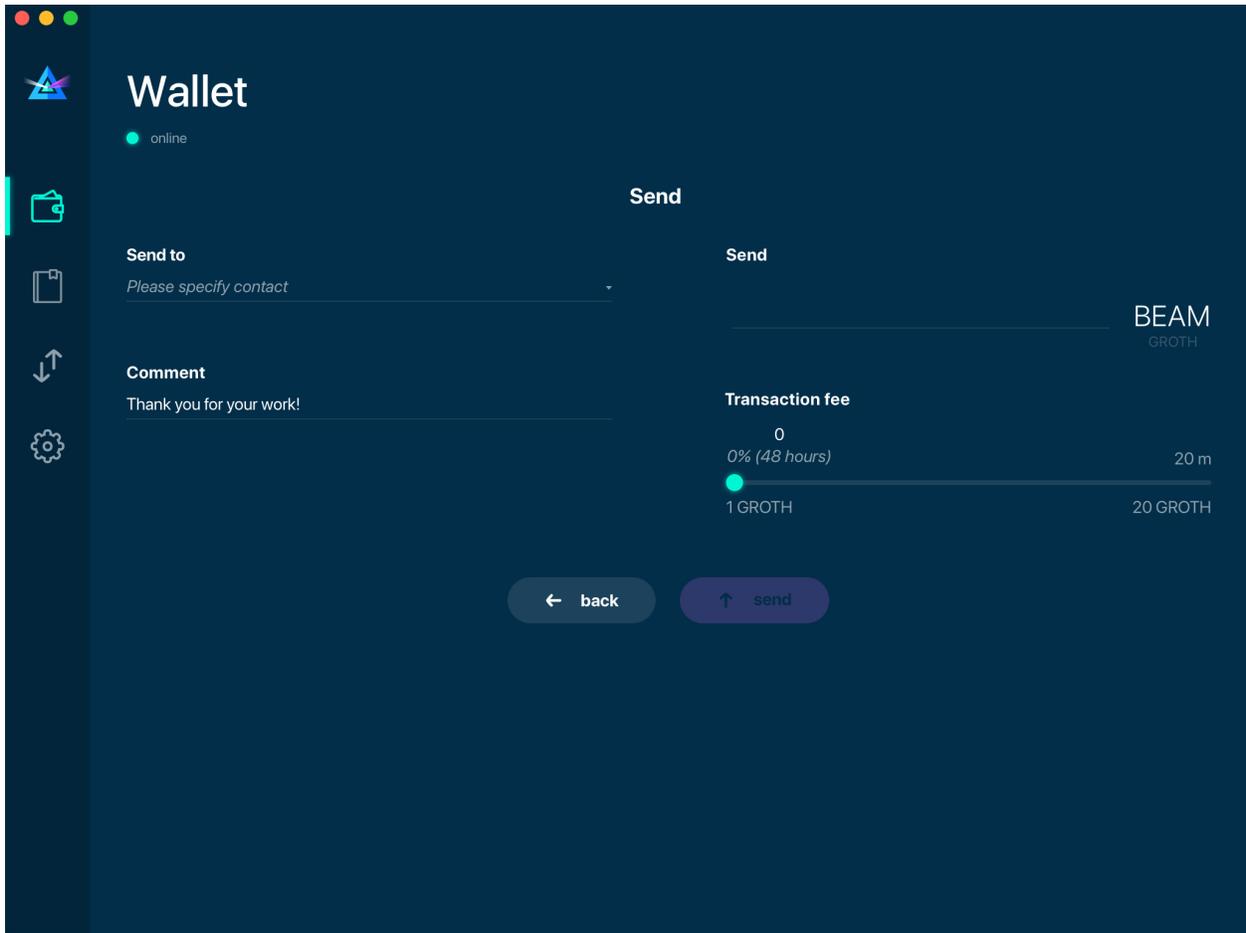
You can see the remaining amount of BEAM in your wallet and the change that will be received after the transaction.

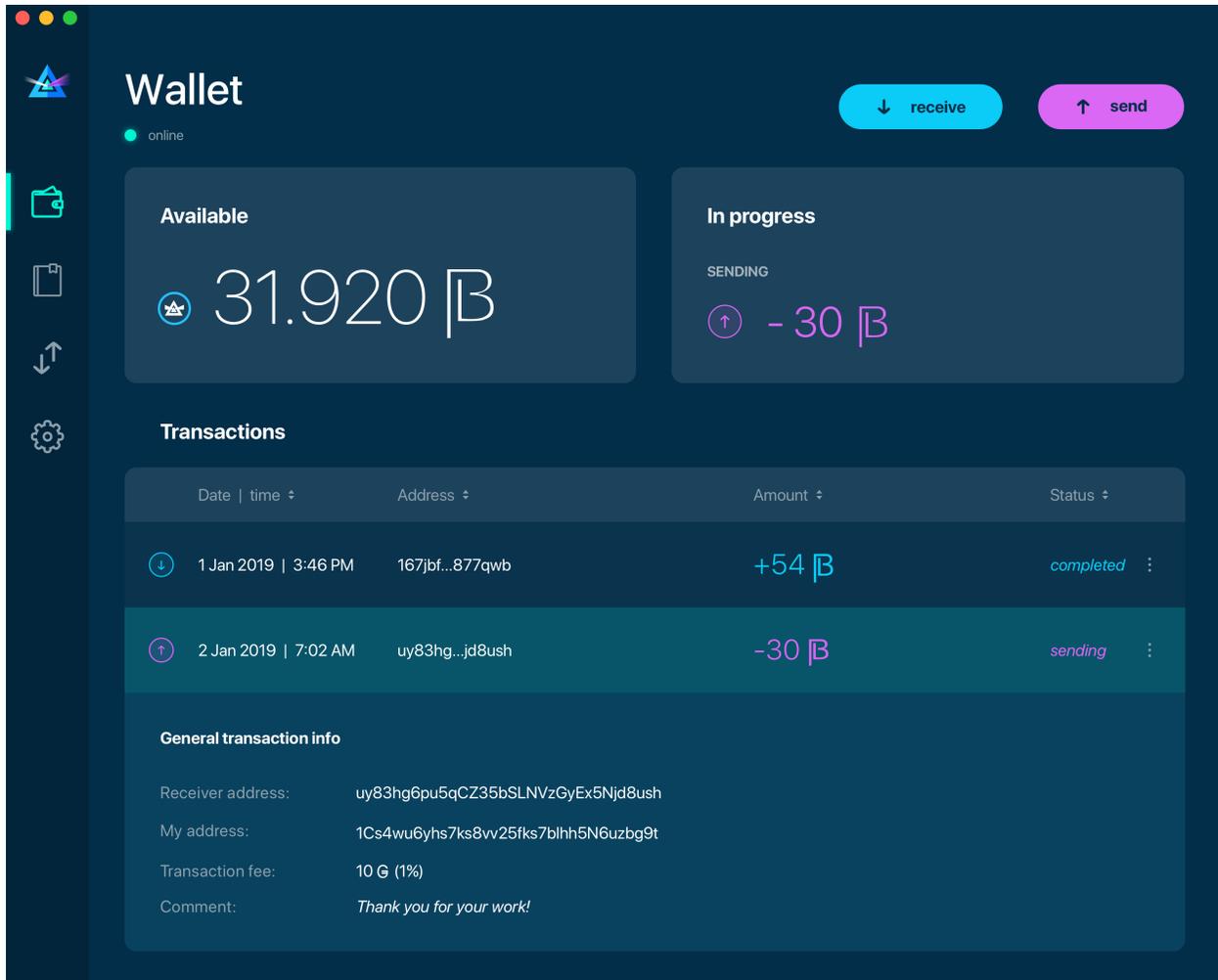
After you click 'Send' you will see a confirmation with the most important transaction details:

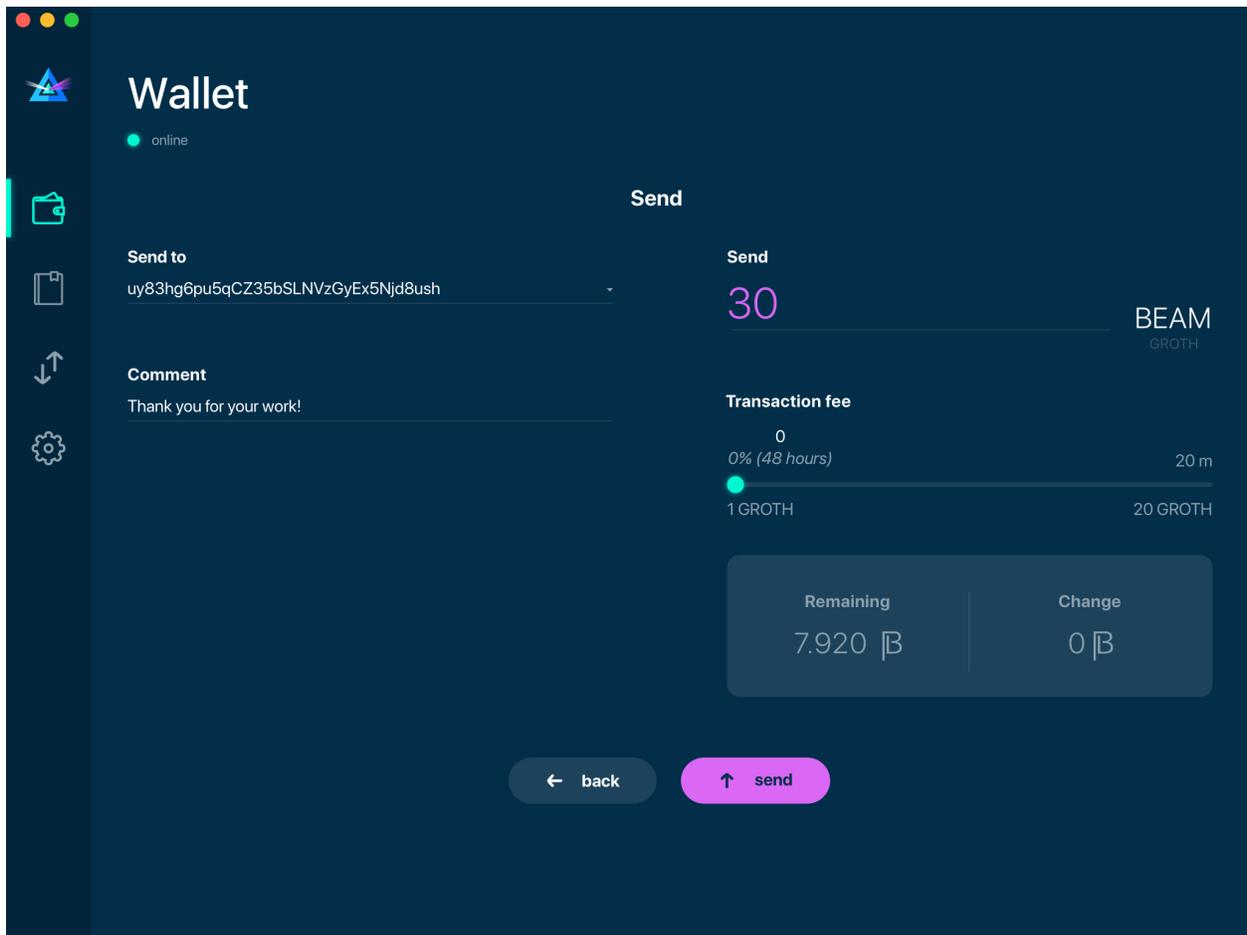
5.1.22 Completing the transaction

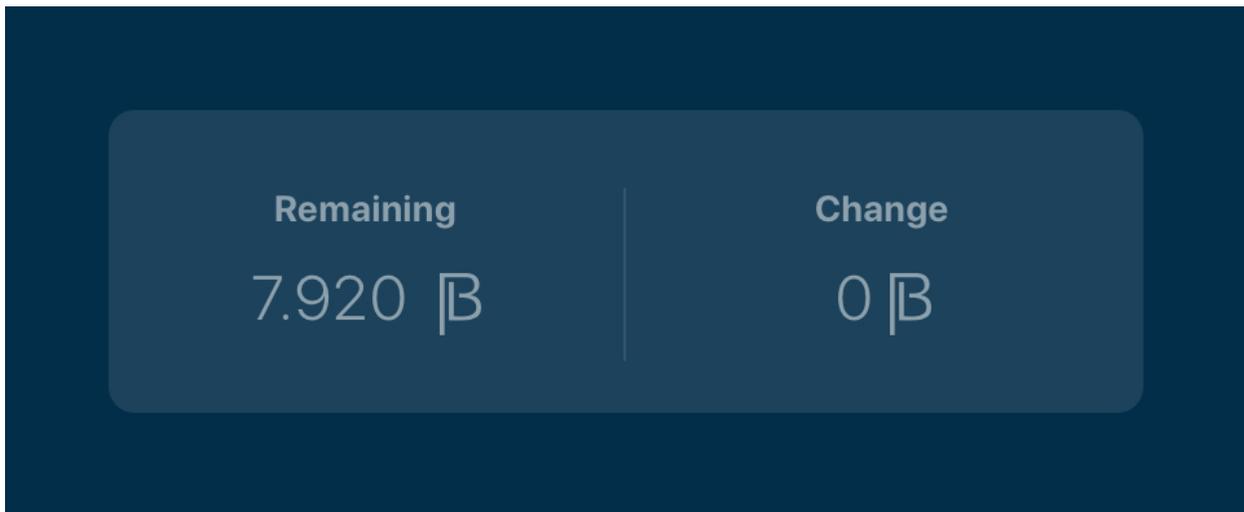
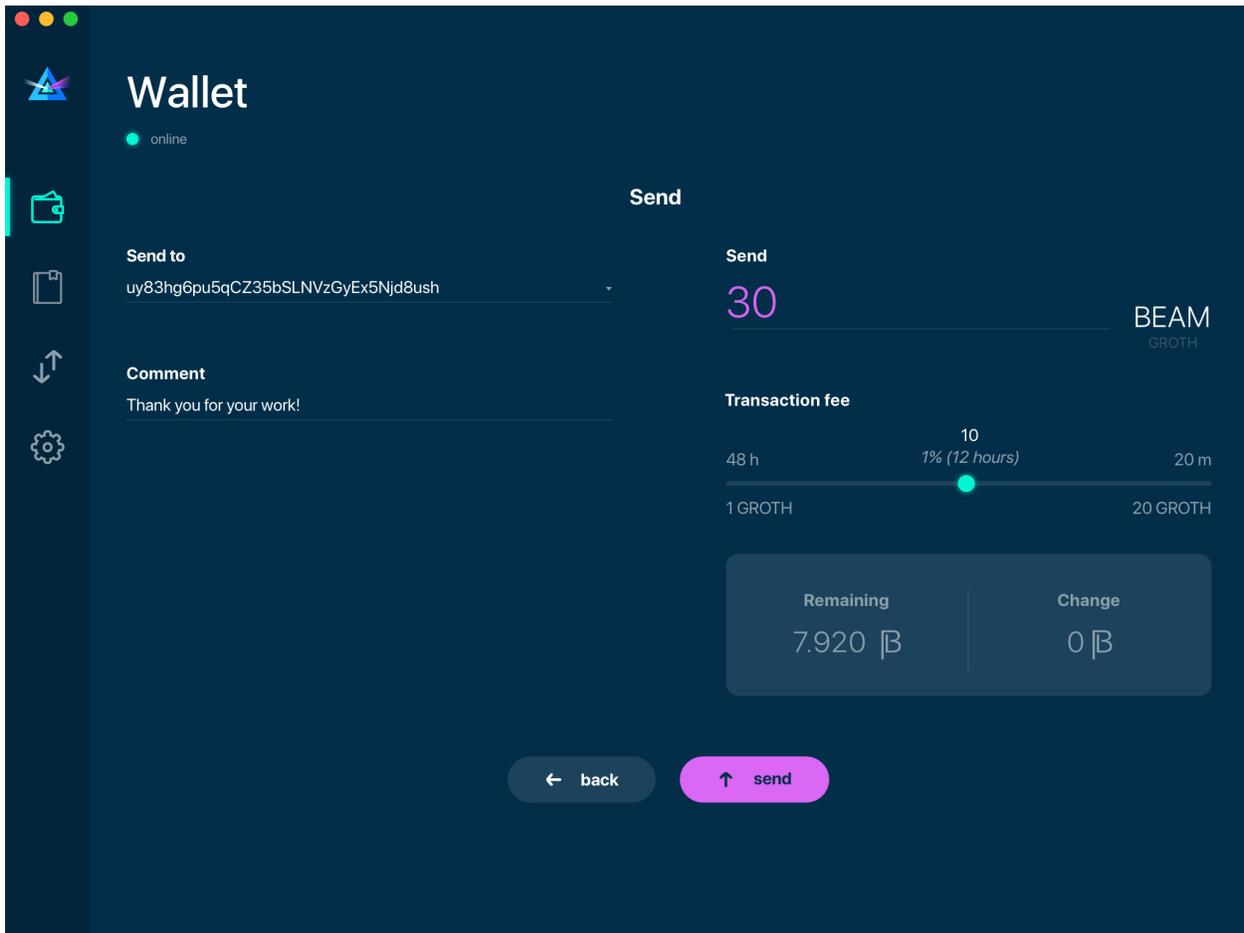
Once you confirm, the transaction is sent to the Receiver's wallet. If Receiver's wallet is currently offline or if the network is loaded, you might see the transaction appear 'In Progress' on your transaction list. Once the transaction is complete, it will be sent to the nodes and shown as 'Confirming'.

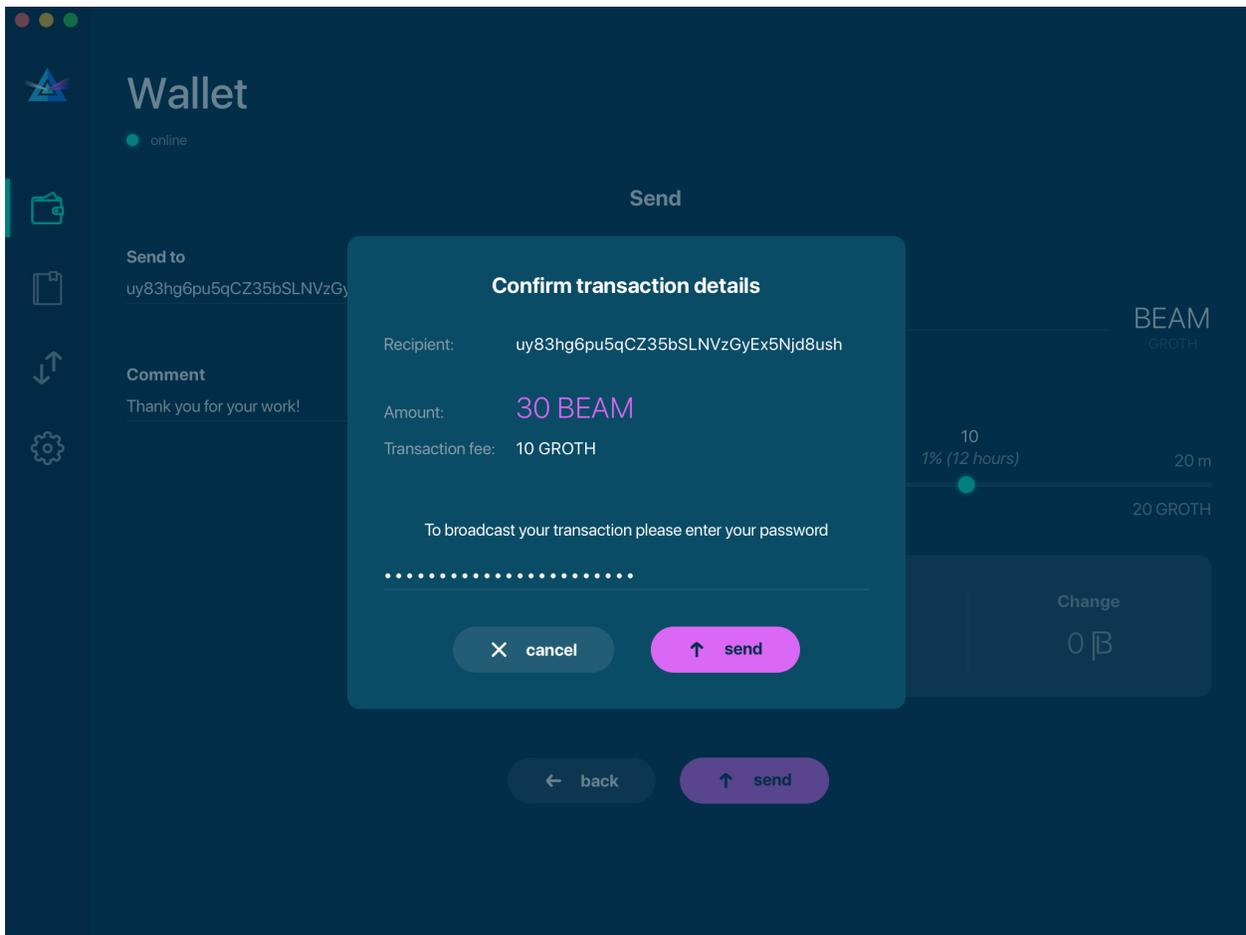




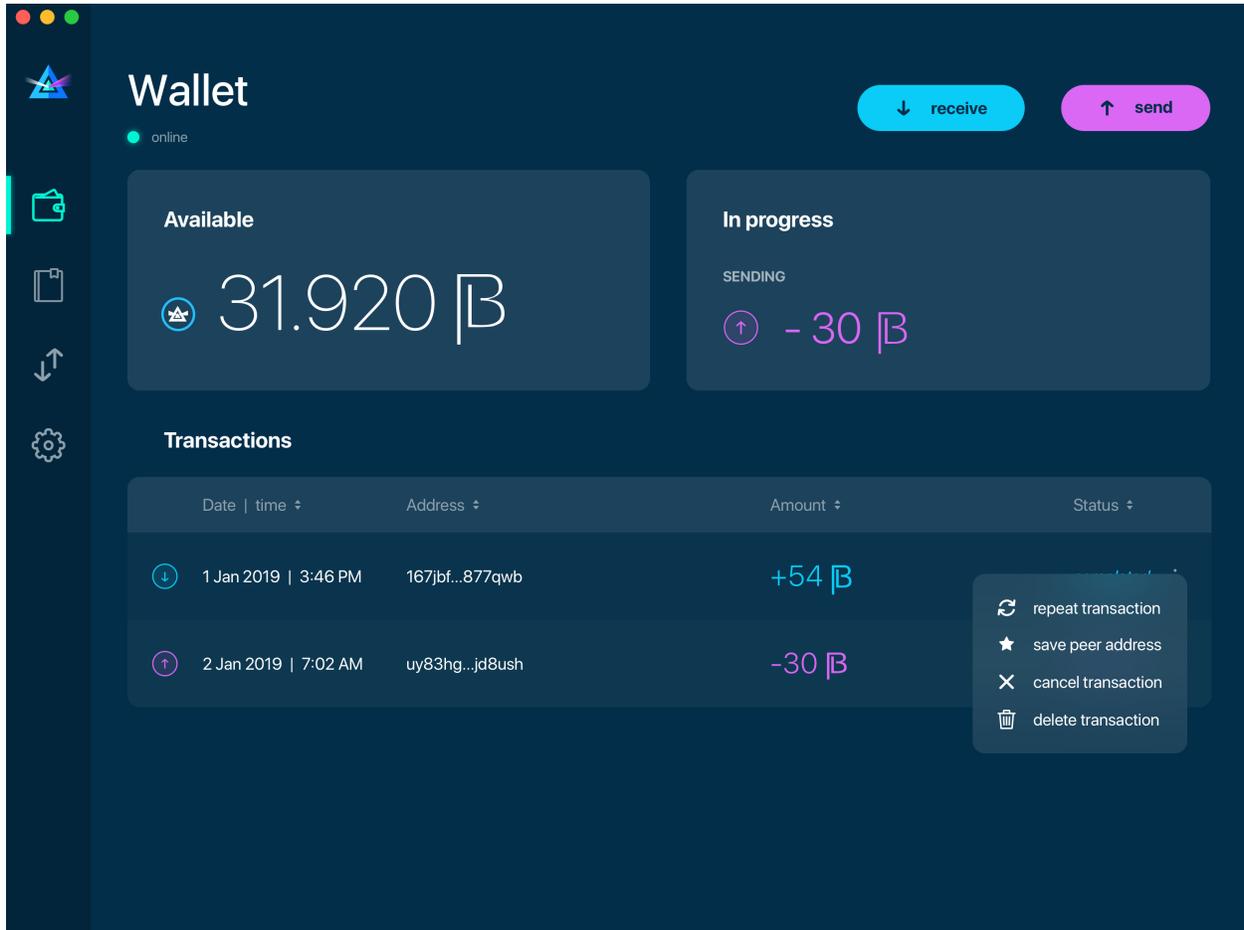








Note: While a transaction is in ‘In Progress’ you can cancel it by clicking on the dropdown to the right of the transaction row and then select ‘Cancel’. The other party will receive notification that the transaction was either ‘Canceled’ or ‘Failed,’ and funds plus fee that were allocated for this transaction will become available again. It is not possible to cancel a transaction in ‘Confirming’ or ‘Completed’ states.



Warning: If your transaction appears as ‘In Progress’ for a long time, it means the Receiver is not online.

Attention: If the transaction was not sent to the nodes, for any reason, it will expire after 1440 blocks, or roughly 24 hours. This is done to avoid a situation in which UTXO is locked forever.

5.1.23 Restoring funds

This process allows you to restore your funds directly from the blockchain. It is useful in the scenarios such as:

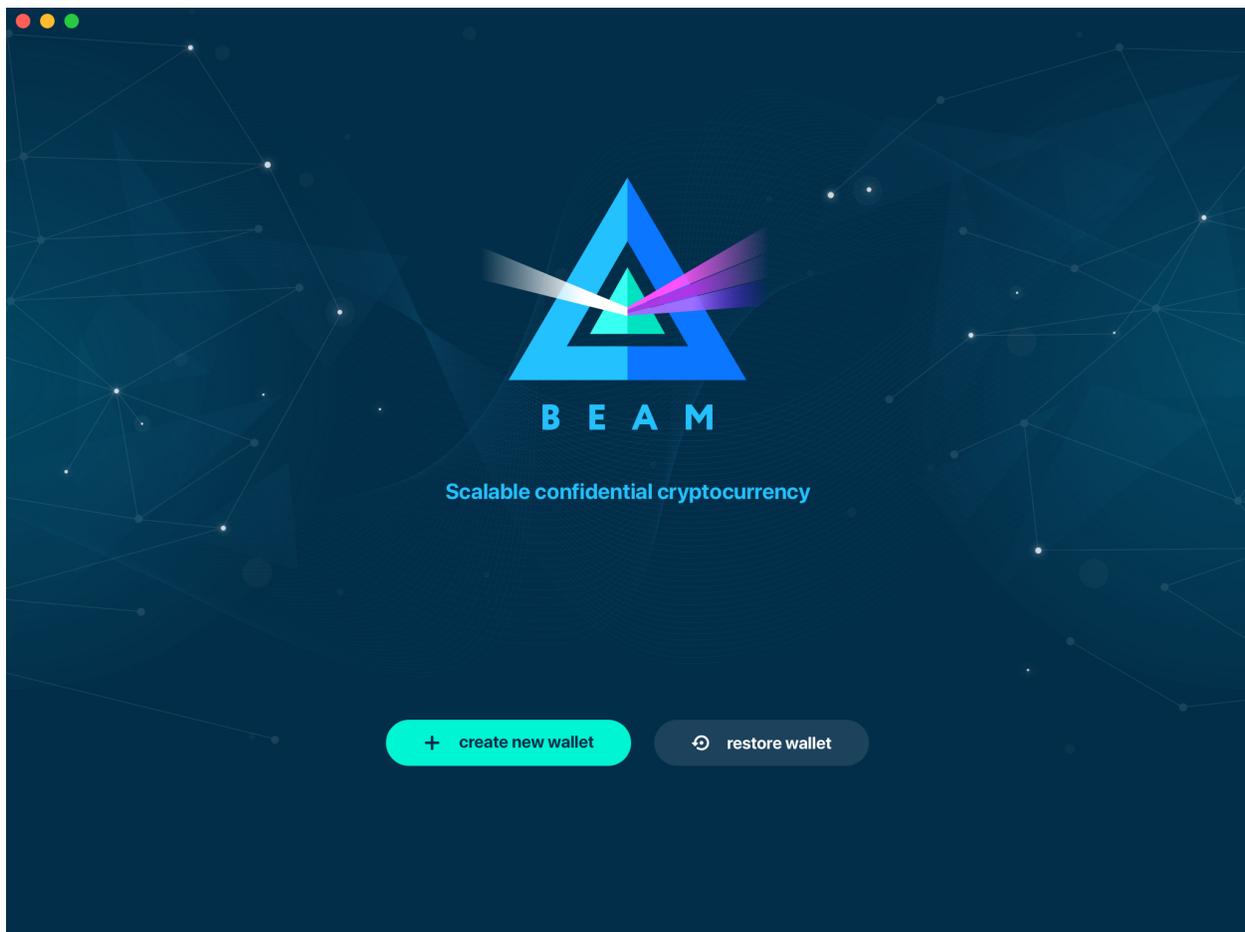
- You’ve got a new device and would like to use your wallet on it
- You forgot your local password and can’t access your funds

Attention: With Beam, only funds are stored on the blockchain. Everything else, such as your active addresses, contacts or transaction history can't be restored.

Note: Very soon exporting transactions history for backup and bookkeeping purposes will be implemented.

Before restoring funds on a machine the wallet was already installed, manually remove the wallet database file as described in *Files and Locations*. No action is required on a new machine.

Start the BEAM desktop wallet app and press 'Restore wallet' button.

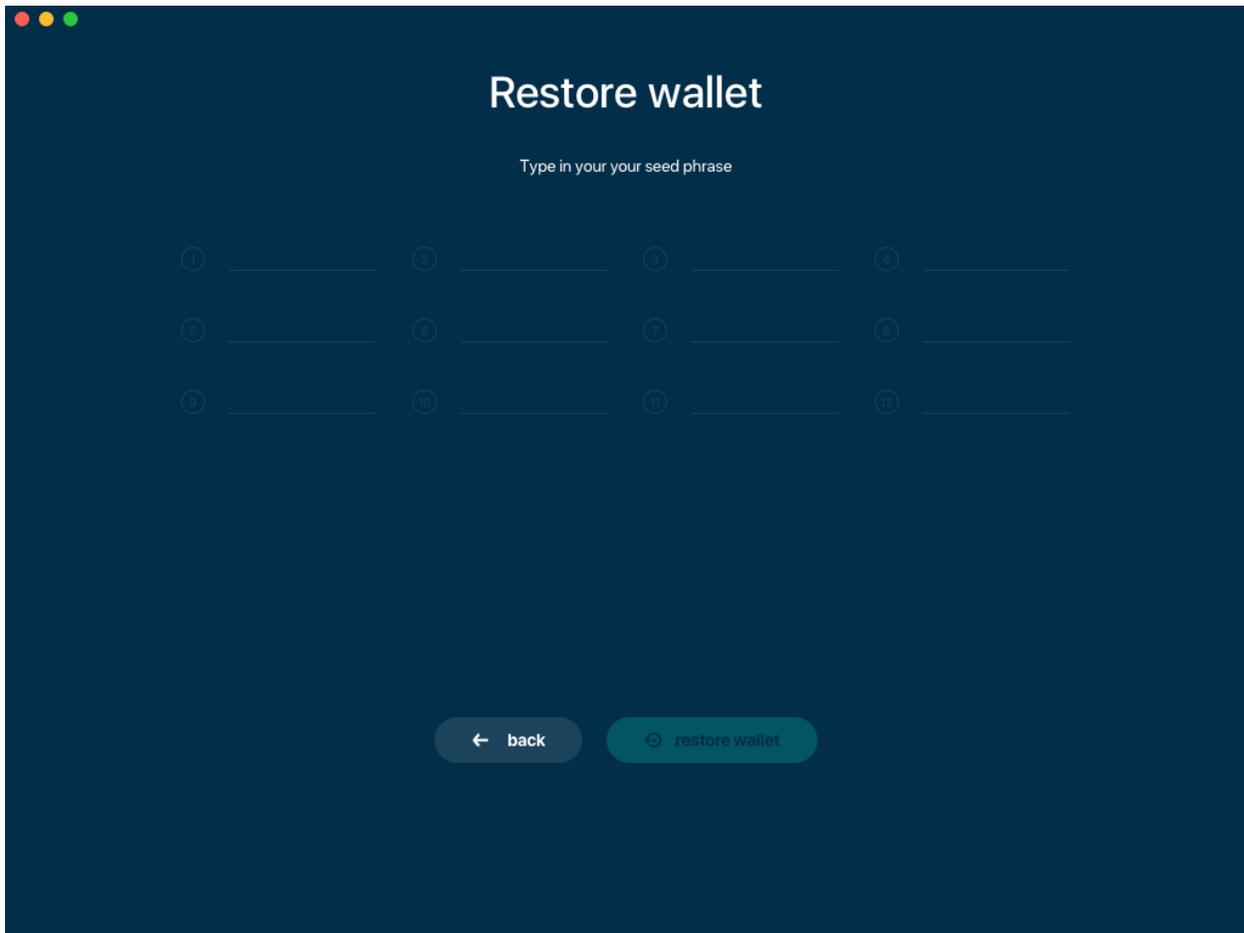


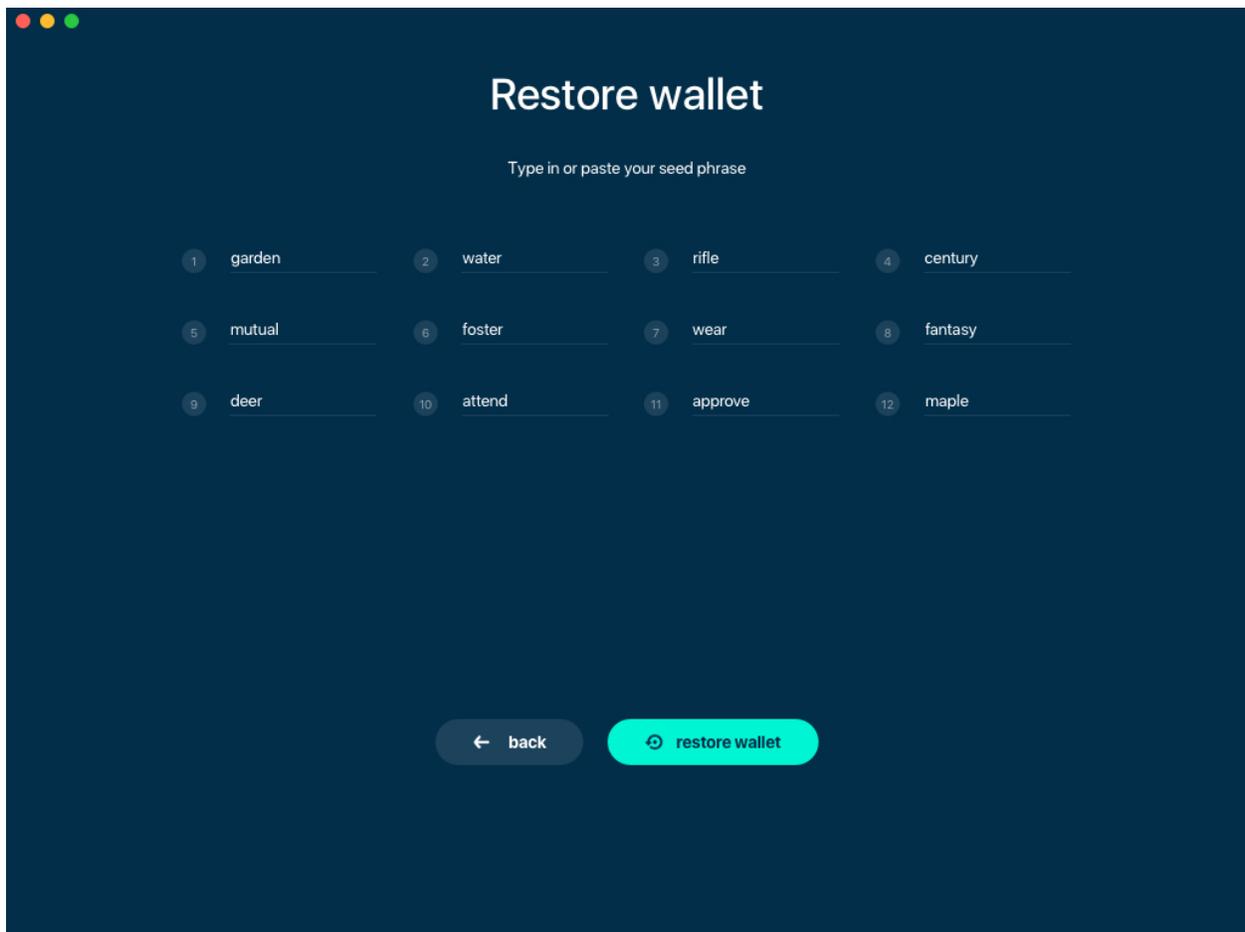
You will be asked to enter your seed phrase. Time to get the phrase out from your safe locker and type the words in.

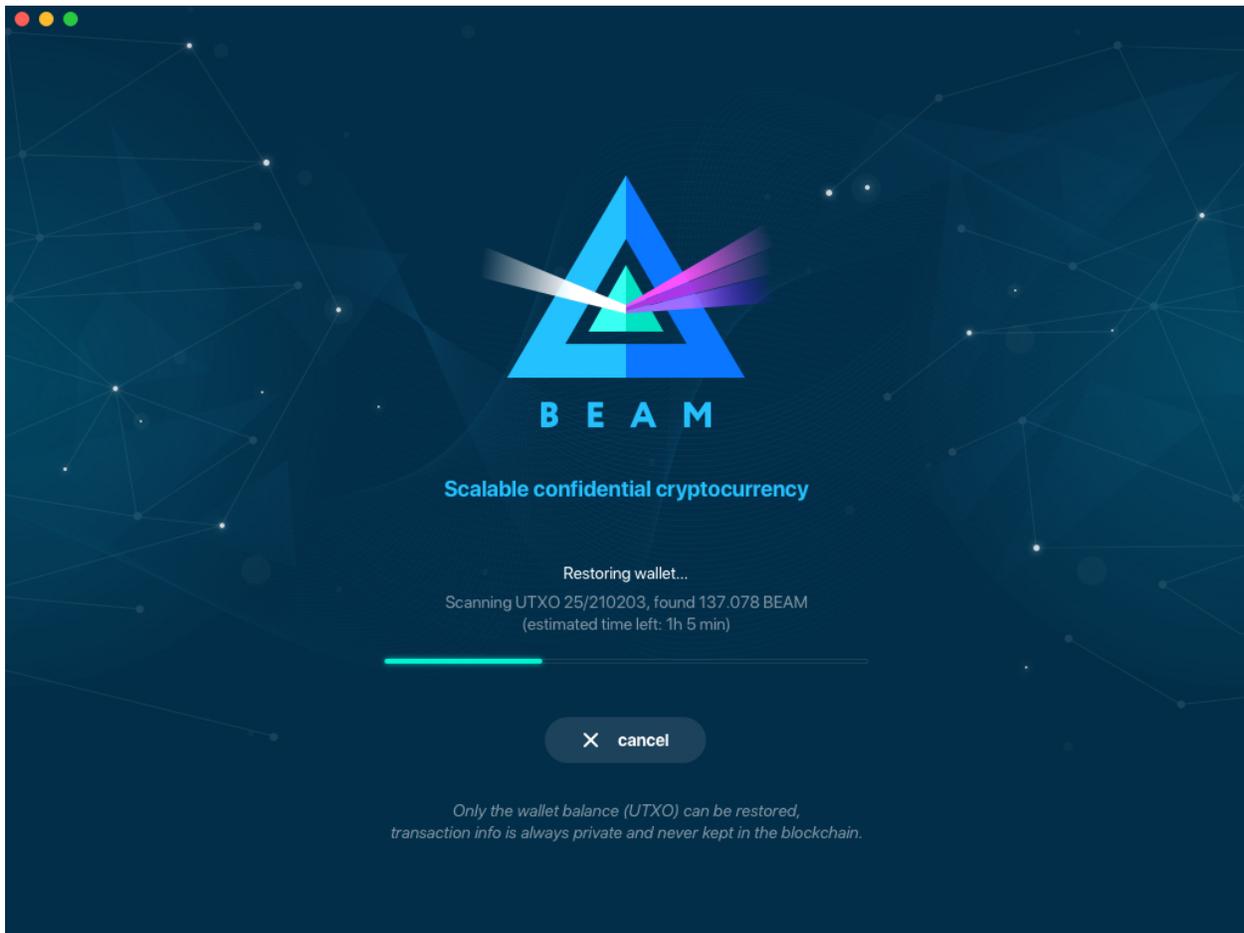
Warning: If a wrong word was typed or an existing word was misspelled, your funds will not be restored successfully. Example: 'litt_el_' instead of 'lit_le_.'

Did you checked your spelling? Once you are sure, click 'Restore wallet.'

Upon completion, you'll see the main screen of the wallet with your restored funds.







Note: Please be patient, restoring funds is a thorough and time consuming operation.

Attention: If the available balance is zero, it means that one or more words from your seed phrase weren't typed correctly or are wrong.

5.1.24 Address

Let's define the meaning of address in the BEAM ecosystem: BEAM is always sent from one address to another. Both sending and receiving addresses are alphanumeric tokens that uniquely define the transaction endpoints.

A person can create as many addresses as required. The address creation process is explained in *Receiving BEAM*.

Example: There are two wallets: one belongs to you and the other belongs to Alice. You have created one address to receive money from Alice and another address to send money to Alice. Since you can generate multiple addresses, Alice will never know that she's receiving money from the same person that she is sending the money to (unless you want her to know).

Example: There are three wallets: one belongs to you, one belongs to Alice and another belongs to Bob. You have created one address to receive money from Alice and another address to receive money from Bob. Alice and Bob will never know that they are sending money to the same person.

Same address can be used for sending and receiving money.

Example: you have created an address to send money to Alice. Alice can see the address the money came from and can send money to the address back to you.

Attention: For ultimate privacy, it is advised to have a **dedicated address for every transaction** (ie. for both Sending or Receiving).

Attention: Although not recommended, an address can be reused until it had reached its expiration (24 hours since when it was generated).

Example: Imagine you've created the address with expiration interval of 24 hours and immediately sent it to Alice. In the next 24 hours, Alice will be able to send BEAM to you as many times as she likes, reusing the same address of yours.

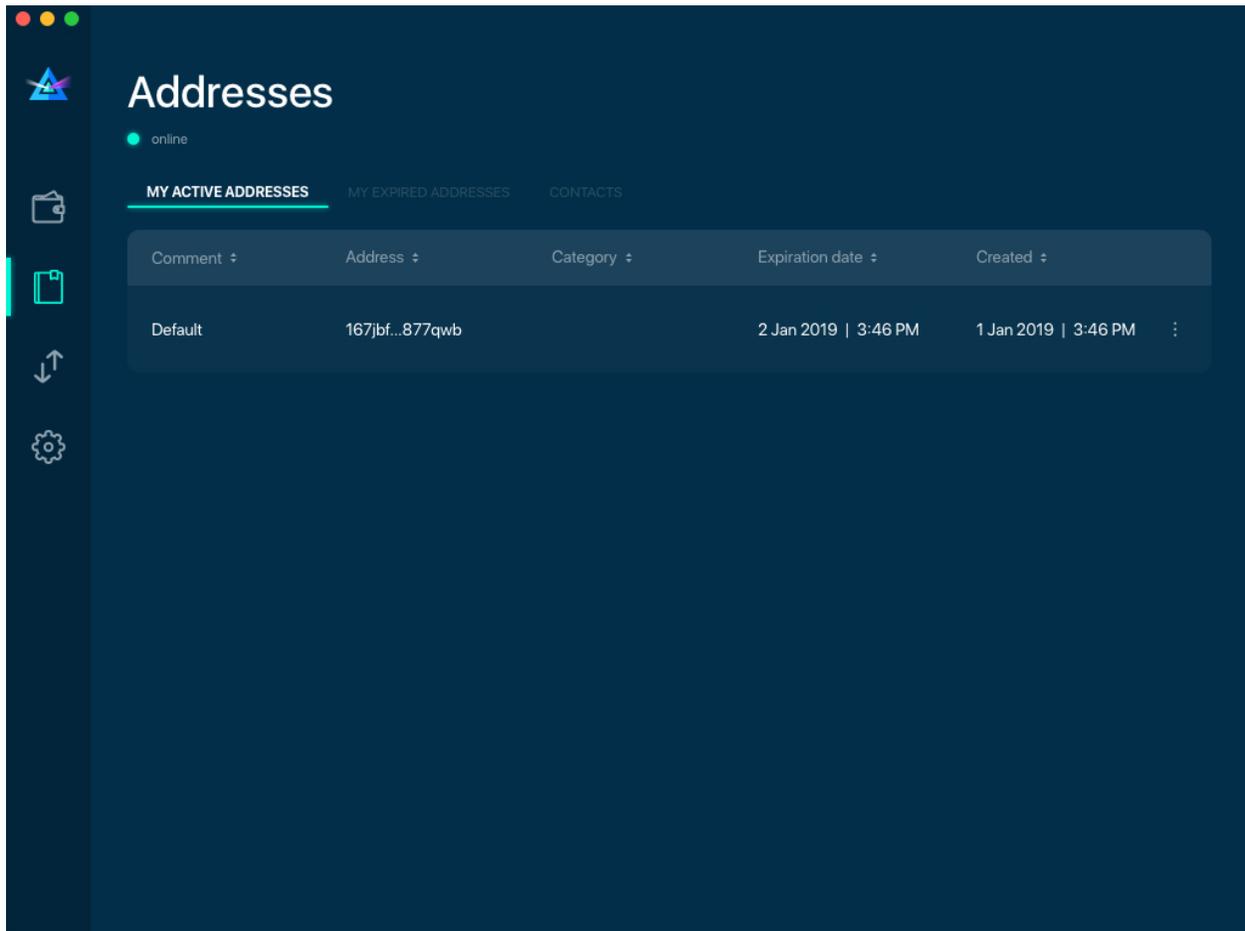
Warning: Reusing same addresses, created with longer expiration interval, can be convenient yet the tradeoff of decreased privacy should be kept in mind.

5.1.25 Address screen

The address screen lists all your incoming and outgoing transactions. It includes the address, comments, date address was created and date address expired of each transaction. All the data in this screen is only stored locally in your wallet and is not related to the blockchain in any way.

Note: A new address is generated for each transaction. Yes, you heard that right! That address can be seen in your active or expired addresses list.

Upon Beam Wallet desktop app installation, a single address is created by default. The address has a default expiration time of 24 hours. You can always create a new one by going to the Receive screen. You can see all your active addresses in the ‘My Active Addresses’ tab.



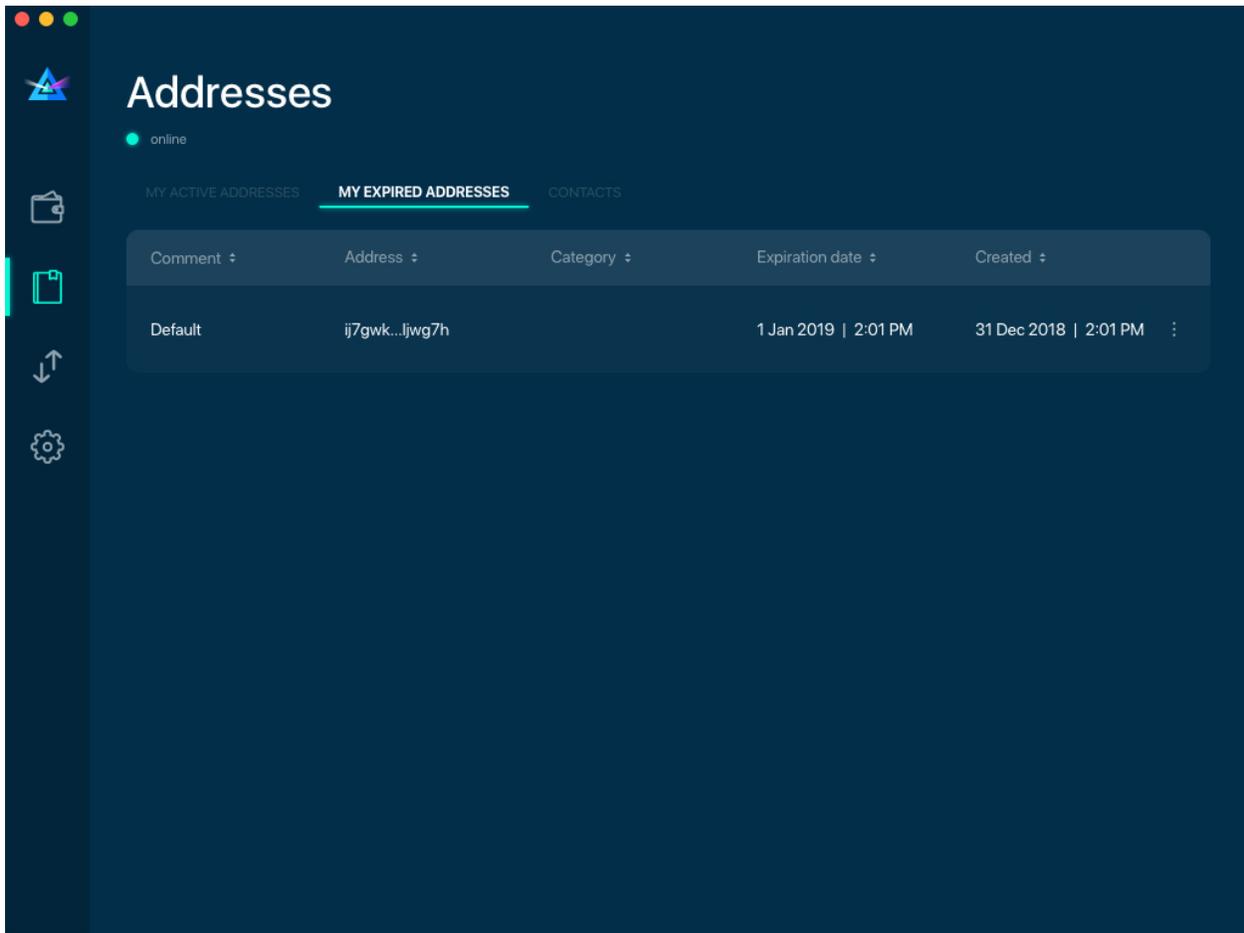
Each address has a default expiration time of 24 hours. An expired address becomes inactive and you won’t be able to use it again. You can see the expired addresses listed under ‘My Expired Addresses.’

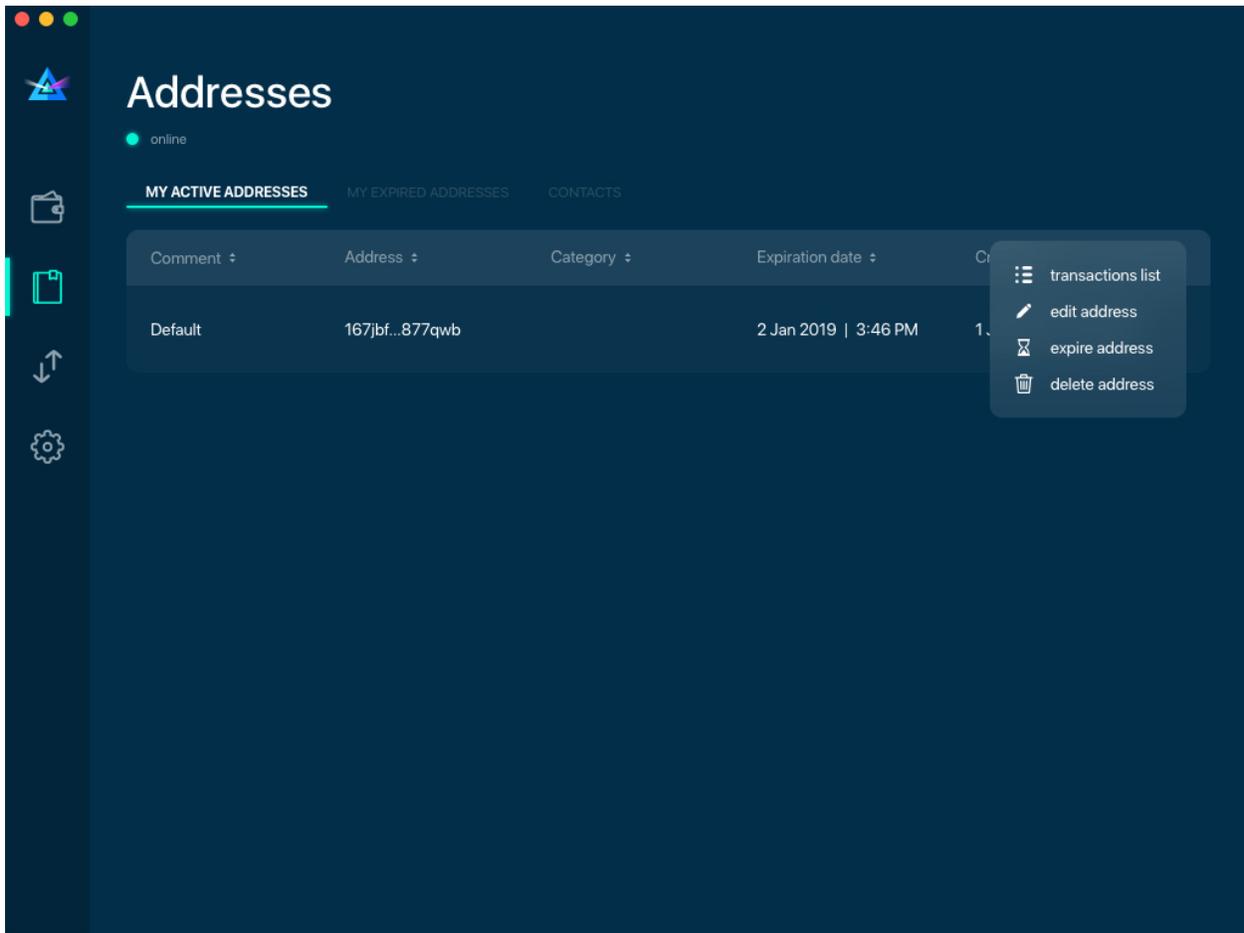
When you’ll click the three dots located to the right of any address, the menu with additional address actions will open. You can manually edit or delete any address choosing ‘Edit’ or ‘Delete’ from the menu.

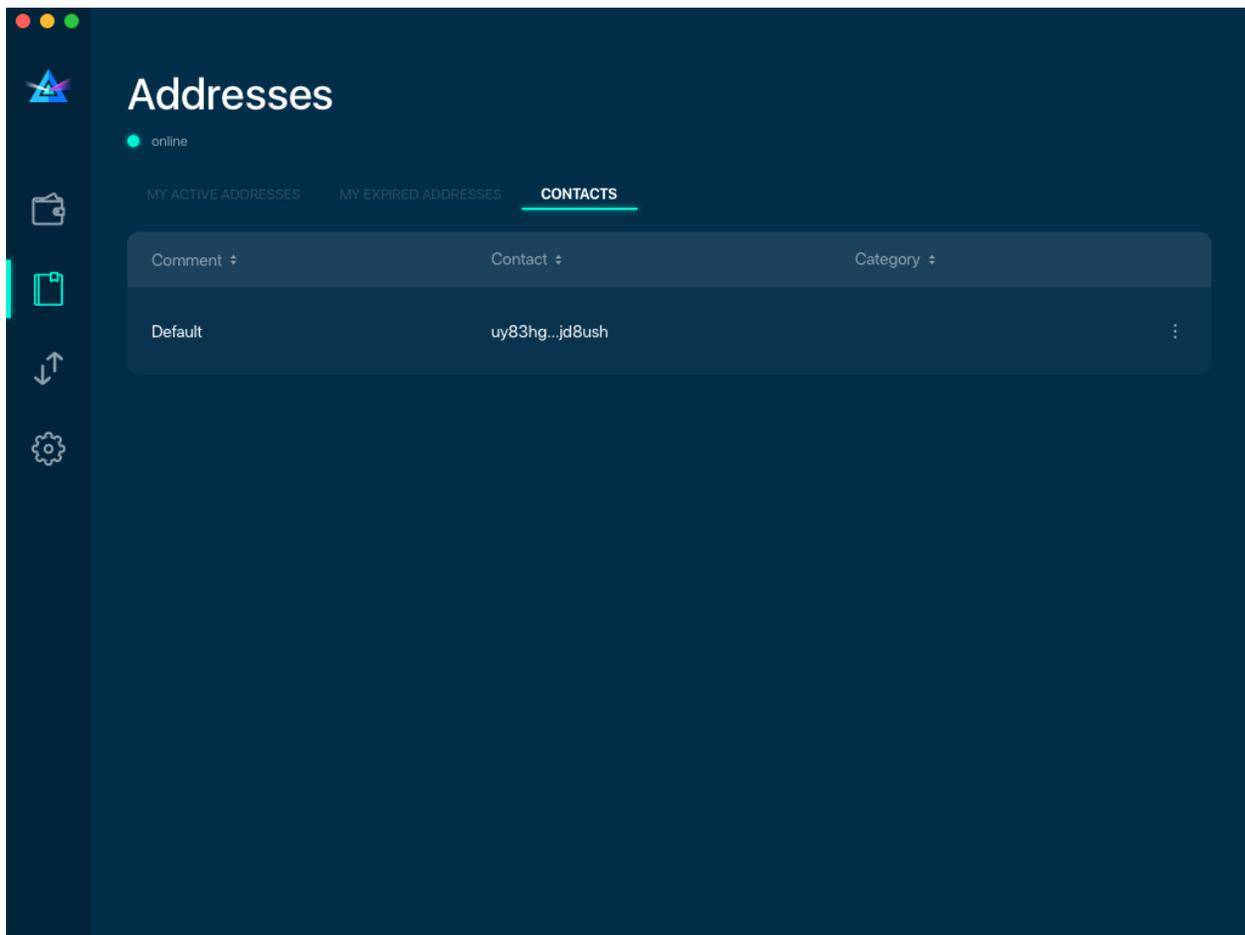
In the ‘Contacts’ tab, you can see every address that sent you BEAM or to which you’ve sent BEAM to.

5.1.26 UTXO

UTXO (Unspent Transaction (TX) Output) is like a banknote of a specific amount. Simply said, if BEAM is the currency, any UTXO can be considered a ‘bill’. You can have multiple ‘bills’ in your wallet at the same time.







5.1.27 UTXO screen

On the technical level, in Beam, like in most other cryptocurrencies, your balance emerges as a result of multiple incoming and outgoing transactions. Each transaction uses some existing inputs and creates new outputs. All the outputs controlled by the wallet are shown in the UTXO screen.

The screenshot shows the UTXO screen with the following data:

Amount	Height	Maturity	Status	Type
23.99999 β	35000	35002	available	Change
3.99999 β	34929	34931	available	Change
9 β	30718	30690	spent	Regular
40 β	30688	30690	available	Regular
5 β	30688	30690	spent	Regular
36 β	30683	30685	spent	Regular

Additional information from the screenshot:

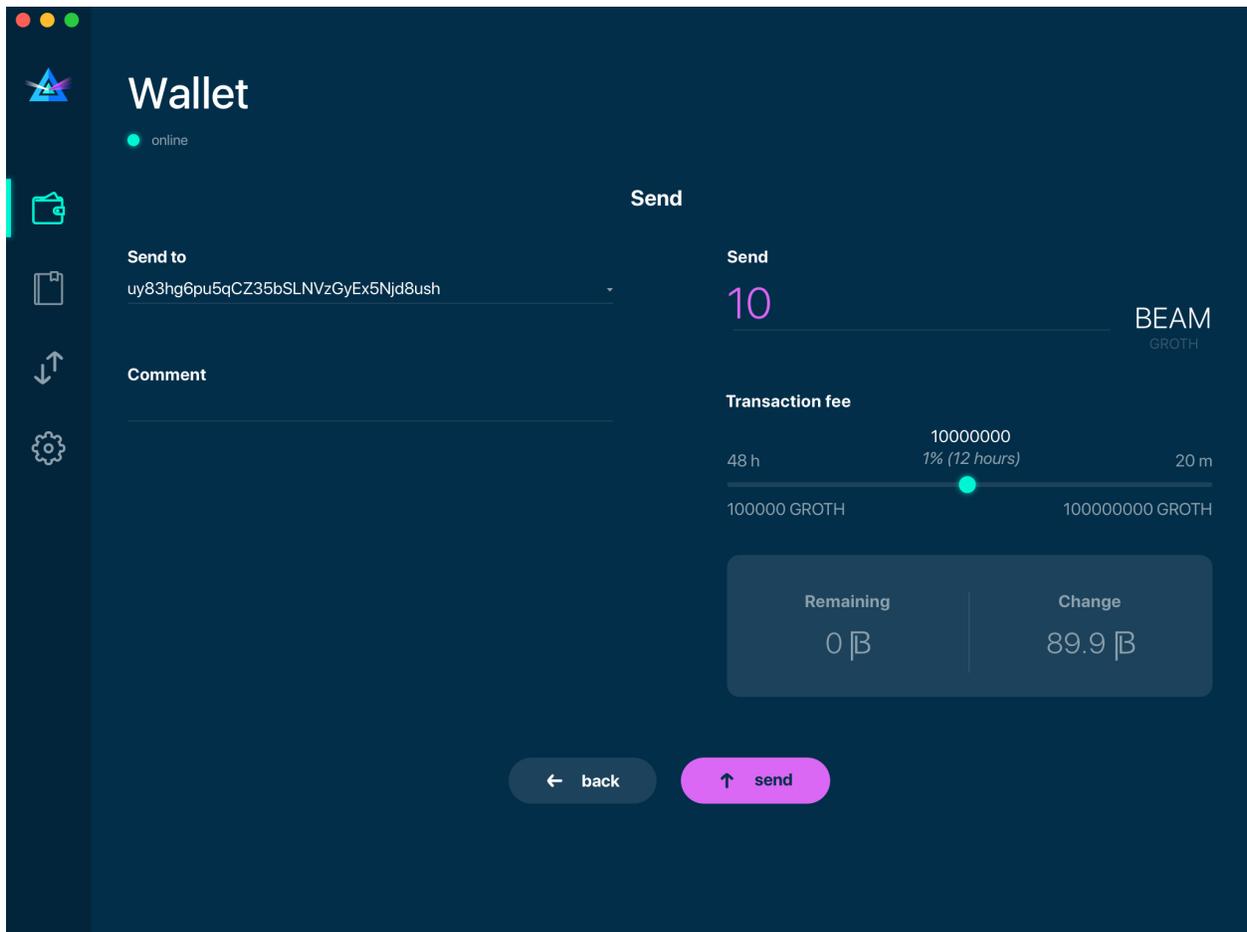
- Height: 35003
- Hash: 880890bf7646e5965ff8
- Status: online

The type of UTXO can be:

- **Regular** - UTXO received as a result of a transaction. It is immediately available for spending
- **Change** - UTXO received as a result of change from a transaction. It is immediately available for spending
- **Transaction fee** - Fees received as a result of mining a block which contain transactions
- **Coinbase** - UTXO you have mined. It has maturity of 3 hours (240 blocks) and will not be immediately seen in Available tab

5.1.28 UTXO in sending BEAM screen (explained by example)

Assume that you have 100 BEAM in a single ‘bill’. So, if you want to send to Alice 10 BEAM, your single ‘bill’ will be split into one ‘bill’ of 10 BEAM to send and another ‘bill’ of 90 BEAM to remain in your wallet, right? Well, almost: we also have to consider the transaction fee. Let’s say the transaction fee is 10M GROTH (0.1 BEAM), so in this case the ‘bill’ will be split into 3 ‘bills’ (10 BEAM to send, 89.9 BEAM to remain in your wallet and 0.1 BEAM to pay fee).

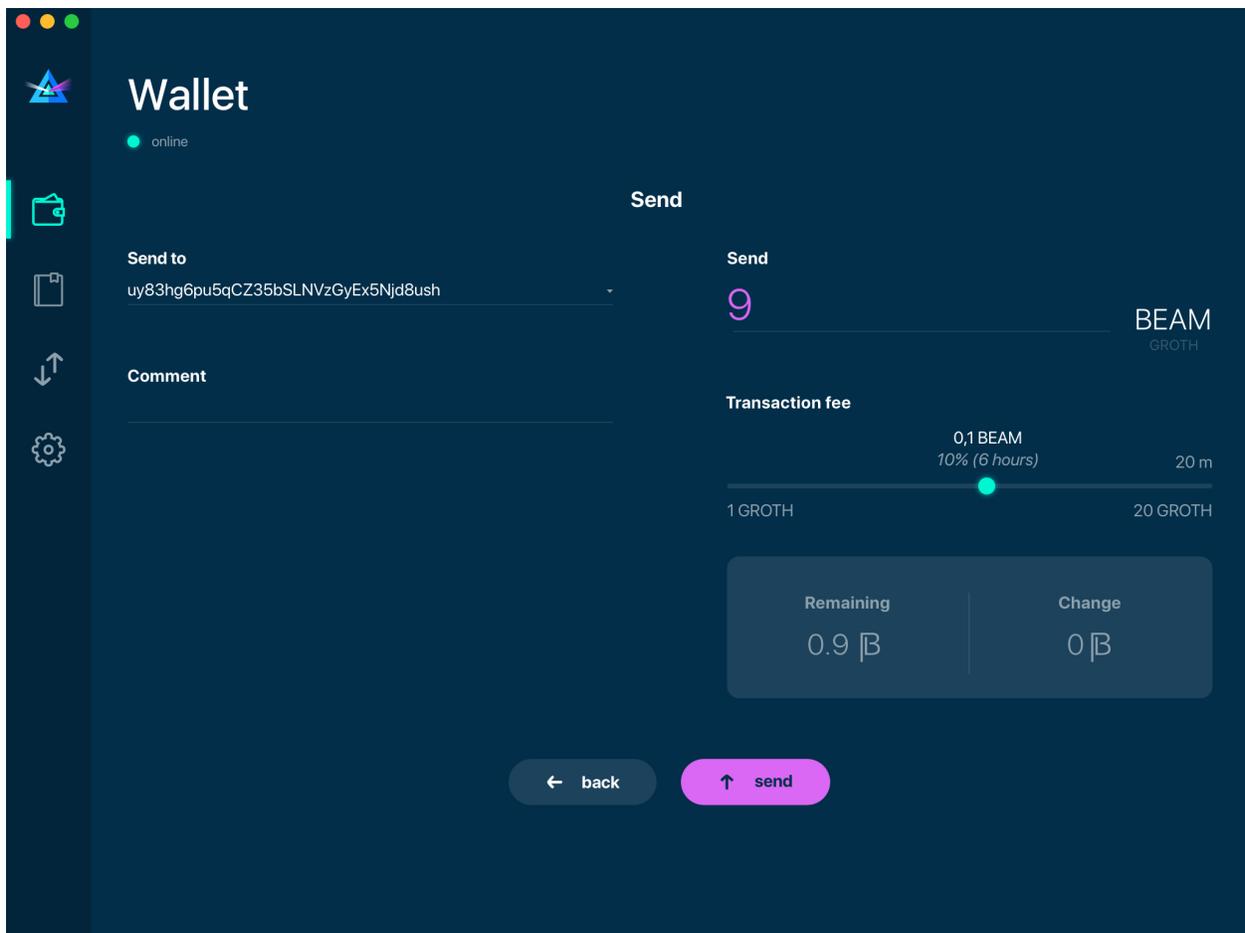


5.1.29 Why UTXO can be locked

Important: Beam Wallet app automatically selects which UTXO will be used for the transaction by trying to minimize the change you should receive as a result. This is important to understand since until the transaction is complete, the UTXOs used in the transaction cannot be used for any other transaction and do not appear in the list of ‘Available’ funds.

Example: you have 100 BEAM in two ‘bills’ (UTXO): 90 BEAM and 10 BEAM. You want to send Alice 9 BEAM. The wallet will automatically select the 10 Beam UTXO and create a transaction with 9 BEAM sent, 0.9 BEAM to remain in your wallet and 0.1 BEAM to pay fee.

This 10 BEAM UTXO will be locked until the 9 BEAM transaction completes. If Alice is currently offline, it might take a time during which you will not be able to send BEAM to anyone else. You can, of course, cancel the transaction and resend when Alice comes online.

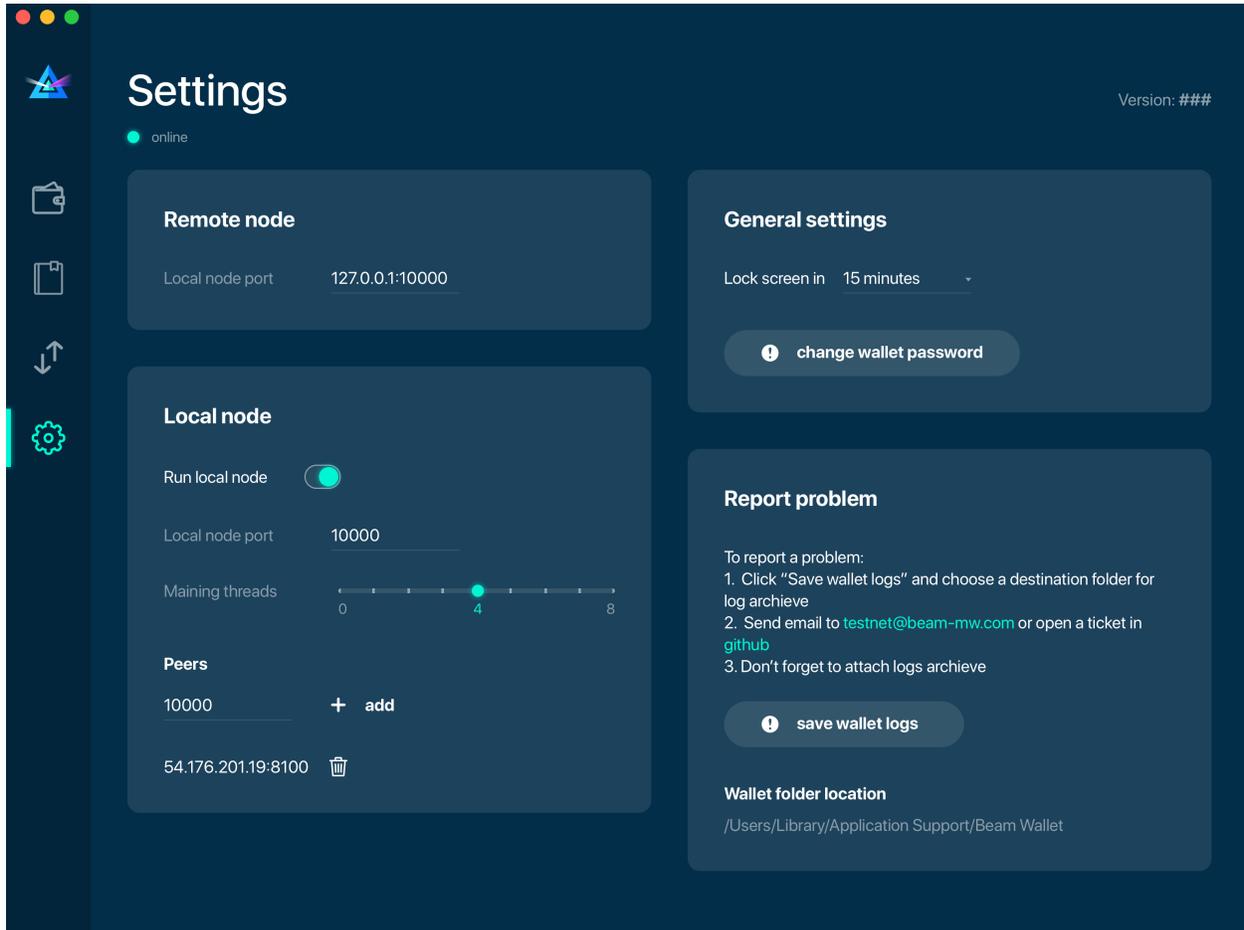


5.1.30 You can split UTXO by yourself

One thing you can do is to split UTXO by sending a transaction to yourself (using your own active address). You may want to do this in the case your UTXO is too large, and you don't want it all locked during a transaction.

Attention: You will pay a fee for this transaction.

5.1.31 Settings screen



Note: At the top right corner of the screen the version is displayed. It is always important to specify the version when asking for support or reporting issues.

For integrated and external nodes settings see *Choosing the node connectivity mode*. When running integrated node you should specify the port on which the node will be listening on and the list of node peers.

The ‘General settings’ section allows you to change your wallet password and lock screen time. The Wallet will automatically lock to protect the funds from accidental unauthorized access to an active wallet in the set time you choose.

The ‘Report problem’ section allows you to create an archive of wallet logs and explains how to report an issue. It also shows the current location of the wallet files. See more details about reporting issues and getting support in the *Reporting Issues and Getting Support* and *Desktop Wallet Troubleshooting* sections.

5.2 Desktop Wallet troubleshooting

5.2.1 Where are the wallet files located?

When Beam Wallet desktop app is installed, the wallet data files are stored separately from the binaries. The locations of all the files are described here: [:ref:'Files and Locations'](#)

5.2.2 Why is my transaction 'In Progress' for so long?

Both Sender and Receiver Wallets need to be online to complete a transaction. All active addresses expires after 24 hours since creation (unless specified otherwise). If Sender / Receiver both do not come online within 12 hours the transaction will be canceled automatically.

5.2.3 Why is my available balance lower than expected while I'm sending BEAM?

UTXO can be locked during active outgoing transaction. The locked amount is displayed as a change in 'Sending screen'. The change will become spendable when the transaction expires or completes.

5.2.4 I've transferred BEAM to someone, but the transaction is stuck 'In Progress'

The most common reasons are listed here:

- Addresses expire in 12 hours by default. Check to see if it's expired in the [:ref:'Expired addresses'](#) section on the [:ref:'Address screen'](#).
- Address might be misspelled. Check to see if the address is complete. Are there any letters or number missing or misspelled?
- Receiver has not come online.
- Receiver's Wallet was restored between the time the address was created and the time it was sent.

5.2.5 I've forgot the local password for my wallet

See [:ref:'Restoring funds'](#)

5.2.6 I've restored the wallet but I can't see my transaction list and/or my active addresses

As explained in *Restoring funds*, only your available balance (i.e. your UTXO) is kept on the blockchain, hence that's all that can be restored.

5.2.7 I've restored the wallet but my balance is zero

One or more the of the words is wrong or misspelled. Triple-check that all the words from the seed phrase are typed in correctly. You will need to repeat the [:ref:'Restoring funds'](#) procedure.

5.2.8 I've restored the wallet using my seed phrase - can someone still send me money to the addresses created in the previous wallet?

When a wallet is restored, *only the balance (UTXO) is restored*. Addresses (active and expired), contacts and transaction history are only stored locally, so they can't be restored from the blockchain. Each wallet is aware of only the active and expired addresses it displays. Therefore, all transactions sent to the addresses no wallet is aware of anymore will fail by timeout and the funds will be automatically released in Sender's wallet.

5.2.9 I've forgot my password

If you lost your password and cannot get into your wallet, you will have to remove `wallet.db` file and to *Restore funds* using your seed phrase to create a new password.

5.2.10 Why is the seed phrase the only thing connecting me to my funds?

To ensure the utmost privacy, the only information we can use to link you to your wallet is your seed phrase. So, if you lose it we cannot recover it for you.

5.2.11 I've lost my seed phrase

By design, the only way to access your funds (UTXO) is to have the seed phrase. If you still have access to your wallet, create another wallet with new seed phrase on another device and transfer funds to there. Any solution that would allow you to access your funds without the seed phrase would severely compromise the privacy of BEAM. Therefore, in case you don't have any active access to your funds there is nothing to do (the funds will be stored in the blockchain forever and no one will be able to access or spend them).

5.2.12 I've copied my `wallet.db` file to the new machine and I'd like to run wallets on both new and old machines simultaneously

At the current implementation each `wallet.db` file should be managed by only a single wallet instance. Any case involving manual transfer of the wallet database **is not supported**.

5.2.13 My question is not answered here

See *Reporting issues and getting support*

5.3 Command Line Wallet User Guide

The purpose of this document is to describe the process of setting up Beam node and command line wallet.

Attention: Beam blockchain does not store transaction history and SBBS addresses. These are only stored in local database inside the wallet data folder.

Please follow the guidelines below to avoid problems with sending or receiving Beam transactions.

1. DO NOT run several wallet processes on the same `wallet.db` file.
2. Do not do listen and send at the same time using CLI wallet

3. Do not copy the wallet.db to another machine and run another wallet simultaneously using the same wallet database
4. Do not run two different wallets with the same seed at the same time
5. SBBS messages sent between wallets expire after 12 hours. You have to connect within 12 hours of the transaction initiation to receive or send the funds.
6. SBBS Addresses by default expire after 24 hours. Always use 'never' expiring addresses with pools and exchanges to make sure you receive payments.

5.3.1 Getting Started

Beam software runs on all operating systems: Linux, Mac OS and Windows. Before you start please verify that your operating system is supported by reviewing the *Supported Platforms*.

All examples in this section are formatted for Linux / Mac platforms. If you are using Windows please substitute `./beam-wallet` with `beam-wallet.exe` and run your commands using Windows Command Prompt.

5.3.2 Creating new wallet

In order to create a new wallet run:

```
./beam-wallet init
```

You will be prompted to enter the Wallet Password, which is used to protect the wallet database

Warning: Choose strong password for the wallet and keep it secret

Anyone who knows your wallet password and can access your machine the wallet is stored on, will be able to spend **all your funds** and will have an access to all the metadata stored in the wallet, including transaction history!

Sample output for the `init` operation will look something like this:

```
$ ./beam-wallet init
I 2018-12-23.15:24:29.461 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.15:24:29.462 starting a wallet...
Enter password: *****
I 2018-12-23.15:24:32.524 Generating seed phrase...

Generated seed phrase:

    despair;evoke;airport;seven;cricket;menu;current;ankle;require;monkey;maple;
↪crawl;

    IMPORTANT

    Your seed phrase is the access key to all the cryptocurrencies in your wallet.
    Print or write down the phrase to keep it in a safe or in a locked vault.
    Without the phrase you will not be able to recover your money.

I 2018-12-23.15:24:32.728 wallet successfully created...
I 2018-12-23.15:24:32.750 New address generated:
```

(continues on next page)

(continued from previous page)

```
14a38140d8e66be9b8f1e8d770161fd33e35f7000053147b5a0f6a83178926b956
I 2018-12-23.15:24:32.750 label = default
```

The `Rules signature` is a hash of current node configuration which is used to determine compatibility between different versions of nodes and wallets.

Generated seed phrase is the *Seed Phrase*.

Warning: Copy the seed phrase to a secure location and keep it safe.

Seed phrase is the **most important secret** you need to keep to protect your funds. Anyone knowing the seed phrase will be able to control all your funds regardless of any other information. When generating new wallet each and every time, the safest scenario would be to make it on a secure air-gapped machine in a private environment and always keep the seed phrase in a secret and protected place.

The following line indicates that a new temporary *SBBS* address has been generated. This address is valid for the next 24 hours and can be used to receive coins. To generate new addresses see ‘Creating new receive address’ section below.

After wallet initialization is succeeded a `wallet.db` file is created in the same folder the wallet was run at. `wallet.db` is the wallet database file which is encrypted with the Wallet Password and contains the entire transaction history, keys and all the rest of the wallet metadata. If this file is deleted or lost, for any reason, you can always restore your funds using the Seed Phrase, however you will lose all transaction history and any additional metadata stored in the wallet database. To understand how to backup and restore the `wallet.db` file please check the *Backup and Restore* page.

In addition to the `wallet.db` file, you will see the `logs` folder. A new log file is created every time you run the CLI wallet. Please attach logs to any support request you might send. See *Reporting Issues and Getting Support* section, for more details.

5.3.3 Restore a wallet from a Seed Phrase

For all the restore procedures see *Restore CLI wallet from Seed Phrase*

5.3.4 Exporting miner key

To generate a secret key used by the miner to attribute mining rewards to your wallet run the following command:

```
./beam-wallet export_miner_key --subkey=<integer miner id, i.e 1,2,3...>
```

You will be prompted for the wallet password

The sample output for this command should look like this:

```
$ beam-wallet.exe export_miner_key --subkey=1
I 2018-12-23.16:36:04.306 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.16:36:04.307 starting a wallet...
Enter password: *****
Secret Subkey 1: OVBSdWQ1OV3WuC6bLXRDJqyDfdxWSuzdA4jEGRAZ1zhy4gA3/
↪KcBTedcmN5wNOv0vQrBWwO1TdIxqyPFzFDFdaVYZPUDoXjggUE=
```

It is important to **keep the Miner Key secret** since anyone who knows the miner key will be able to spend all the rewards mined by that miner.

5.3.5 Exporting owner key

The purpose of the `Owner Key` is to allow all nodes mining for you to be aware of all mining rewards mined by other nodes so that you would only need to connect to one node to collect all the rewards into your wallet. While in most other cryptocurrencies this is done by simply mining to a single address you control, in Mumblewimble it is not as simple since there are no addresses and the mining rewards should be coded with unique blinding factors which are deterministically derived from the `Master Key`, and then tagged by the single `Owner Key`.

`Owner Key` should be kept secret. `Owner Key` does not allow to spend coins, however it will allow to see all coins mined for you by all miners that use this `Owner Key`.

To export the `Owner Key` run the following command:

```
./beam-wallet export_owner_key
```

You will be prompted for the wallet password

Sample output for this command should look like this:

```
$ ./beam-wallet export_owner_key
I 2018-12-23.16:53:04.973 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.16:53:04.974 starting a wallet...
Enter password: *
Owner Viewer key: dmVxtRCM3BH1VakviSB/
↪XY86DsCKuWDLKk51eLDlibgMeL2fZ317Zdqx3E6oXbKtldqZz/lo5stTCSz9M1bdJdYUF4DG/ZaIuHHszi/
↪H9wDmNDVboUdNtC/1Z/haWr9JxeIDtRSDBN+xpUbv
```

5.3.6 Receiving BEAMs

To receive BEAMs you need to connect to a specific node by running the following command:

```
./beam-wallet listen -n <node address and port, ex: 127.0.0.1:10000>
```

You will be prompted for the wallet password

A sample output for this command should look like:

```
I 2018-12-23.17:07:55.526 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.17:07:55.527 starting a wallet...
Enter password: *****
I 2018-12-23.17:07:58.076 wallet sucessfully opened...
I 2018-12-23.17:07:58.078 WalletID_
↪14a38140d8e66be9b8f1e8d770161fd33e35f7000053147b5a0f6a83178926b956 subscribes to_
↪BBS channel 20
I 2018-12-23.17:07:59.297 Sync up to 8304-2dc4e5a393d6774b
I 2018-12-23.17:07:59.318 Current state is 8304-2dc4e5a393d6774b
```

Once launched, the wallet will listen to updates from the server and any incoming transactions on the advertise SBBS address.

To receive funds you should send the address to the sending party via any available secure channel (Email, Telegram etc.)

When funds are sent you will see the incoming transaction in the wallet logs and on the screen. It should look similar to:

```
I 2018-12-23.17:55:08.556 [7997ecd5c59e4865a6d938dbf339567e] Receiving 300 beams
↳ (fee: 10 groth )
I 2018-12-23.17:55:08.608 [7997ecd5c59e4865a6d938dbf339567e] Invitation accepted
D 2018-12-23.17:55:09.203 Received PeerSig:      596857beae016ebd
I 2018-12-23.17:55:09.216 [7997ecd5c59e4865a6d938dbf339567e] Transaction kernel:
↳ 95a8e48587c452b3
D 2018-12-23.17:55:09.346 [7997ecd5c59e4865a6d938dbf339567e] has registered
D 2018-12-23.17:55:09.367 Received PeerSig:      596857beae016ebd
I 2018-12-23.17:55:09.428 Get proof for kernel: 95a8e48587c452b3
```

5.3.7 Sending BEAMs

To send beams you need to run the following command:

```
./beam-wallet send -n <node address and port, ex: 127.0.0.1:10000> -r <sbbs address> -
↳ a <amount (in Beams), ex: 11.3> -f <fee (in Groth) , ex: 0.2>
```

Note: 1 Groth equals 10^8 Beam

The wallet log should look similar to something like:

```
$ ./beam-wallet send -n 172.104.249.212:8101 -r
↳ 14a38140d8e66be9b8f1e8d770161fd33e35f7000053147b5a0f6a83178926b956 -a 10
I 2018-12-23.18:05:49.037 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.18:05:49.038 starting a wallet...
Enter password: *
I 2018-12-23.18:05:50.725 wallet sucessfully opened...
I 2018-12-23.18:05:50.726 WalletID
↳ 14a38140d8e66be9b8f1e8d770161fd33e35f7000053147b5a0f6a83178926b956 subscribes to
↳ BBS channel 20
I 2018-12-23.18:05:50.775 [b21f08337dd94603bb038c82c1888eac] Sending 10 beams (fee:
↳ 0 groth )
I 2018-12-23.18:05:50.986 [b21f08337dd94603bb038c82c1888eac] Invitation accepted
I 2018-12-23.18:05:51.053 [b21f08337dd94603bb038c82c1888eac] Transaction kernel:
↳ 71cf20c4c94f25ce
```

Sending transactions to yourself

It is possible, and sometimes necessary to create a transaction to your own SBBS address to split a large UTXO. To do that just issue a send command with required amounts to your own SBBS address. Please note that you will pay the fee for the transaction.

5.3.8 Printing the wallet info

To print the current status of your wallet, run the following command:

```
./beam-wallet info
```

You will be prompted for the wallet password

A sample output for this command should look like this:

```
I 2018-12-23.17:56:19.368 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.17:56:19.369 starting a wallet...
Enter password: *
I 2018-12-23.17:56:21.144 wallet sucessfully opened...
____Wallet summary____

Current height.....8353
Current state ID.....72329a2efa2ddad4

Available.....300 beams
Maturing.....0 groth
In progress.....0 groth
Unavailable.....0 groth
Available coinbase .....0 groth
Total coinbase.....0 groth
Avaliable fee.....0 groth
Total fee.....0 groth
Total unspent.....300 beams

      id |          Beam |          Groth |          height |
↔maturity |          status |          type |          |
      1545571472000001 |          300 |          0 |          8347 |
↔8351 [Available] |          norm |          |          |
```

5.3.9 Creating new SBBS address

In order to create new SBBS address, run the following command:

```
./beam-wallet new_addr --expiration_time=never|24h --label="some label"
```

You will be prompted for the wallet password

Sample output from this command should look like this:

```
I 2018-12-23.18:16:44.112 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.18:16:44.113 starting a wallet...
Enter password: *
I 2018-12-23.18:16:45.392 New address generated:

646a773da4d4651f35fd75ca958b7859e89d8d8382b8155773bd396e2cc49cca
```

5.4 Beam Node User Guide

5.4.1 General

Beam Node is an essential part of the Beam blockchain and is responsible for validating transactions and blocks. It runs on all platforms: Linux, Windows and Mac (for detailed list of supported and tested platforms please see *Supported Platforms*)

Beam Node can be run in either Mining or Validating mode.

5.4.2 Mining mode

Beam Node supports two options for mining Beam:

1. Using Internal Miner

Beam Node has built in GPU and CPU miners. To start the node with internal miner, specify miner type using *miner_type* parameter. Built in GPU mining is only supported on Linux and Windows platforms.

2. Using External Miner via Stratum Server API

Beam Node provides built in support for Stratum API allowing to connect multiple external mining clients to a single node. (see :ref: *user_beam_stratum_server* for more details). To start the node with stratum server use *stratum_port* and *stratum_secrets_path* parameters. Stratum clients can be run together with the Internal Miner

Mining keys management

In order for the mining node to be able to attribute mining rewards to your wallet, it needs a special secret *mining key*. The mining key is derived from the primary secret key by running *export_miner_key* command with *-subkey=<node id>* parameter in the Beam CLI Wallet (See *Command Line Wallet User Guide* for more details). You can generate a multiple separate mining keys for different mining nodes.

Optionally, in order to allow each mining node to be able to see all rewards mined by all your mining nodes Beam provides an additional option called *owner_key*. Owner key is a secret view key, it can not be used to spend coins, just to identify your mining rewards regardless of which node was used to mine it. Owner key is derived from primary secret key as well using the same *key_export* command, but without additional parameters.

Both keys are protected using Wallet Password, which should also be provided

5.4.3 Validating mode

By default (without *-miner_type* flag) Beam Node is run in validating mode, meaning that mining is disabled. Validating nodes are still very important for the overall health and safety of the network since they:

1. Help in propagating transactions and blocks through the network
2. Relay SBBS messages to enable Wallet to Wallet communication.
3. Serve as Dandelion Stem relays to improve P2P level security

If possible, always prefer running a local node either with or without mining!

5.4.4 Node Settings

Beam Node allows to provide the settings via command line or using a configuration file called *beam-node.cfg* and located in the same folder as Beam Node binary.

Command line parameters override configuration file settings

The configuration file is loaded automatically and sets all parameters that were not provided via command line. To reload configuration file after a change you should manually restart Beam Node

Parameter	Description & Example
-port	Port to start the server on <code>port=10000</code>
-log_level	Log level [info debug verbose] <code>log_level=info</code>
-file_log_level	File log level [info debug verbose] <code>file_log_level=info</code>

Parameter	Description & Example
-storage	Path to node database file (defaults to node.db in the same folder) <code>storage=node.db</code>
-history_dir	Path to folder where compressed (cut-through) history files are stored. Defaults to same folder. <code>history_dir=.</code>
-temp_dir	Path to temp folder for compressed (cut-through) history files. Must be on the same volume as history_dir <code>temp_dir=.</code>
-miner_type	Type of built in miner [cpulgpu]. Only relevant for Linux and Windows builds which support GPU mining. In case of CPU mining uses number of threads specified in the mining_threads parameter (see below). <code>miner_type=cpu</code>
-mining_threads	Number of concurrent threads used in CPU mining (if set to 0, mining is disabled) Relevant for CPU mining only <code>mining_threads=0</code>

Using CPU mining is not recommended

Beam uses Equihash mining algorithm with (150,5) parameters and customized data path. It is efficiently mined on GPUs. Using CPU is most likely to be not cost effective.

Parameter	Description & Example
-key_mine	Secret key to attribute mining rewards mined by the node to your wallet Created using CLI <code>wallet export_miner_key</code> command with <code>-subkey=<miner id></code> parameter See <i>Command Line Wallet User Guide</i> for more details
-key_owner	Secret key allowing the node to monitor mining rewards mined by all mining nodes marked by this key. Created using CLI <code>wallet export_owner_key</code> command See <i>Command Line Wallet User Guide</i> for more details
-pass	Wallet password. It is required since both Miner Key and Owner Key are protected by wallet password
-stratum_port	Port on which stratum server will listen to incoming connections. 0 if stratum server is disabled. <code>stratum_port=0</code>
-stratum_secrets_path	Path to folder containing stratum certificates <code>stratum_secrets_path=.</code>

Warning: The following document is still under construction and is subject to changes

5.5 Backup and Restore

This document lists all backup and restore procedures necessary to keep your data safe.

5.5.1 Restore Desktop wallet from Seed Phrase

To be completed...

5.5.2 Restore CLI wallet from Seed Phrase

It is only possible to restore your coins by running your own node with your *owner key*

This process involves several steps

First you need to recreate your wallet using your *seed phrase* by running the following command:

```
./beam-wallet restore --seed_phrase=<semicolon separated list of 12 seed phrase words>
↵;
```

Now you need to export the owner key by running:

```
./beam-wallet export_owner_key
```

(see *Exporting owner key* for more details)

Then you need to run your own node, providing the owner key as a parameter as follows:

```
./beam-node --peer=<ip and port of peer node> --key_owner=<owner key exported from_
↳the wallet>
```

Once the node has synchronized, you need to connect your wallet to the node to update the wallet database.

To do that run the following command:

```
./beam-wallet listen -n <ip and port of your node, ex:127.0.0.1:10000>
```

After wallet synchronizes, use `info` command to check wallet status

```
./beam-wallet info
```

Warning: The following document is still under construction and is subject to changes

5.6 Mining Beam

Alike most cryptocurrencies, Beam relies on miners to add transactions to the blockchain. While all nodes in the Beam network confirm the validity of transactions, Beam counts on miners to take on the massive heavy lifting to guard the network.

Beam is a Mimblewimble implementation. We use classic Proof-of-Work (PoW) consensus.

We welcome everyone to join our mining community to support the network and earn Beam coins.

5.6.1 Mining Algorithm

To secure the network, Beam uses the [Equihash](#) proof-of-work mining algorithm). Miners compete against each other using their computing power produce a new block on the chain. The first miner that gets to complete the precise computation for each block is granted with a network standard block reward and any fees for transactions added to that block.

At Mainnet launch, we will use the following Equihash parameters: $n=150$, $k=5$. In addition, we will introduce a small change to the datapath to further reduce the chance of zero-day ASICs.

Note: Testnet 3 is still using $n=144$, $k=5$

The minimal memory requirement for the GPU will be 4 GB. The most up-to-date list of supported GPUs will be available [here](#).

5.6.2 Block Size and Time

A Beam block will be generated approximately every minute and contain about 1000 transactions. Block size will be roughly 1MB.

5.6.3 Mining Difficulty

Mining difficulty is a measure of how many attempts on average it is required to find the proof-of-work solution required to mine a block and receive the mining reward. One can define the Difficulty as the inverse probability of a

random solution being the correct one. Thus, a difficulty of 100 means that one in 100 tries should produce a valid block in average.

In Proof of Work blockchains, the difficulty is a dynamic parameter, periodically retargeted to reflect the fluctuations in the total computational power of all the miners. Accounting for changes in mining competition, as well as improvements in mining technology.

The goal of updating the difficulty is to keep the average block time at a certain value. In Beam, the target block time is 60 seconds, which underpins a constant currency issuance and the transaction settlement time of the network.

Here's an example. Let's assume we have 100 miners in our network. They are trying different random solutions and find one roughly every minute. Now 100 more miners join, the hashrate of the network doubles, and it will take just 30 seconds to find the solution in average. And if 200 and more join, the time will halve into 15 seconds, and so on.

To mitigate that, Beam nodes adjust [shall we mention that they reach consensus here?]the difficulty, increasing the difficulty of the proof of work algorithm as more miners join, and lowering it if the number of miners decreases.

In Beam, the difficulty is reassessed with every block, by every client independently. The algorithm looks at the average time and the average difficulty the last 1440 blocks. If the time required to mine the last 1440 blocks is higher or lower than 24 hours, the difficulty is retargeted accordingly.

The detailed algorithm is outlined below:

1. Look at the last 7 blocks. Identify the block that has the median block time. This is the Window End block.
2. Look at blocks from 127 to 120 before the current block (7 blocks altogether). Identify the block that has the median block time out of those. This will be the Window Start block.
3. Sum all the difficulties of the blocks from Window Start to Window End. This is Delta Work.
4. Calculate the time difference between Window Start and Window End blocks. This is Delta Time.
5. Calculate the difficulty for next block as: $\text{NewDifficulty} = (\text{Delta Work} / \text{Delta Time}) * 60\text{seconds}$.
6. The Delta Time is bounded by 1 hour and 4 hours. I.e. if the difference is less than 1 hour, 1 hours is still used as Delta Time, same for differences larger than 4 hours. This is done to prevent extreme changes in difficulty.

The algorithm above ensures that as the total solution power of the network fluctuates, the difficulty is gradually adjusted and thus the target block time is ensured.

5.6.4 Miner Rewards

During the first year of Beam existence, miner reward will be 80 coins per block. In years 2-5 the reward will be 40 coins per block. In year 6 the reward will be to 25 coins, and then halving will occur every 4 years until year 129. After year 133, Beam emission will stop.

Mining reward (coinbase UTXO) has 4 hours maturity, meaning that it will be available for spending 4 hours after it was mined.

5.6.5 Treasury

In the first five years of existence, additional coins will be issued to Beam Treasury with each newly mined Beam block.

In the first year, the Treasury will receive additional 20 Beams per block, and in the years 2-5 the Treasury will receive 10 coins per block.

The Treasury will be used to repay Beam investors, Incentivize the Core Team and to support the Beam Foundation (largest single beneficiary of the Treasury).

The distribution of the Treasury Coins is performed on a quarterly basis in the following proportion:

- Investors: 40%
- Core Team: 40%
- Beam Foundation: 20% (Biggest single beneficiary)

5.6.6 ASIC Resistance

To ensure better decentralization, Beam plans to stay ASIC resistance in the first 12-18 months. To achieve this, we plan to perform one or two hard forks – first after approximately 6 months of existence and another one after approximately 12 months. Each hard fork will change the mining algorithm. The exact modifications will be revealed several weeks before the actual hard fork.

5.6.7 Mining Guide

The following section describes how to set up mining for Beam Network

5.6.8 Mining using Desktop Wallet

Our official Desktop Wallet supports both CPU and GPU mining.

Attention: Testnet 4 does not support built in GPU miner. It will be added back for Mainnet release
--

To mine Beam using the Wallet, go to the Settings screen, select “Run Local Node” and set the number of Mining Threads to a value that is greater than zero. Note: the more threads you dedicate to mining, the more strain on your CPU. If you have a supported GPU, turn the “Use GPU” switch to On. If you have a supported GPU, you can also choose “Use GPU” in the Wallet settings. Thread count is not relevant for GPU. Make sure your GPU has the latest drivers installed.

Please bear in mind that since GPUs are much more capable of parallel computations than CPUs, mining with GPUs is much more efficient.

If you want to setup a stand alone mining node and use either built in or external miner, follow sections below.

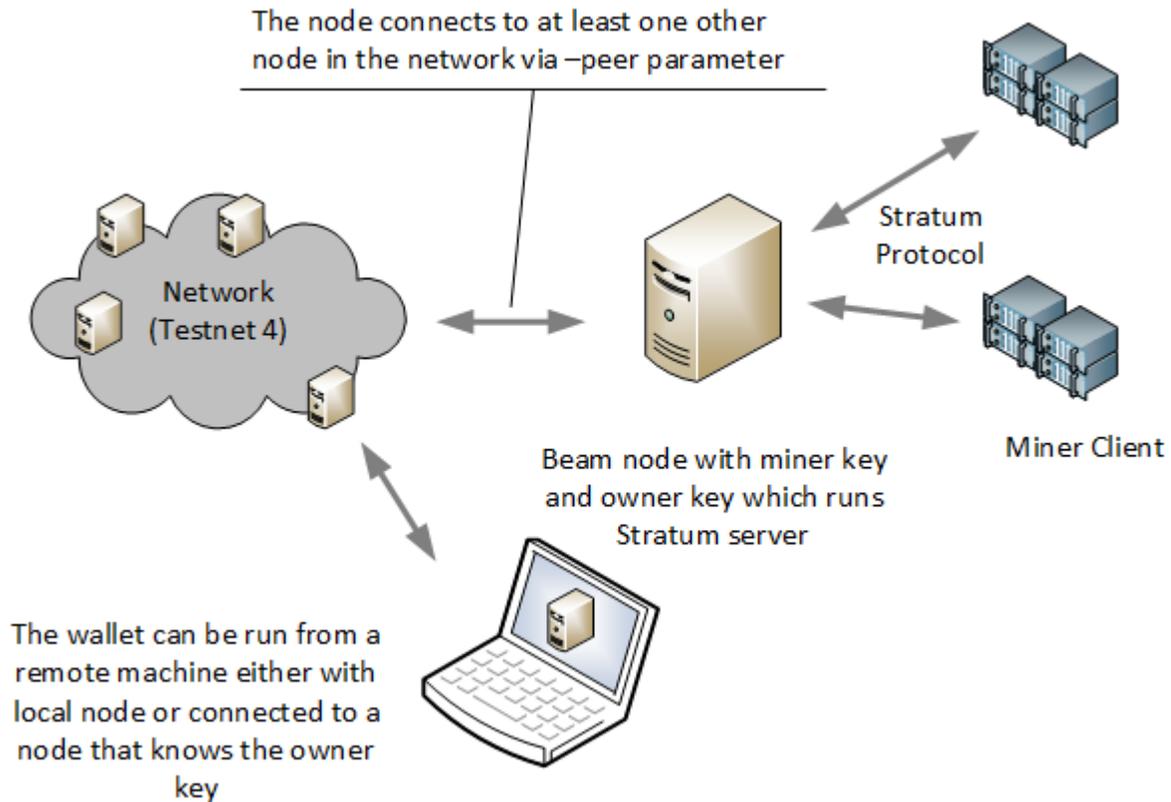
5.6.9 Mining using external miner

This is a step by step guide on how to setup mining using stand alone Beam Node with Stratum Server and a mining client.

Before you start with the steps, please review the sample Mining Architecture

Important points to understand

- Beam node should connect to some other node on the network via `-peer` parameter (for example `-peer=3.0.115.1:8100` for testnet 4)
- Beam node should run Stratum server by setting up `-stratum_port` parameter (for example `-stratum_port=10002`)
- Beam node should know **both** mining key and owner key in order to attribute mining rewards to a specific wallet. Mining and owner keys are exported from the wallet as explained in the steps 5 and 6 of the step by step guide. They are passed to the node via `-key_mine` and `-key_owner` parameters respectively



Important: Miner keys and Owner key should be kept secret at all times

- Several Mining clients can connect to the same node. Mining clients should run on machines with GPUs.
- The connection between the Mining Client and the Beam Node is encrypted using TLS. In order for TLS to work you need to provide a certificate file and secret keys for it, as described in Step 9.
- Mining Client and Beam Node should use the same API key. Details on how to setup an API key are described in Step 9.
- Your wallet will see mining rewards only if the following conditions hold:
 1. It was created using the same seed phrase as the wallet which created the miner and owner keys
 2. It is connected to the node that knows the owner key.

Note: In case of Desktop wallet, you need to run local node from within the wallet (which automatically knows the owner key) or connect to a remote node that knows the owner key. If you just connect to a random node you will NOT see your mining rewards

Beam node should

Now let's start with the steps:

1. Download CLI Wallet archive for your platform from [Beam Website](#)
2. Extract the CLI Wallet to any folder on your machine (we will call it Wallet Folder)

3. Open a Terminal window (on Mac / Linux) or Command Prompt (on Windows) and change directory to Wallet Folder

Attention: The following steps involve seed phrase and should be done in secure environment to avoid someone stealing your seed phrase

4. If you want to create a new wallet (with new seed phrase) run the following command:

```
./beam-wallet init
```

If you have already created a wallet run the following command:

```
./beam-wallet restore --seed_phrase=<semicolon separated list of 12 seed phrase words>  
↵;
```

Attention: Remember your wallet password, you will need it in step 11 to run Stratum server

5. Export miner key by running the following command

Warning: Mining key should be kept **secret**. Never send it or show it to anyone.

```
./beam-wallet export_miner_key --subkey=1
```

Note: If you want to run several different mining nodes with different keys you can run the command again with different subkeys.

Example: For second node, run: `./beam-wallet export_miner_key --subkey=2`

For third node, run: `./beam-wallet export_miner_key --subkey=3`

and so on

Save the exported mining key in some text file, you will need it later

6. Export owner key by running the following command:

Warning: Owner key should be kept **secret**. Never send it or show it to anyone.

```
./beam-wallet export_owner_key
```

Save the exported owner key in some text file, you will need it later

7. Download Beam Node archive for your platform from [Beam Website](#)
8. Extract Beam Node to any folder on your machine (from now on we will call it Node Folder)
9. Create certificate and API key for Stratum server

Note: If you are only testing you can download the sample certificate and key files from here:

Certificate File

Certificate Secret Key File

API Keys file

API Keys file currently contains one key: aaaa1234. You will need to provide it later in the `--key=aaaa1234` parameter for miner client.

You should copy these files to the Beam Node folder (same folder as beam-node binary)

You can now jump to step 10

Attention: For production setup please read the following section carefully

Beam node implements Stratum protocol for connecting external miner clients. Clients open a TCP connection to the node through which they receive jobs to mine blocks using Equihash mining protocol.

Stratum server connections are protected using Transport Layer Security (TLS) protocol and require TLS certificates in order to work properly. You can either buy the certificates or create self signed certificates on your local machine. Instructions on how to do this are outside the scope of this guide. You should receive two files: one for certificate and one with the certificate secret key. For testing purposes you can always use sample files provided in the note above.

In addition you should create a file 'stratum.api.keys' which will contain one or more lines. Each line represents one *API key* - random strings of 8 characters or more. You should generate these keys yourself and put each one in new line. These keys are then used by the miner client via `-key` flag.

As a result you will have three files:

stratum.crt	TLS certificate
stratum.key	Private key for TLS certificate
stratum.api.keys	Text file with list of allowed API keys Each key should have 8 symbols or more. example: abcd1234

All three files should be copied into the same folder. The path to this folder will be provided via `-stratum_secrets_folder` parameter. By default the path points to the same folder as the node binary.

At this point Node Folder should look something like this:

```
29/12/2018 07:43 <DIR> .
29/12/2018 07:43 <DIR> ..
28/12/2018 13:04      1,374 beam-node.cfg
28/12/2018 13:14  4,523,008 beam-node.exe
29/12/2018 07:41  2,670,592 node.db
28/12/2018 13:46          8 stratum.api.keys
28/12/2018 13:45     1,233 stratum.crt
28/12/2018 13:45     1,704 stratum.key
          6 File(s)    7,197,919 bytes
          2 Dir(s)  32,194,318,336 bytes free
```

10. Open a Terminal window (on Mac / Linux) or Command Prompt (on Windows) and change directory to Node Folder
11. Run Beam Node with stratum server using the following command:

```
./beam-node
  --port=10001
  --peer=3.0.115.1:8100
  --stratum_port=10002
  --stratum_secrets_path=.
  --key_mine=<mining key you got in step 5 >
  --key_owner=<owner key you got in step 6>
  --pass=<your wallet password (not seed phrase) >
```

Note: Parameters in the example above are good for testing. You can always change them if necessary. You can also change the beam-node.cfg file and set all these parameters there instead of the command line.

The following table describes all parameters in more details

Parameter	Description & Example
port	Port to start the server on <code>port=10000</code>
stratum_port	Port the stratum server is listening for incoming connections <code>--stratum_port=10002</code>
peer	Comma separated list of peer ip:port (must have at least one peer) Peer should be a machine on the network you want to connect to (for example Testnet 4) List of peers is published on the downloads page at https://beam.mw/downloads <code>--peer=3.1.46.96:8100</code>
stratum_secrets_path	Path to a folder which holds TLS Certificate and API keys files described above. <code>--stratum_secrets_path=.</code>
key_mine	Miner key, exported by CLI wallet (see :ref: <i>Creating CLI wallet for mining rewards</i>) <code>--key_mine=c3C9TVdEgza7w8p9na/ ↪B9rNeC8FvQAbJSPBfLZpW0sw</code>
key_owner	Owner key, exported by CLI wallet <code>--key_owner=mW9ItV9dUsSY9hN/ ↪dh19GEbzIUHQpw6VgDaCPYZiAsNL1LU</code>
pass	Wallet password. <code>--pass=1234</code>

12. Downloads miner client archive for your GPU and platform from [Beam Website](#)

Beam provides two mining clients for Equihash 150,5 with data path change: one for OpenCL and one for CUDA

Attention: Only OpenCL mining client is currently available in Testnet 4

Note: Mining clients are only supported on Linux and Windows platforms

13. Extract miner client to a folder on your mining machine (from now on we will call it Miner Folder)
14. Open a Terminal window (on Mac / Linux) or Command Prompt (on Windows) and change directory to Miner Folder
15. Run the following command (example on Windows):

```
beamMiner.exe --server 127.0.0.1:10002 --key aaaa1234
```

If your node runs on different machine than the miner, change IP address above to the IP of the node machine

If you have set a different API key than 'aaa1234' from the example set your key in the `-key` parameter.

Detailed explanation about mining client parameters is provided in the table below:

Parameter	Description & Example
server	IP and port of the Stratum server to connect to <code>--server 127.0.0.1:10001</code>
key	API key you have set in your Stratum server (In <code>stratum.api.keys</code> file) <code>--key abcd1234</code>
devices	Only specify this flag to use specific GPU By default, miner will use all available GPUs <code>--devices 0</code>

Your mining should start now.

To see your mining rewards use one of two options below:

1. Run Beam Desktop Wallet with the same seed phrase using built in node.
2. Run either CLI or Desktop wallet and connect it to *Your* node which was started with your owner key parameter (via `-key_owner` flag). It could be the same node as the miner, or another node - as long as it has your owner key

Warning: You will NOT be able to see your mining rewards if you connect to a node which does not know your owner key.

5.6.10 GPU Support

Here are some performance stats reported by our community

OpenCL Miner

GPU	Supported	Reported Sol/s rate
AMD RX560	Yes	~4
AMD RX570	Yes	~7-8
AMD RX580	Yes	~8-9
AMD Rx Vega 56	Yes	~13
nVidia GTX 1066	Yes	~5.25
nVidia GTX 1050Ti	Yes	~2.2-4.8
nVidia GTX 1060 6Gb	Yes	~5
nVidia GTX 1070	Yes	~7
nVidia GTX 1080	Yes	~8-9
nVidia GTX 1080Ti	Yes	~10-11
nVidia GTX 2080	Yes	~10-11

CUDA Miner

Note: CUDA Mining client is still in development.

5.7 Blockchain Explorer

Note: Blockchain Explorer for Testnet 4 is currently running at <https://t4.explorer.beam.mw>

Warning: The following document is still under construction and is subject to changes

5.8 Supported Platforms

Beam Node and Beam Wallet are currently supported on the following platforms:

64-bit Linux OS Ubuntu 14.04 LTS Trusty Tahr, Ubuntu 16.04 LTS Xenial Xerus, Ubuntu 18.04 LTS Bionic Beave 64-bit Processor 5GB of free RAM 10GB of free Disk (*the size of the block chain increases over time*)

64-bit Mac OS 10.13 or higher 64-bit Processor 5GB of free RAM 10GB of free Disk (*the size of the block chain increases over time*)

64-bit Windows 8 or higher 64-bit Processor 5GB of free RAM 10GB of free Disk (*the size of the block chain increases over time*)

Attention: At the moment Beam only works on processors that support AVX instruction set.

Note: Please report any issues you might encounter while running BEAM software on your system.

5.9 Files and Locations

5.9.1 Desktop Wallet app

General points to mention:

- The default location of the Desktop Wallet app can be modified during the installation process (Windows only).
- The default Database location for the Desktop Wallet app can be changed setting the *appdata* parameter to *beam-wallet.cfg* (Windows only).
- Memory dump files are generated on Windows only. A dedicated memory dump file is created per each crash case.
- Each version of the wallet keeps the wallet database file (*wallet.db*) in the dedicated sub-folder, designated by the version number. On each wallet update the new folder is created to which and the wallet database file from previous version is copied into (and updated if necessary).

5.9.2 Windows

File	Location
Main Executable	<i>\Program Files\Beam\<version>\Beam Wallet.exe</i>
Configuration	<i>\Program Files\Beam\<version>\beam-wallet.cfg</i>
Logs	<i>\Users\{your User name}\AppData\Local\Beam Walle\logs</i>
Database	<i>\Users\{your User name}\AppData\Local\Beam Walle\<version>\wallet.db (node.db)</i>
Dumps	<i>\Users\{your User name}\AppData\Local\Beam Walle\Beam Wallet.exe0.dmp</i>

5.9.3 Mac

File	Location
Main Executable	<i>/Applications/Beam Wallet.app</i>
Configuration	N/A
Logs	<i>/Users/{your User name}/Library/Application Support/Beam Wallet/logs</i>
Database	<i>/Users/{your User name}/Library/Application Support/Beam Wallet/<version>/wallet.db</i>

5.9.4 Linux

File	Location
Main Executable	<i>/usr/bin/BeamWallet</i>
Configuration	<i>/usr/bin/beam-wallet.cfg</i>
Logs	<i>/home/{your User name}/.local/share/Beam Wallet/logs</i>
Database	<i>/home/{your User name}/.local/share/Beam Wallet/<version>/wallet.db (node.db)</i>

5.9.5 Node or CLI wallet

All Platforms (small differences apply, see below)

User can unpack the archive in any folder to his convenience. All files mentioned below are located within this folder

File	Location
Main Executable	<i>beam-node</i> or <i>beam-wallet</i>
Configuration	<i>beam-node.cfg</i> or <i>beam-wallet.cfg</i>
Logs	logs
Database	<i>node.db</i> or <i>wallet.db</i>
Dumps (windows only)	<i>beam-node.exe0.dmp</i>

5.10 Reporting Issues and Getting Support

Work in progress

5.11 Troubleshooting

5.11.1 Command Line Wallet

1. I am getting the following error when trying to run command line wallet:

```
I 2018-12-23.17:32:34.619 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.17:32:34.620 starting a wallet...
Enter password: ***
D 2018-12-23.17:32:36.664 sqlite error code=26, file is not a database
E 2018-12-23.17:32:36.665 Wallet data unreadable, restore wallet.db from latest
↳backup or delete it and reinitialize the wallet
```

Resolution:

You are trying to enter incorrect password. The wallet can not decrypt the database file and hence reports that data is unreadable. *Only if you are absolutely sure that password is correct, remove the database file and restore wallet from Seed Phrase*

2. I am getting the following error when trying to run command line wallet:

```
I 2018-12-23.17:45:12.529 Rules signature: ddccf5d8d0f77bd2
I 2018-12-23.17:45:12.530 starting a wallet...
Enter password: *
I 2018-12-23.17:45:13.226 wallet sucessfully opened...
I 2018-12-23.17:45:13.228 WalletID
↳14a38140d8e66be9b8f1e8d770161fd33e35f7000053147b5a0f6a83178926b956 subscribes to
↳BBS channel 20
I 2018-12-23.17:45:13.271 [9edc454f2752461eb682f21c4efbd33e] Sending 10 beams (fee:
↳0 groth )
E 2018-12-23.17:45:13.272 You only have 0 groth
E 2018-12-23.17:45:13.273 [9edc454f2752461eb682f21c4efbd33e] exception msg:
E 2018-12-23.17:45:13.273 [9edc454f2752461eb682f21c4efbd33e] Failed. No inputs
I 2018-12-23.17:45:13.293 [9edc454f2752461eb682f21c4efbd33e] Transaction failed.
↳Rollback...
```

Resolution:

The most common cause of this error is trying to send a transaction with insufficient funds. You can not send more than you have.

Warning: The following document is still under construction and is subject to changes

5.12 Resources

This page includes a list of useful and recommended resources to learn about cryptocurrencies in general and Beam in particular.

5.12.1 Blockchain

5.12.2 Cryptocurrencies

5.12.3 Mimblewimble

5.12.4 Beam

5.13 Glossary

Address In Beam, addresses are only used by SBBS system to connect between Wallets during transaction creation. Unlike most other blockchains, addresses are not recorded in the blockchain and are not used to prove ownership of the coins. Each address has a default expiration time of 24 hours (which can be changed using Wallet UI). In general, it is recommended to generate fresh receiving address for each transaction.

Blinding factor

Blinding factor is a secret key that is deterministically derived from a *master key* and is used to

Block A block is a record in the Beam blockchain that contains a set of transactions sent on the network. Pending inclusion in a block, a transaction is kept in the **'mempool'** in an **'unconfirmed'** state. Roughly every 2.5 minutes, on average, a new block is appended to the *blockchain* through **'mining'** and the transactions included receive their first **'confirmation'**.

Block reward A block reward is new **'beams'** created after the successful **'mining'** of a block. For the first five years, the block reward in Beam is split into a *miners' reward* and a *treasury*. During this time, miners receive 80% per block with the remaining 20% split between *Beam Foundation*, founders, employees, investors and advisors. Block reward is subjected to periodic halving and is halved for the first time after about one year (in blocks) and then every four years.

Blockchain The blockchain is a public record of Beam transactions. The blockchain can be downloaded and verified by all Beam nodes to make sure the transaction history is valid and no **double spending** occurs. In Beam, a compacted version of the blockchain is maintained by each node using the *cut through* feature of *mimblewimble* which enables new nodes to only download minimal amount of information to start mining and verifying new blocks.

Blockchain Explorer The Blockchain explorer is a website showing the current status of Beam blockchain and history of mined blocks.

Cut through In *mimblewimble* protocol, cut through feature enables creation of compacted, yet still verifiable, blockchain history by removing all intermediate transactions and thus significantly reducing the amount of information required by a new node in the system to start mining or verifying new blocks.

Macroblock

A macroblock is a compressed version of blockchain history implementing the cut-through feature of Mimblewimble protocol. Each node generates macroblocks in the background and stores them on the local disk. When new node connects to the system it first downloads the latest Macroblock and then updates more recent blocks in the blockchain one by one. This allows to significantly reduce the time of onboarding new nodes into the system.

Master key Master key (or master secret key) is the key used to generate all blinding factors in a single wallet. Master key is also used to generate Miner Keys and Owner key used to mine coins for a specific wallet. In general

Mining difficulty

Mining difficulty is a dynamic parameter which determines amount of calculations necessary to solve Beam Proof of Work puzzle. It is designed to ensure that blocks are created once a minute (on average) regardless of number of miners in the network. Current block creation time and difficulty can be seen using the [blockchain explorer](#).

Seed Phrase

A list of 12 words that hold all information necessary to generate Master Key and hence to recover all Beam UTXOs belonging to a wallet created using this phrase. Seed phrase is generated during wallet initialization and should be kept secret at all times. If lost, the Seed Phrase CAN NOT BE RECOVERED by any means. Keep it safe.

Mimblewimble Mimblewimble is the protocol used by Beam to provide confidentiality of transactions and scalability in terms of compact blockchain size. [Mimblewimble white paper](#) was published in July 2016 by an anonymous author under the pseudonym Tom Elvis Jedusor.

SBBS

SBBS (abbreviation of Secure Bulletin Board System) is a subsystem within Beam Node that allows wallets to securely exchange encrypted messages and create transactions without having to be online at the same time.

Wallet Password

Wallet Password is a password that protects wallet and encrypts wallet database used to store information about UTXOs, transactions and any additional metadata stored by the wallet. Wallet Password is NOT used for or has any relation to ownership of the coins. If Wallet Password is lost, the wallet database can no longer be accessed and the transaction history and metadata will be lost. However it is possible to recover all the currently owned UTXOs, by creating a new wallet and initializing it using the same [seed phrase](#) as the original one.

Transactions

In [mimblewimble](#) protocol transactions contain of Inputs, Outputs and Kernels. Each input and output are represented by Pedersen Committments in a form: $P = v * H + b * G$, where v is transaction value, b is the [blinding factor](#) and G and H are two known ‘nothing up my sleeve’ generator points on the same elliptic curve.

5.14 Building Beam

Warning: The following document is still under construction and is subject to changes

The following document describes how to build Beam binaries from sources located at: <https://github.com/BeamMW/beam>

5.14.1 Windows

1. Install Visual Studio \geq 2017 with CMake support.
2. Download and install Boost prebuilt binaries https://sourceforge.net/projects/boost/files/boost-binaries/1.68.0/boost_1_68_0-msvc-14.1-64.exe, also add BOOST_ROOT to the Environment Variables.
3. Download and install OpenSSL prebuilt binaries <https://slproweb.com/products/Win32OpenSSL.html> (Win64 OpenSSL v1.1.0h for example) and add OPENSSL_ROOT_DIR to the Environment Variables.
4. Download and install QT 5.11 https://download.qt.io/official_releases/qt/5.11/5.11.0/qt-opensource-windows-x86-5.11.0.exe.mirrorlist and add QT5_ROOT_DIR to the Environment Variables (usually it looks like `../5.11.0/msvc2017_64`), also add QML_IMPORT_PATH (it should look like `%QT5_ROOT_DIR%qml`). BTW disabling system antivirus on Windows makes QT installing process much faster.
5. Add `../qt511/5.11.1/msvc2017_64/bin` and `../boost_1_68_0/lib64-msvc-14.1` to the System Path.
6. Open project folder in Visual Studio, select your target (Release-x64 for example, if you downloaded 64bit Boost and OpenSSL) and select CMake -> Build All.
7. Go to CMake -> Cache -> Open Cache Folder -> beam (you'll find beam.exe in the beam subfolder, beam-wallet.exe in ui subfolder).

5.14.2 Linux (Ubuntu 14.04 and higher)

1. Install `gcc7 boost ssl` packages.

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt update
sudo apt install g++-7 libboost-all-dev libssl-dev -y
```

2. Set it up so the symbolic links `gcc`, `g++` point to the newer version:

```
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 60 \
                        --slave /usr/bin/g++ g++ /usr/bin/g++-7
sudo update-alternatives --config gcc
gcc --version
g++ --version
```

3. Install latest CMake

```
wget "https://cmake.org/files/v3.12/cmake-3.12.0-Linux-x86_64.sh"
sudo sh cmake-3.12.0-Linux-x86_64.sh --skip-license --prefix=/usr
```

4. Add proper QT 5.11 repository depending on your system <https://launchpad.net/~beineri> (for example, choose *Qt 5.10.1 for /opt Trusty* if you have Ubuntu 14.04), install `sudo apt-get install qt510declarative qt510svg` packages and add `export PATH=/opt/qt511/bin:$PATH`.
5. Go to Beam project folder and call `cmake -DCMAKE_BUILD_TYPE=Release . && make -j4`.
6. You'll find *Beam* binary in *bin* folder, *beam-wallet* in *ui* subfolder.

5.14.3 Mac

1. Install Brew Package Manager.
2. Install necessary packages using `brew install openssl boost cmake qt5` command.

3. Add `export OPENSSL_ROOT_DIR="/usr/local/opt/openssl"` and `export PATH="/usr/local/opt/qt/bin:$PATH"` to the *Environment Variables*.
4. Go to Beam project folder and call `cmake -DCMAKE_BUILD_TYPE=Release . && make -j4`.
5. You'll find *Beam* binary in *bin* folder, *beam-wallet* in *ui* subfolder.

Note: If you don't want to build UI don't install QT5 and add `-DBEAM_NO_QT_UI_WALLET=On` command line parameter when you are calling `cmake`.

Warning: The following document is still under construction and is subject to changes

5.15 Understanding Beam logs

5.15.1 Log locations

For CLI wallet and Node logs are usually located in the same folder as the binary file.

For Desktop Wallet logs are located in the following folders:

Mac: `/Users/{your_user_name}/Library/Application Support/Beam Wallet/`

Windows: `:\Users\{your_user_name}\AppData\Local\Beam Wallet`

Linux: `/home/{your_user_name}/.local/share/Beam Wallet`

For a complete list of file locations see [:ref: Files and Locations](#)

5.15.2 Node logs

A log will start with the Rules signature. Rules signature is the hash of the Consensus Rules and it should be compatible with the network you are connecting to. It will be different between the Testnet and Mainnet. The relevant Rules signature for each network can be seen here: [:ref: rules_signature](#)

Beam Logs have a simple structure. First field is the severity level, followed by a timestamp and the log message.

```
Log Severity +-----+-----+ | I | Info | +-----+ +-----+ | W | Warn-
ing | +-----+ +-----+ | E | Error | +-----+ +-----+ | D | Debug |
+-----+ +-----+
```

The sample below shows a start of the new node

```
I 2018-12-31.16:48:58.838 Rules signature: 7e16d65b64ef2fbb
I 2018-12-31.16:48:58.986 starting a node on 10000 port...
I 2018-12-31.16:48:58.996 Node ID=5c8f92a1cfaee337
I 2018-12-31.16:48:58.996 Initial Tip: 0-0000000000000000
I 2018-12-31.16:48:58.996 Requesting block 0-0000000000000000
I 2018-12-31.16:48:58.997 PI 0000000000000000--0.0.0.0 New
I 2018-12-31.16:48:58.997 PI 0000000000000000--0.0.0.0 Address changed to 23.239.24.
→209:8201
I 2018-12-31.16:48:58.999 stratum server listens to 0.0.0.0:10002
```

The node connects to the first peer, in this case 23.239.24.209:8201 and downloads the initial Tip at height 0. It then requests the matching block. In this specific example the node also starts the Stratum server.

5.15.3 CLI Wallet logs

5.15.4 Desktop Wallet logs

5.16 Consensus Rules

The following parameters allow to configure consensus rules in Beam Node and Wallet

WARNING: Used for development and testing only

Consensus parameters are only relevant for testing and development purposes. Changing these parameters will change the Rule Signature and hence break the compatibility of the Node and Wallet with running Mainnet or Testnet servers.

5.16.1 Rules

Parameter	Description & Example
EmissionValue0	Initial coinbase emission in a single block (in Groth, 10^{-8} of Beam) <code>EmissionValue0=800000000</code>
EmissionDrop0	Height of the last block that still has the initial emission. Emission drops by half in the next block. Default equals 1 year, assuming 1 block per minute <code>EmissionDrop0=525600</code>
EmissionDrop1	Number of blocks in halving cycle (defaults to four years, assuming 1 block per minute) <code>EmissionDrop1=2102400</code>
MaturityCoinbase	Number of blocks that should be mined (confirmations) before coinbase UTXO can be spent. <code>MaturityCoinbase=240</code>
MaturityStd	Number of blocks that should be mined (confirmations) before normal (non coinbase) UTXO can be spent. <code>MaturityStd=0</code>
MaxBodySize	Block body size (in bytes) <code>MaxBodySize=0x100000</code>
DesiredRate_s	Target block rate (in seconds) <code>DesiredRate_s=60</code>
DifficultyReviewWindow	Number of blocks in the window for the mining difficulty adjustment <code>DifficultyReviewWindow=1440</code>
TimestampAheadThreshold_s	Block timestamp tolerance (in seconds) <code>TimestampAheadThreshold_s=7200</code>
WindowForMedian	Number of blocks considered in calculating the timestamp median <code>WindowForMedian=25</code>
AllowPublicUtxos	Flag allowing regular (non-coinbase) UTXO to have non-confidential signature <code>AllowPublicUtxos=0</code>
FakePoW	Flag to disable verification of PoW. Mining is simulated by timer. <code>FakePoW=0</code>

Below is an example of corresponding .cfg file section:

```
#####
# Rules configuration:
#####

# initial coinbase emission in a single block
# EmissionValue0=800000000

# height of the last block that still has the initial emission, the drop is starting_
↳from the next block
# EmissionDrop0=525600

# Each such a cycle there's a new drop
# EmissionDrop1=2102400

# num of blocks before coinbase UTXO can be spent
# MaturityCoinbase=240

# num of blocks before non-coinbase UTXO can be spent
# MaturityStd=0

# Max block body size [bytes]
# MaxBodySize=0x100000

# Desired rate of generated blocks [seconds]
# DesiredRate_s=60

# num of blocks in the window for the mining difficulty adjustment
# DifficultyReviewWindow=1440

# Block timestamp tolerance [seconds]
# TimestampAheadThreshold_s=7200

# How many blocks are considered in calculating the timestamp median
# WindowForMedian=25

# set to allow regular (non-coinbase) UTXO to have non-confidential signature
# AllowPublicUtxos=0

# Don't verify PoW. Mining is simulated by the timer
# FakePoW=0
```

5.17 Local Setup

In some cases you would want to start a local network for testing and development purposes.

5.17.1 Starting a new network

To start a new Beam Network follow the steps below:

1. Download or build Beam binaries for node and CLI wallet

From this point the folders containing node and wallet binaries will be called ‘node folder’ and ‘wallet folder’ respectively

2. Initialize new wallet by running the following command in the wallet folder:

```
./beam-wallet init
```

See *Creating new wallet* for more details

3. Export miner key, by running the following command in the wallet folder:

```
./beam-wallet export_miner_key --subkey=1
```

Note: If you are running more than one mining node repeat step 3 with `--subkey=N` (where N is the id of the mining nodes 1,2,3...)

4. Export owner key by running the following command in the wallet folder:

```
./beam-wallet export_owner_key
```

5. Locate sample Beam treasury file `treasury.bin` (it is located in the root of Beam source folder) and copy it to the same folder as `beam-node` binary.

6. Launch the first node using the following command:

```
./beam-node --mining_threads=1 --treasury_path=treasury.bin --key_owner=<owner key> --  
↪key_mine=<miner key> --pass=<wallet password>
```

Sample output printed by the node to the console (and in the logs) should look like this:

```
I 2018-12-25.09:43:56.712 Rules signature: ddccf5d8d0f77bd2  
I 2018-12-25.09:43:56.925 starting a node on 10000 port...  
I 2018-12-25.09:43:56.930 Treasury size: 584661  
I 2018-12-25.09:43:56.955 Node ID=a82306ff757cca58  
I 2018-12-25.09:43:56.955 Initial Tip: 0-000000000000000000  
I 2018-12-25.09:43:56.955 Sync mode  
I 2018-12-25.09:43:56.955 Searching for the best peer...  
I 2018-12-25.09:43:56.959 GenerateNewBlock: size of block = 294; amount of tx = 0  
I 2018-12-25.09:43:56.960 Block generated: Height=1, Fee=0, Difficulty=02-000000(4),  
↪Size=294  
I 2018-12-25.09:43:56.960 Mining nonce = 428dcf03b88fa57d
```

As new blocks are mined, the console will indicate this as follows:

```
I 2018-12-25.09:46:00.513 New block mined: 1-a690c8aa7e14225c  
I 2018-12-25.09:46:00.520 1-a690c8aa7e14225c Header accepted  
I 2018-12-25.09:46:00.520 1-a690c8aa7e14225c Block received  
I 2018-12-25.09:46:00.523 1-a690c8aa7e14225c Block interpreted. Fwd=1  
I 2018-12-25.09:46:00.524 My Tip: 1-a690c8aa7e14225c, Work = 4  
I 2018-12-25.09:46:00.527 GenerateNewBlock: size of block = 294; amount of tx = 0  
I 2018-12-25.09:46:00.527 Block generated: Height=2, Fee=0, Difficulty=02-000000(4),  
↪Size=294  
I 2018-12-25.09:46:00.528 Mining nonce = 32d6ce8dba24d0dd
```

Note: For GPU mining, add `--miner_type=gpu` parameter

7. Launch additional nodes

To launch more mining nodes either for the same wallet or different ones, just repeat the relevant portions of steps 1 through 6. For validating nodes set `mining_threads` parameter to 0.

In addition add `-peer=<ip:port of the first node>` to connect new nodes to the first node, thus extending the network

Attention: Beam Nodes have automatic discovery mechanism that will find and automatically connect to the other running nodes on the local network. If you do not want the networks to mix, just change any of the *Consensus Rules* to intentionally break compatibility between the networks and avoid unwanted behavior. The simplest way to achieve this is to set 'Prehistoric' parameter to a random hash value, i.e Prehistoric=1234