
GoByte Documentation

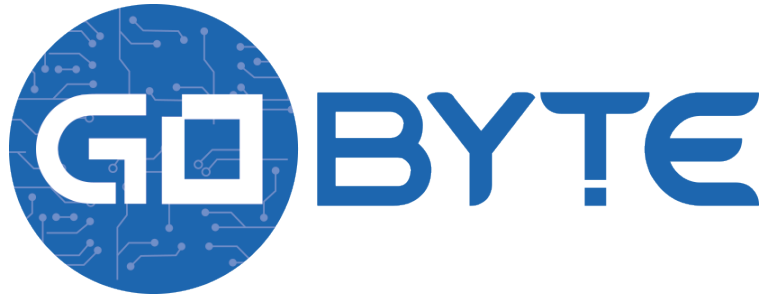
Release latest

antonimoratti

Aug 10, 2020

1	Contents	3
1.1	What is GoByte?	3
1.1.1	GoByte Videos	3
1.1.2	Whitepaper	4
1.1.3	Roadmap	4
1.2	Features	4
1.2.1	Specifications	4
1.2.2	Masternodes	5
1.2.3	PrivateSend	5
1.2.4	InstantSend	6
1.2.5	ChainLocks	6
1.2.6	Sporks	7
1.2.7	NeoScript Hash Algorithm	7
1.2.8	Dark Gravity Wave	7
1.2.9	Emission Rate	8
1.2.10	Decentralized Governance	9
1.2.11	Sentinel	9
1.2.12	Fees	9
1.2.13	Evolution	11
1.3	How To Buy	11
1.3.1	Exchanges	12
1.3.2	Instant exchanges	14
1.3.3	Over the Counter	15
1.3.4	ATMs	16
1.4	Safety	17
1.4.1	Impersonation	17
1.4.2	Scams	17
1.4.3	Ponzi Schemes	18
1.5	Links and Information	19
1.5.1	Links	19
1.5.2	Tools	22
1.5.3	Mobile Apps	23
1.5.4	Glossary	23
1.6	Wallets	32
1.6.1	GoByte Core Wallet	32
1.6.2	GoByte Electrum Wallet	114

1.6.3	GoByte Android Wallet	157
1.6.4	GoByte iOS Wallet	157
1.6.5	GoByte Paper Wallet	158
1.6.6	Hardware Wallets	164
1.6.7	Third Party Wallets	175
1.6.8	Web Wallets	185
1.6.9	Text Wallets	189
1.6.10	Wallet Guides	191
1.7	Earning and Spending	205
1.7.1	Earning	205
1.7.2	Spending	205
1.7.3	Tax	207
1.8	Getting Started	207
1.8.1	Payment Processors	208
1.8.2	Installation Examples	209
1.9	Administrative Processes	219
1.9.1	Onboarding Process	220
1.9.2	Promoting GoByte	220
1.9.3	Currency Conversion	220
1.9.4	Legal considerations	220
1.10	Governance	220
1.10.1	Understanding GoByte Governance	221
1.10.2	Using GoByte Governance	226
1.10.3	8 Steps to a Successful Proposal	235
1.11	Masternodes	236
1.11.1	Understanding Masternodes	237
1.11.2	Hosting Services	240
1.11.3	Setup	248
1.11.4	Maintenance	270
1.11.5	Advanced Topics	289
1.12	Mining	290
1.12.1	Masternodes vs. Mining	290
1.12.2	Mining Pools	291
1.12.3	CPU Mining	301
1.12.4	GPU Mining	305
1.12.5	FPGA Mining	308
1.13	Developers	311
1.13.1	Translating GoByte	311
1.13.2	Compiling GoByte Core	314
1.13.3	Testnet and devnets	320
1.13.4	Sporks	322
1.13.5	Version History	323
1.14	Marketing	324
1.14.1	Design Materials	324
1.14.2	Business Templates	329
1.15	Legal	330
1.15.1	How the Law Applies to GoByte	330
1.15.2	PrivateSend Legal Position	332
1.15.3	ATM & Fiat Compliance	332



GoByte is an open source peer-to-peer cryptocurrency with a strong focus on the payments industry. GoByte offers a form of money that is portable, inexpensive, divisible and fast. It can be spent securely both online and in person with only minimal transaction fees. Based on the Bitcoin project, GoByte aims to be the most user-friendly and scalable payments system in the world. In addition to Bitcoin's feature set, GoByte currently also offers instant transactions (*InstantSend*), private transactions (*PrivateSend*) and operates a self-governing and self-funding model that enables the GoByte network to pay individuals and businesses for work that adds value to the network. This *decentralized governance and budgeting system* makes it one of the first ever successful decentralized autonomous organizations (DAO).

If you are new to cryptocurrencies, the most important change to understand is that transactions occur directly between two parties without any central authority to facilitate the transaction. This also means that you are responsible for your own security - there is no bank or credit card company to reverse a transaction if your funds are stolen or lost. In this sense, it is similar to cash or gold, but cryptocurrency can be spent locally and internationally with equal ease, if you are confident you are sending funds to the right destination. For these reasons, the GoByte documentation has a strong focus on safety and understanding the concepts and features that drive the GoByte ecosystem. The videos, links and documentation below can help you get started, or use the table of contents on the left to find a specific topic of interest.

1.1 What is GoByte?

GoByte aims to be the most user-friendly and scalable payments-focused cryptocurrency in the world. The GoByte network features *instant transaction confirmation*, double spend protection, optional *privacy* equal to that of physical cash, a *self-governing, self-funding model* driven by *incentivized full nodes* and a *clear roadmap* for on-chain scaling to up to 400MB blocks using custom-developed open source hardware. While GoByte is based on Bitcoin and compatible with many key components of the Bitcoin ecosystem, its two-tier network structure offers significant improvements in transaction speed, privacy and governance. This section of the documentation describes these and many more key features that set GoByte apart in the blockchain economy.

Check out the [official GoByte website](#) to learn how [individuals](#) and [businesses](#) can use GoByte. The videos, links and documentation collected here can help you get started, or use the table of contents on the left to find a specific topic of interest. New users may be interested in getting started with an appropriate *wallet*, learning about *how to buy GoByte* and *where to spend GoByte*, learning about *safety* or joining one of the many *GoByte community sites*.

1.1.1 GoByte Videos

What is GoByte?

GoByte Crypto YT has produced an in-depth video describing GoByte and its many distinguishing features.

GoByte Academy

GoByte Academy is a six-part video series produced by Antonio M. Moratti. It explains GoByte from a beginner's level up to descriptions of the more advanced features.

1.1.2 Whitepaper

The GoByte Whitepaper describes the original unique value proposition and key innovations in GoByte from an academic and theoretical perspective. It is a historical document available as a GitHub wiki, and no longer receives updates and as new features are implemented. Instead, these features are described in successive GoByte Improvement Proposals (DIPs), while larger architectural changes are described in separate whitepapers. Features backported from Bitcoin are described in Bitcoin Improvement Proposals (BIPs).

- [Whitepaper and translations](#)
- [PDF whitepaper](#)
- [GoByte Improvement Proposals \(GIPs/DIPs\)](#)
- [Bitcoin Improvement Proposals \(BIPs\)](#)
- [Dash Evolution Initial Design Document](#)
- [Original Darkcoin whitepaper \(PDF\)](#)
- [InstantSend whitepaper \(PDF\)](#)

1.1.3 Roadmap

The GoByte Roadmap sets out delivery milestones for future releases of GoByte and includes specific technical details describing how the development team plans to realise each challenge.

- [GoByte Roadmap](#)

1.2 Features

1.2.1 Specifications

- First block mined at 16:00:01 UTC, 16th November 2017
- 850,000 GBX premine
- NeoScript hashing algorithm, CPU/GPU/FPGA mining available
- 2.5 minute block time, 2MB blocks, ~56 transactions per second
- Block reward decreases by 8.333% per year
- Dark Gravity Wave difficulty adjustment algorithm
- Between 34.05M and 37.84M total coin supply
- Decentralized second-tier masternode network
- Superior privacy using PrivateSend
- Instant transactions using InstantSend
- Protection against blockchain reorganization events (commonly called 51% attacks) using ChainLocks
- Decentralized Governance By Blockchain allows masternode owners to vote on budget proposals and decisions that affect GoByte

1.2.2 Masternodes

In addition to traditional Proof of Work (PoW) rewards for mining GoByte, users are also rewarded for running and maintaining special servers called masternodes. Thanks to this innovative two tier network, GoByte can offer innovative features in a trustless and decentralized way. Masternodes are used to power PrivateSend, InstantSend, and the governance and treasury system. Users are rewarded for running masternodes; 65% of the block reward is allocated to pay the masternode network. You can view practical guides on all topics relating to masternodes [here](#).

Masternodes enable the following services:

- **InstantSend** allows for near-instant transactions. GoByte InstantSend transactions are fully confirmed within two seconds.
- **PrivateSend** gives financial privacy through a decentralized implementation of CoinJoin.
- **ChainLocks**, which protects the blockchain against 51% mining attacks by signing blocks as they are mined.
- **Governance and Treasury** allows stakeholders in GoByte to determine the direction of the project and devotes 10% of the block reward to development of the project and ecosystem.
- **GoByte Evolution** will make using cryptocurrency as easy as using PayPal.

Masternode owners must have possession of 1000 GBX, which they prove by signing a message included in a special transaction written to the blockchain. The GoByte can be moved or spent at any time, but doing so will cause the masternode to fall out of queue and stop earning rewards. Masternode users are also given **voting rights** on proposals. Each masternode has one vote and this vote can be used on budget proposals or important decisions that affect GoByte.

Masternodes cost money and effort to host so they are paid a percentage of the block reward as an incentive. Because only one masternode is paid in each block, the frequency of the payment can vary, as well as the value of the GoByte paid out. This [tool](#) shows a live calculation of masternode earnings. These rewards decrease by 8.33% each year, together with the block reward. There is also the possibility for masternodes to earn money from fees in the future.

1.2.3 PrivateSend

PrivateSend gives you true financial privacy by obscuring the origins of your funds. All the GoByte in your wallet is comprised of different “inputs”, which you can think of as separate, discrete coins. PrivateSend uses an innovative process to mix your inputs with the inputs of at least two other people in a single transaction, so the value in GoByte never leaves your wallet. You retain control of your money at all times.

You can view a practical guide to use PrivateSend [here](#).

The PrivateSend process works like this:

1. PrivateSend begins by breaking your transaction inputs down into standard denominations. These denominations are 0.001, 0.01, 0.1, 1 and 10 GBX – much like the paper money you use every day.
2. Your wallet then sends requests to specially configured software nodes on the network, called “masternodes”. These masternodes are informed then that you are interested in mixing a certain denomination. No identifiable information is sent to the masternodes, so they never know “who” you are.
3. When two other people send similar messages, indicating that they wish to mix the same denomination, a mixing session begins. The masternode mixes up the inputs and instructs all three users’ wallets to pay the now-transformed input back to themselves. Your wallet pays that denomination directly to itself, but in a different address (called a change address).
4. Your wallet must repeat this process a number of times with each denomination. Each time the process is completed, it’s called a “round”. Each round of PrivateSend makes it exponentially more difficult to determine where your funds originated. The user may choose between 1-16 rounds of mixing.
5. This mixing process happens in the background without any intervention on your part. When you wish to make a private transaction, your funds will be ready to spend. No additional waiting is required.

Note that PrivateSend transactions will be rounded up so that all transaction inputs are spent. Any excess GoByte will be spent on the transaction fee.

IMPORTANT: Your wallet only contains 1000 of these “change addresses”. Every time a mixing event happens, one of your addresses is used up. Once enough of them are used, your wallet must create more addresses. It can only do this, however, if you have automatic backups enabled. Consequently, users who have backups disabled will also have PrivateSend disabled.

1.2.4 InstantSend

Traditional decentralized cryptocurrencies must wait for certain period of time for enough blocks to pass to ensure that a transaction is both irreversible and not an attempt to double-spend money which has already been spent elsewhere. This process is time-consuming, and may take anywhere from 15 minutes to one hour for the widely accepted number of six blocks to accumulate. Other cryptocurrencies achieve faster transaction confirmation time by centralizing authority on the network to various degrees.

GoByte suffers from neither of these limitations thanks to its second-layer network of masternodes. Masternodes regularly form voting quorums to check whether or not a submitted transaction is valid. If it is valid, the masternodes “lock” the inputs for the transaction and broadcast this information to the network, effectively promising that the transaction will be included in subsequently mined blocks and not allowing any other spending of these inputs during the confirmation time period.

InstantSend technology will allow for cryptocurrencies such as GoByte to compete with nearly instantaneous transaction systems such as credit cards for point-of-sale situations while not relying on a centralized authority. Widespread vendor acceptance of GoByte and InstantSend could revolutionize cryptocurrency by shortening the delay in confirmation of transactions from as long as an hour (with Bitcoin) to as little as a few seconds.

You can view a practical guide to use InstantSend here. InstantSend was introduced in a whitepaper called [Transaction Locking and Masternode Consensus: A Mechanism for Mitigating Double Spending Attacks](#), and further improved through the introduction of LLMQ-based InstantSend in GoByte 0.14.

How Dash & GoByte ‘InstantSend’ Protects Merchants from Double Spends, Dash Detailed by Amanda B. Johnson, 16 September 2016

1.2.5 ChainLocks

ChainLocks are a feature provided by the GoByte Network which provides certainty when accepting payments. This technology, particularly when used in parallel with *InstantSend*, creates an environment in which payments can be accepted immediately and without the risk of “Blockchain Reorganization Events”.

The risk of blockchain reorganization is typically addressed by requiring multiple “confirmations” before a transaction can be safely accepted as payment. This type of indirect security is effective, but at a cost of time and user experience. ChainLocks are a solution for this problem.

ChainLocks Process Overview

Every twelve hours a new “LLMQ” (Long-Living Masternode Quorum) is formed using a “DKG” (Distributed Key Generation) process. All members of this Quorum are responsible for observing, and subsequently affirming, newly mined blocks:

1. Whenever a block is mined, Quorum Members will broadcast a signed message containing the observed block to the rest of the Quorum.
2. If 60% or more of the Quorum sees the same new block they will collectively form a “CLSIG” (ChainLock Signature) message which will be broadcast to the remainder of the network.

3. When a valid ChainLock Signature is received by a client on the network, it will reject all blocks at the same height that do not match the block specified in that message.

The result is a quick and unambiguous decision on the “correct” blockchain for integrated clients and wallets. From a security perspective, this also makes reorganizations prior to this block impossible. See [DIP0008 ChainLocks](#) for a full description of how ChainLocks work.

1.2.6 Sporks

In response to unforeseen issues with the rollout of the major “RC3” update in June 2014, the Dash development team created a mechanism by which updated code is released to the network, but not immediately made active (“enforced”). This innovation was adopted by GoByte and it allows for far smoother transitions than in the traditional hard fork paradigm, as well as the collection of test data in the live network environment. This process of multi-phased forking was originally to be called “soft forking” but the community affectionately dubbed it “the spork” and the name stuck.

New features or versions of GoByte undergo extensive testing on testnet before they are released to the main network. When a new feature or version of GoByte is released on mainnet, communication is sent out to users informing them of the change and the need for them to update their clients. Those who update their clients run the new code, but it is not activated until a sufficient percentage of network participants (usually 80%) reach consensus on running it. In the event of errors occurring with the new code, the client’s blocks are not rejected by the network and unintended forks are avoided. Data about the error can then be collected and forwarded to the development team. Once the development team is satisfied with the new code’s stability in the mainnet environment – and once acceptable network consensus is attained – enforcement of the updated code can be activated remotely by multiple members of the core development team signing a network message together with their respective private keys. Should problems arise, the code can be deactivated in the same manner, without the need for a network-wide rollback or client update. For technical details on individual sporks, see [here](#).

1.2.7 NeoScript Hash Algorithm

NeoScript is a widely used hashing algorithm created by Feathercoin core developers. As the name suggests, NeoScript is a further development of Script. It is aimed at increased security and better performance on general purpose computer hardware while maintaining comparable costs and requirements.

Although a very innovative design back in time, Script has developed certain vulnerabilities. The first announced differential cryptanalysis of Salsa20/8 by Tsunoo in 2007 did not deliver any advantage over 256-bit brute force attack, but the following research by Aumasson reduced time complexity to break it from 2^{255} to 2^{251} with 50% success probability. It was improved by Shi in 2012 to 2^{250} . Although this is not critical yet, better attacks on Salsa20/8 may be developed in the future.

NeoScript addresses these issues. The core engine is configured to employ non-reduced Salsa20 of 20 rounds (Salsa20/20) as well as non-reduced ChaCha20 of 20 rounds (ChaCha20/20). Both of them are used to produce the final salt as their outputs are XOR’ed into it. They may be configured to run either in series or parallel depending on application objectives. The default NeoScript configuration is (128, 2, 1). There are no known successful attacks on non-reduced Salsa20 and ChaCha20 other than exhaustive brute force search. NeoScript replaces SHA-256 with BLAKE2s which is a further development of BLAKE-256, one of 5 NIST SHA-3 contest finalists.

Information on mining with NeoScript can be found in the [Mining](#) section of this documentation.

1.2.8 Dark Gravity Wave

DGW or *Dark Gravity Wave* is an open source difficulty-adjusting algorithm for Bitcoin-based cryptocurrencies that was first used in Dash and has since appeared in other digital currencies, including GoByte. DGW was authored by Evan Duffield, the developer and creator of Dash, as a response to a time-warp exploit found in *Kimoto’s Gravity*

Well. In concept, DGW is similar to the Kimoto Gravity Well, adjusting the difficulty levels every block (instead of every 2016 blocks like Bitcoin) based on statistical data from recently found blocks. This makes it possible to issue blocks with relatively consistent times, even if the hashing power experiences high fluctuations, without suffering from the time-warp exploit.

- Version 2.0 of DGW was implemented in Dash from block 45,000 onwards in order to completely alleviate the time-warp exploit.
- Version 3.0 was implemented on May 14 of 2014 to further improve difficulty re-targeting with smoother transitions. It also fixes issues with various architectures that had different levels of floating-point accuracy through the use of integers.
- Version 3.0 of DGW was implemented in GoByte from block 1 onwards.

1.2.9 Emission Rate

Cryptocurrencies such as GoByte and Bitcoin are created through a cryptographically difficult process known as mining. Mining involves repeatedly solving *hash algorithms* until a valid solution for the current *mining difficulty* is discovered. Once discovered, the miner is permitted to create new units of the currency. This is known as the block reward. To ensure that the currency is not subject to endless inflation, the block reward is reduced at regular intervals, as *shown in this calculation*. Graphing this data results in a curve showing total coins in circulation, known as the coin emission rate.

While GoByte is based on Bitcoin, it significantly modifies the coin emission rate to offer a smoother reduction in coin emission over time. While Bitcoin reduces the coin emission rate by 50% every 4 years, GoByte reduces the emission by one-twelfth (approx. 8.3333%) every 210240 blocks (approx. 364.25 days). It can be seen that reducing the block reward by a smaller amount each year offers a smoother transition to a fee-based economy than Bitcoin.

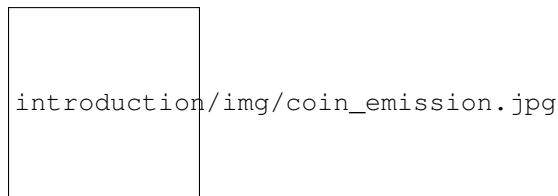


Fig. 1: Bitcoin vs. GoByte coin emission rate

Total coin emission

Bitcoin's *total coin emission* can be calculated as the sum of a geometric series, with the total emission approaching (but never reaching) 21,000,000 BTC. This will continue until 2140, but the mining reward reduces so quickly that 99% of all bitcoin will be in circulation by 2036, and 99.9% by 2048.

GoByte's *total coin emission* is also the sum of a geometric series, but the ultimate total coin emission is uncertain because it cannot be known how much of the 10% block reward reserved for budget proposals will actually be allocated, since this depends on future voting behavior. GoByte will continue to emit coins for approximately 34 years before a full block reward creates less than 1 GBX. Based on these numbers, a maximum and minimum possible coin supply in the year 2269 can be calculated to be between:

GBX | Assuming zero treasury allocation |

GBX | Assuming full treasury allocation |

Block reward allocation

Unlike Bitcoin, which allocates 100% of the block reward to miners, GoByte holds back 10% of the block reward for use in the decentralized *budget system*. The remainder of the block, as well as any transaction fees, are split 30/70 between the *miner* and a *masternode*, which is deterministically selected according to the *payment logic*. GoByte features superblocks, which appear every 17280 blocks (approx. 30.00 days) and can release up to 10% of the cumulative budget held back over that *budget cycle period* to the winning proposals in the budget system. Depending on budget utilization, this results in an approximate coin reward allocation over a budget cycle as follows:

25%	Mining Reward
65%	Masternode Reward for Proof-of-service
10%	Decentralized Governance Budget

See [this site](#) for live data on current network statistics.

1.2.10 Decentralized Governance

Decentralized Governance by Blockchain, or DGBB, is GoByte’s attempt to solve two important problems in cryptocurrency: governance and funding. Governance in a decentralized project is difficult, because by definition there are no central authorities to make decisions for the project. In GoByte, such decisions are made by the network, that is, by the owners of masternodes. The DGBB system allows each masternode to vote once (yes/no/abstain) for each proposal. If a proposal passes, it can then be implemented (or not) by GoByte’s developers. A key example is early in 2018, when GoByte’s Core Team submitted a proposal to the network asking whether the block reward should be changed. Within 24 hours, consensus had been reached to approve this change. Compare this to Bitcoin, where debate on the blocksize has been raging for nearly three years.

DGBB also provides a means for GoByte to fund its own development. While other projects have to depend on donations or premined endowments, GoByte uses 10% of the block reward to fund its own development. Every time a block is mined, 25% of the reward goes to the miner, 65% goes to a masternode, and the remaining 10% is not created until the end of the month. During the month, anybody can make a budget proposal to the network. If that proposal receives net approval of at least 10% of the masternode network, then at the end of the month a series of “superblocks” will be created. At that time, the block rewards that were not paid out (10% of each block) will be used to fund approved proposals. The network thus funds itself by reserving 10% of the block reward for budget projects.

You can read more about GoByte governance in the [Governance](#) section of this documentation.

1.2.11 Sentinel

Introduced in GoByte 0.12.1, Sentinel is an autonomous agent for persisting, processing and automating GoByte governance objects and tasks. Sentinel is implemented as a Python application that binds to a local version dashd instance on each GoByte masternode.

A Governance Object (or “govObject”) is a generic structure introduced in GoByte 0.12.1 to allow for the creation of Budget Proposals and Triggers. Class inheritance has been utilized to extend this generic object into a “Proposal” object to supplant the current GoByte budget system.

1.2.12 Fees

Transactions on the GoByte network are recorded in blocks on the blockchain. The size of each transaction is measured in bytes, but there is not necessarily a correlation between high value transactions and the number of bytes required to process the transaction. Instead, transaction size is affected by how many input and output addresses are involved, since more data must be written in the block to store this information. Each new block is generated by a miner, who

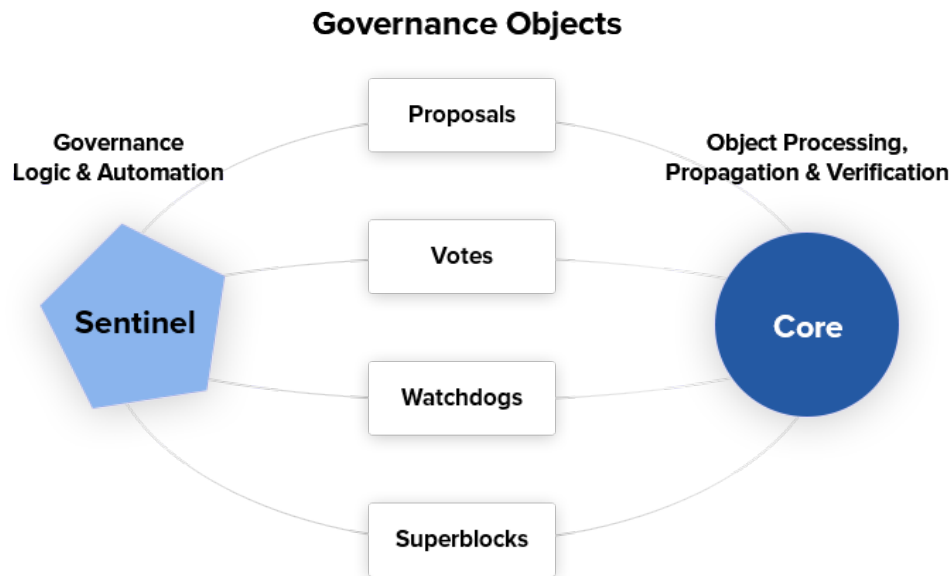


Fig. 2: Diagram highlighting the relationship between GoByte Sentinel and Core

is paid for completing the work to generate the block with a block reward. In order to prevent the network from being filled with spam transactions, the size of each block is artificially limited. As transaction volume increases, the space in each block becomes a scarce commodity. Because miners are not obliged to include any transaction in the blocks they produce, once blocks are full, a voluntary transaction fee can be included as an incentive to the miner to process the transaction. Most wallets include a small fee by default, although some miners will process transactions even if no fee is included.

The release of GoByte 0.12.2.0 and activation of DIP0001 saw a simultaneous reduction of fees by a factor of 10, while the block size was increased from 1MB to 2MB to promote continued growth of low-cost transactions even as the cost of GoByte rises. GoByte also supports *InstantSend* and *PrivateSend* transactions, which operate on a different and mandatory fee schedule. GoByte 0.13.0.0 introduced InstantSend autolocks, which causes masternodes to automatically attempt to lock any transaction with 4 or fewer inputs—which are referred to as “simple” transactions—and removes the additional fee for InstantSend. The current fee schedule for GoByte is as follows:

As an example, a standard and relatively simple transaction on the GoByte network with one input, one output and a possible change address typically fits in the range of 200 - 400 bytes. Assuming a price of US\$100 per GBX, the fee falls in the range of \$0.0002 - \$0.0004, or 1/50th of a cent. Processing a simple transaction using InstantSend at the same price is free of charge, while more complex InstantSend transactions may cost around 1-2 cents per transaction, depending on the number of inputs. These fees apply regardless of the GoByte or dollar value of the transaction itself.

PrivateSend works by creating denominations of 10, 1, 0.1, 0.01 and 0.001 GBX and then mixing these denominations with other users. Creation of the denominations is charged at the default fee for a standard transaction. Mixing is free, but to prevent spam attacks, an average of one in ten mixing transactions are charged a fee of 0.0001 GBX. Spending inputs mixed using PrivateSend incurs the usual standard or InstantSend fees, but to avoid creating a potentially identifiable change address, the fee is always rounded up to the lowest possible denomination. This is typically .001 GBX, so it is important to deduct the fee from the amount being sent if possible to minimise fees. Combining InstantSend and PrivateSend may be expensive due to this requirement and the fact that a PrivateSend transaction may require several inputs, while InstantSend charges a fee of 0.0001 GBX per input. Always check your fees before sending a transaction.

1.2.13 Evolution

GoByte Evolution (Dash Evo) is the code name for a decentralized platform built on GoByte blockchain technology. The goal is to provide simple access to the unique features and benefits of GoByte to assist in the creation of decentralized technology. GoByte introduces a tiered network design, which allows users to do various jobs for the network, along with decentralized API access and a decentralized file system.

GoByte Evolution will be released in stages. GoByte Core releases 0.12.1 through to 0.12.3 lay the groundwork for the decentralized features behind the scenes. Version 0.13 introduces the foundation of Evolution, specifically [DIP2 Special Transactions](#) and [DIP3 Deterministic Masternode Lists](#). Version 0.14 establishes [DIP6 Long Living Masternode Quorums](#). Expected in late 2021, GoByte Core 1.0 will introduce key Evolution features such as username-based payments, the world's first decentralized API (DAPI) and a decentralized data storage system (Drive) based on IPFS.

Included below is our current Dash work on Evolution, that adds many components such as:

- **Drive:** A decentralized shared file system for user data that lives on the second tier network
- **DAPI:** A decentralized API which allows third tier users to access the network securely
- **GoBytePay Decentralized Wallets:** These wallets are light clients connected to the network via DAPI and run on various platforms
- **Second Tier:** The masternode network, which provides compensated infrastructure for the project
- **Budgets:** The second tier is given voting power to allocate funds for specific projects on the network via the budget system
- **Governance:** The second tier is given voting power to govern the currency and chart the course the currency takes
- **Deterministic Masternode Lists:** This feature introduces an on-chain masternode list, which can be used to calculate past and present quorums
- **Social Wallet:** We introduce a social wallet, which allows friends lists, grouping of users and shared multisig accounts

Evolution Previews

The following videos featuring GoByte Founder Evan Duffield and Head of UI/UX Development Chuck Williams describe the development process and upcoming features of the GoByte Evolution platform.

Evolution Demo #1 - The First GoByte DAP, TBA

Evolution Demo #2 - Mobile Evolution, TBA

Evolution Demo #3 - GoBytePay User Experience, TBA

1.3 How To Buy

GoByte can be purchased and sold in several ways, each with different advantages and disadvantages. The following options are available:

- [Exchanges](#) are one of the most popular ways to trade cryptocurrency. A wide range of exchanges exist, each offering slightly different features. Some serve different markets, some are in direct competition, some have cheaper fees, and some are subject to more or less strict regulatory requirements. Most exchanges are centralized, meaning they are operated by a single company, which may be obliged by the laws of the jurisdiction in which it operates to collect data on its customers. Others are decentralized, but as a result have higher escrow requirements since you are dealing peer-to-peer instead of with a trusted entity. Exchanges can be broadly broken

down into two categories: exchanges which accept national currency (fiat money) and exchanges which deal in cryptocurrencies only. For safety, exchanges should not be used as wallets. Exchanges are for trading, not for savings.

- *Instant exchanges* perform a similar function to normal exchanges, but without the requirement to log in. They effectively convert one currency to another, with some limits on the amount to be exchanged and usually at a less advantageous rate. Others may even offer to sell cryptocurrency as a credit card purchase.
- *Over the counter* exchanges have recently appeared to facilitate sale of GoByte directly from a company to the individual at a specified price, or peer-to-peer between individuals at a negotiated price. Volume may be limited compared to exchanges, but these services are usually much easier to use. More advanced peer-to-peer sites offer escrow services for a fee to prevent cheating during the sale between two parties who have never met.
- *ATMs* accepting card and cash payments in return for crypto are widely available. Mapping services can show the specific location of these machines, or you can even set one up at your own business and earn a percentage of sales.

DISCLAIMER: This list is provided for informational purposes only. Services listed here have not been evaluated or endorsed by GoByte Core and no guarantees are made as to the accuracy of this information. Please exercise discretion when using third-party services.

1.3.1 Exchanges

Cryptocurrency exchanges exist to convert national currency, also known as fiat money, into cryptocurrency. Many exchanges do not accept fiat money, and exchange between various cryptocurrencies only. Trades are handled on markets, and trades are created between pairs of currencies, identified by their ticker codes. GoByte is widely accepted on exchanges and many pairs exist against both fiat money and cryptocurrency. This means it is possible to exchange EUR for GBX, or GBX for BTC, for example. The volume traded on an exchange provides a good indication of how quickly a buy or sell order you place will be filled. This section introduces some of the most popular exchanges for trading GoByte.

Marketplace comparison websites

Coinpaprika

<https://coinpaprika.com>

Coinpaprika lists hundreds of cryptocurrencies by their market capitalization, traded volume and recent price performance. A number of advanced features to research cryptocurrency projects and exchanges are also available.



Cryptoradar

<https://cryptoradar.co>

Cryptoradar is a real-time cryptocurrency marketplace price comparison and review platform. The website compares dozens of markets based on prices, fees, payment methods, reviews and more.



CoinMarketCap

<https://coinmarketcap.com>



CoinMarketCap lists all cryptocurrencies by their market capitalization. Clicking one of these currencies allows you to view price charts, and clicking Markets allows you to view the markets available and the trading pairs they offer.

GoByte.Network markets



<https://www.gobyte.network/exchanges>

The official GoByte website also provides a list of major exchanges offering GoByte.

List of exchanges

Please see [here](#) for a detailed guide on how to buy GoByte on an exchange. The exchanges listed here are for informational purposes only and do not indicate endorsement or affiliation with any particular platform.

HitBTC



<https://hitbtc.com>

HitBTC offers facilities to major investors to credit USD, EUR and GBP, as well as BTC, ETH and USDT trading pairs against GBX for normal users.

Sistemkoin



<https://sistemkoin.com>

Sistemkoin serves the Turkish market and offers trading pairs for GBX against the Turkish Lira and Bitcoin.

Cryptopia



<https://www.cryptopia.co.nz>

Cryptopia is a New Zealand cryptocurrency exchange with a reputation for supporting a large number of low-volume altcoins. It offers GBX trading pairs for BTC, LTC, DOGE and USDT.

Birake DEX



<https://trade.birake.com/>

Birake is a decentralized exchange serving worldwide and offering GBX trading pairs against BTC.



Block DX

<https://blocknet.co/block-dx/>

Every trade is peer-to-peer, wallet-to-wallet, with no middleman. Through the power of Atomic Swaps, you never have to trust a third party. Block DX offers GBX trading pairs against BTC.

1.3.2 Instant exchanges



Changelly

<https://changelly.com>

Changelly is a broker service offering a range of cryptocurrency, for instant exchange against other cryptocurrencies without needing to create an account. Be sure to check the fees and rates before purchasing.



ShapeShift

<https://shapeshift.io>

ShapeShift allows users to directly exchange one crypto asset for another, albeit with a higher markup than most exchanges. ShapeShift supports Bitcoin and over 70 other cryptocurrencies.



CoinSwitch

<https://coinswitch.co>

CoinSwitch is a crypto to crypto exchange aggregate with more than 300 different coins and tokens listed. Also offers purchases through credit/debit cards.



fox.exchange

<https://fox.exchange>

fox.exchange is a crypto to crypto exchange allowing instant exchange between BTC, LTC, BCH, BSV, ETH and others.



changeNOW

<https://changenow.io>

changeNOW is a non-custodian exchange service based in the Netherlands, with low commissions and quick service. Offers crypto to crypto exchanges, as well as purchases through credit/debit cards.

InstaSwap.io



<https://instaswap.io/>

InstaSwap is a crypto to crypto exchange aggregate with and more than 300 different coins and tokens listed. Also offers purchases through credit/debit cards.

1.3.3 Over the Counter



Coinbase

<https://www.coinbase.com>

Coinbase is a large US-based cryptocurrency exchange with a focus on making it easy to buy, sell and manage your cryptocurrency portfolio. With trading allowed between any of the 17+ cryptocurrencies supported and many major fiat currencies including EUR, USD and GBP, Coinbase is a great place to buy your first cryptocurrency.



Uphold

<https://uphold.com>

Uphold accounts may be funded with over 30 national currencies by bank account or credit card to purchase and spend multiple cryptocurrencies.



Bitpanda

<https://www.bitpanda.com> <https://www.bitpanda.com/togo>

Bitpanda is a broker service offering Bitcoin, Ethereum, Litecoin and Dash both online and at over 400 Post branches and about 1300 Post partners throughout Austria. Pay with cash, credit card or bank transfer.



Bitit

<https://bitit.io>

Bitit is a broker service offering Bitcoin, Ethereum and several other cryptocurrencies for sale online. Payment in a range of currencies is support using both direct banking, credit cards and vouchers.



Kraken

<https://www.kraken.com>

Kraken offers private, personalized OTC service with deep liquidity to institutions and high net-worth individuals needing to fill orders in excess of \$100,000. Simply send an email to otc@kraken.com to get started.



Easy Crypto

<https://www.easycrypto.nz>

Easy Crypto allows New Zealanders to buy and sell Bitcoin and 45 other cryptocurrencies instantly, with fast and friendly service.



Changelly

<https://changelly.com>

Changelly is a popular instantaneous crypto to crypto exchange platform with more than 100 different coins and tokens listed. Also offers purchases via credit/debit cards.



Stratum

<https://stratum.hk>

Stratum is a Brazil-based cryptocurrency company offering a variety of services including an exchange, mining, bill payment, point-of-sale, and more. Bitcoin is available for purchase at over 13,000 locations around Brazil.



AnycoinDirect

<https://anycoindirect.eu>

AnycoinDirect.eu is a broker service offering 14 cryptocurrencies, including Dash, for sale online. Pay by bank transfer or various national instant payment schemes.

1.3.4 ATMs

ATMs are a popular method of buying cryptocurrency at businesses to encourage adoption and spending in these currencies. A number of ATMs support GoByte, and the mapping services listed on this page can help you find one near you. It is also possible to operate your own ATM to sell GoByte on-site at your business - simply contact the companies listed on this page.

iVend Pay ATMs



<https://www.ivendpay.com>

iVend Pay offers a range of two-way cash ATM and Point of Sale solutions integrating GoByte.

1.4 Safety

If you are new to cryptocurrencies, the most important change to understand in comparison with the traditional banking system is that transactions occur **directly between two parties without any central authority** to facilitate the transaction. This also means that **you are responsible for your own security** - there is no bank or credit card company to reverse a transaction if your funds are stolen or lost. If you forget or lose your wallet file, recovery phrase or PIN, you will permanently and irrevocably lose access to your funds.

GoByte is designed from the ground up to be fast, secure, fungible and private. In this sense, it is similar to cash or gold, but cryptocurrency can be spent locally and internationally with equal ease, if you are confident you are sending funds to the right destination. For these reasons, the GoByte documentation has a strong focus on safety and understanding the concepts and features that drive the GoByte ecosystem.

A few general safety guidelines:

- Do not trust any online service or person because they sound or look reputable. Always use an escrow service if you are buying peer-to-peer.
- Store your GoByte on a *hardware wallet* if possible. If not, then store your coins in the official *GoByte Core Wallet* or the official *GoByte Electrum Wallet*.
- Do not use exchanges as wallets. Exchanges are for trading, not for savings.
- Mobile wallets should be used for day-to-day purchases, but do not keep large amounts of funds in them. Transfer funds as necessary.

A list of known scams, fake wallets and Ponzi or pyramid schemes can be seen below. Do NOT trust them.

1.4.1 Impersonation

Scammers may attempt to impersonate well-known community members and manipulate you into granting them access to your system or wallets. This is usually done via private messages on the forum, Discord or email. The attacks are often targeted against masternode owners. If you require technical assistance, it is best to ask in a public channel/forum or go to <https://support.gobyte.network> and open a ticket. If you engage in personal chat with a well-known community member, verify their identity by their chat history or through their publicly available cryptographic keys. All community members and *GoByte Core Group staff* will be able to verify their identity using signed PGP messages. Identities can also be verified on Keybase:

- [Antonio Moratti](#)

1.4.2 Scams

There are many “fake” GoByte pages on the internet attempting to trick users into sending GoByte or other cryptocurrencies or “open a wallet”. Other scams include selling fake mining hardware, fake GoByte or altcoins with a similar name, and Ponzi schemes (see below). Please be careful and do NOT trust any third parties listed here!!

List of known GoByte-related scams:

- **gobyte-wallet dot com** is a known scam!

- **electrumgobyte dot org** is a fake clone of the official site!
- **gobytecoinmining dot com** is not affiliated with GoByte!
- **gobytecrypto dot info** is not affiliated with GoByte!
- **oncloud dot com** is not affiliated with GoByte!
- **as-shop dot su** is selling fake Baikal miners!
- **minershop dot biz** is selling fake Baikal miners!
- **gobytecoinclub dot com** is a Ponzi scheme not affiliated with GoByte!
- **gobyte-coin dot net** is a fake web wallet, do not send them money!
- **coinvert dot io** is a fake exchange!
- **gobytecash dot io** is not affiliated with GoByte and may be distributing a compromised wallet!
- **gobytedaowallet dot com** is a mydashwallet clone and confirmed scam

Beware of fake Twitter accounts impersonating GoByte! The official Twitter account is: <https://twitter.com/gobytenetwork>

Please report these and any others scams you encounter as follows:

1. Report phishing and scams to Google: https://www.google.com/safebrowsing/report_phish
2. Look up the registrar of the domain and send a complaint: <https://www.whois.com/whois>
3. Report phishing to Netcraft: <https://www.netcraft.com>
4. Report scams to the BadBitcoin Project: <http://www.badbitcoin.org>
5. If in doubt, use Crypto Scam Checker to see if already report and report there as well: <https://fried.com/crypto-scam-checker>

1.4.3 Ponzi Schemes

A **Ponzi scheme**, **Pyramid scheme** or **Multi-level marketing** are a fraudulent investment operations where the operator provides fabricated reports and generates returns for older investors through revenue paid by new investors. More and more users must constantly join the scheme in order for it to continue operation, with ever greater numbers of people losing money to the originators of the scheme.

- [What is a Pyramid Scheme?](#)
- [How to spot a Ponzi Scheme](#)
- [BehindMLM - News and blog about Ponzi schemes](#)

If you encounter a Ponzi scheme, follow the same reporting steps as above for scam websites!

List of known Ponzi schemes (there are many more - stay vigilant!):

OneCoin

- <http://themerple.com/dr-ruja-flees-sinking-ship-as-regulators-crack-down-on-onecoin/>
- <http://siliconangle.com/blog/2016/09/29/dodgy-cryptocurrency-onecoin-under-police-investigation-accused-of-being-a-ponzi-scheme/>
- <https://cointelegraph.com/news/one-coin-much-scam-onecoin-exposed-as-global-mlm-ponzi-scheme>
- <http://www.makemoneyexpert.com/online/network-marketing/reviews/onecoin/>
- <https://pageone.ng/2016/11/05/beware-onecoin-ponzi-scheme/>

SwissCoin

- <http://behindmlm.com/mlm-reviews/swisscoin-review-25-to-15000-eur-ponzi-points-investment/>
- <http://ethanvanderbuilt.com/2017/01/26/swisscoin-scam-warning/>
- <https://news.bitcoin.com/dissecting-swisscoin-cryptocurrency-ponzi-horizon/>

The Billion Coin

- <https://steemit.com/news/@rahmat/review-the-billion-coin-ponzi-scheme>
- <https://coins.newbium.com/post/728-scam-alert-the-billion-coins-scam-ponzi-scheme>
- <https://bitcointalk.org/index.php?topic=1592288.0>

Sustaincoin

- <http://www.scamvoid.com/check/sustaincoin.com>

E-Dinar

- <http://behindmlm.com/mlm-reviews/e-dinar-review-edr-unit-ponzi-points-cryptocurrency/>
- <https://www.scam.com/showthread.php?714218-E-dinar-coin>
- <https://bitcointalk.org/index.php?topic=1569896.0>

DasCoin

- <http://behindmlm.com/mlm-reviews/coin-leaders-review-dascoin-is-a-onecoin-ponzi-points-clone/>
- <https://bitcointalk.org/index.php?topic=1636850.0>

BitConnect

- https://www.reddit.com/r/Bitconnect/comments/76fa9k/bitconnect_investigated_as_a_ponzi_scheme/
- <https://www.youtube.com/watch?v=6fujWfmgRJu>
- <http://www.binaryoptionsarmy.com/2017/11/bitconnect-scam-review/>
- <https://satoshiwatch.com/hall-of-shame/bitconnect-coin/>

HashOcean

- <http://themerple.com/bitcoin-scam-risk-warning-hashocan/>

CryptoDouble

- <http://themerple.com/bitcoin-hyip-ponzi-scheme-alert-coindouble/>

1.5 Links and Information

1.5.1 Links

Official sites

- **Website:** <https://www.gobyte.network>
- **User documentation:** <https://docs.gobyte.network>
- **GoByte Core Documentation:** <https://gobytecore.readme.io> <TO SET UP>
- **GoByte Platform Documentation:** <https://gobyteplatform.readme.io> <TO SET UP>
- **Foundation:** <https://www.gobytefoundation.org>

- **GitHub:** <https://github.com/gobytecoin/>
- **GitHub (Evolution):** <https://github.com/gobyteevo>
- **Roadmap:** <https://www.gobyte.network/roadmap>
- **Dash DIPs:** <https://github.com/dashpay/dips>

Community sites

- <https://devs.gobyte.network> <TO SET UP>
- <https://www.nexus.gobytenation.org> <TO SET UP>
- <https://www.central.gobytenation.org> <TO SET UP>
- <https://masternodes.gobyte.network/>
- <https://www.gobytenews.org> <TO SET UP>

Forums

- **GoByte Forum:** <https://community.gobyte.network/>
- **BitcoinTalk thread:** <https://bitcointalk.org/index.php?topic=2442185.0>
- **Cryptocurrencytalk.com:** #To-do
- **(8BTC) Forum:** #To-do
- **(Baidu Tieba):** #To-do
- **(CYBTC GoByte):** #To-do

Chat

- **GoByte Official Discord:** <https://discord.gg/vRbVszC>
- **GoByte Nation Discord:** <https://discord.gg/ExSU77y>
- **GoByte Dapp Devs Discord:** #To-do
- **GoByte English Telegram:** <https://t.me/joinchat/Hd8RBU4JkNUbCrZDwbMjWw>
- **GoByte Russia Telegram:** <https://t.me/gobyteRU>
- **GoByte Romanian Telegram:** <https://t.me/gobyteRO>
- **GoByte en Español Telegram:** <https://t.me/gobyteES>
- **GoByte Dutch Telegram:** <https://t.me/GobyteNL>
- **GoByte South Africa Telegram:** <https://t.me/gobyteZA>
- **GoByte Korea Telegram:** <https://t.me/gobyteKR>
- **GoByte China Telegram:** <https://t.me/gobyteCN>
- **GoByte Turkey Telegram:** <https://t.me/gobyteTR>
- **GoByte Telegram News:** <https://t.me/gbxnews>
- **QQ GBX.China:** #To-do
- **Freenode IRC:** #To-do

Social media

- **Reddit:** <https://www.reddit.com/r/gobytenetwork>
- **Twitter:** <https://twitter.com/gobytenetwork>
- **Steemit:** #To-do
- **LinkedIn:** <https://www.linkedin.com/company/gobytenetwork/>
- **YouTube:** https://www.youtube.com/channel/UC2wlaXQJm6pwRfzl6xoRhA?view_as=subscriber
- **Instagram:** <https://www.instagram.com/gobytenetwork/>
- **Dailymotion:** #To-do
- **Youku:** #To-do
- **Soundcloud:** #To-do

Facebook

- **English (Official):** <https://www.facebook.com/gobytenetwork>
- **Español:** #To-Do
- **Brazil:** #To-Do
- **Denmark:** #To-Do
- **Germany:** #To-Do
- **Poland:** #To-Do
- **Russia:** #To-Do
- **Thailand:** #To-Do
- **Venezuela:** #To-Do
- **Vietnam:** #To-Do

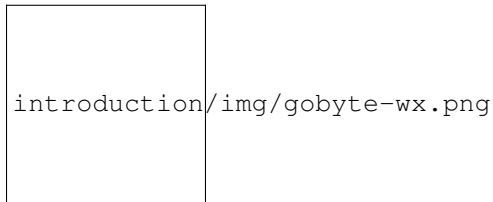
Twitter

- **GoByte Official Account:** <https://twitter.com/gobytenetwork>
- **Antonio Moratti, Founder of GoByte Core Group:** <https://twitter.com/AntonioMoratti1>
- **GoByte News:** #To-do
- **Brad Nickel, Business Development Manager:** <https://twitter.com/b05crypto>
- **GoByte Japan:** #To-do
- **GoByte Vietnam:** #To-do

News

- **GoByte News:** #To-do
- **GoByte News (YouTube):** #To-do
- **Cointelegraph:** <https://cointelegraph.com/tags/gobyte>

- **(8BTC): #To-do**
- **GoByte News China (Wechat):** gobytenews (or scan QR below)



Blogs

- **MEDIUM Blog:** <https://medium.com/gobytenetwork>

Wikipedia

- [https://en.wikipedia.org/wiki/GoByte_\(cryptocurrency\)](https://en.wikipedia.org/wiki/GoByte_(cryptocurrency))

Inactive

- **Bitcoin.com forum:** <https://bitcointalk.org/index.php?topic=2414021.0>

1.5.2 Tools

Block explorers, statistics and visualizations

- <https://explorer.gobyte.network>
- <https://insight.gobyte.network/>
- <https://blockchain.gobyte.network/>
- <https://masternodes.gobyte.network/>
- <https://altnix.org/coins/16-GoByte/explorer>

Treasury tools

- <https://central.gobytenation.org>
- <https://nexus.gobytenation.org>
- <https://proposal.gobyte.network>
- <https://masternodes.gobyte.network/governance.html>

Masternode management

- <https://masternodes.gobyte.network/>
- <https://masternodes.online/currencies/GBX/>
- <https://github.com/gobytecoin/gobyte-masternode-tool>

- <https://masternodes.pro/stats/gbx/statistics>
- <https://masternodesolutions.online/>

Price monitoring and statistics

- <https://coinmarketcap.com/currencies/gobyte>
- <https://www.coingecko.com/en/coins/gobyte>
- <https://bitinfocharts.com/gobyte/>
- <https://www.cryptonator.com/widget>

GoByte Community project

- #
- #

DarkNet pages

- **GoByte (Mirror of Main Page):** #To-do
- **The Hidden Wiki:** #To-do

1.5.3 Mobile Apps

iOS

- **GoByte Wallet:** #To-do
- **Coinomi:** <https://itunes.apple.com/app/id1333588809>
- **CoinCap:** <https://itunes.apple.com/app/id1074052280>
- **Blockfolio:** <https://itunes.apple.com/app/id1095564685>

Android

- **GoByte Wallet:** #To-do
- **Coinomi:** <https://play.google.com/store/apps/details?id=com.coinomi.wallet>
- **CoinCap:** <https://play.google.com/store/apps/details?id=io.coinCap.coinCap>
- **Blockfolio:** <https://play.google.com/store/apps/details?id=com.blockfolio.blockfolio>

1.5.4 Glossary

51% Attack A condition in which more than half the computing power on a cryptocurrency network is controlled by a single miner or group of miners. That amount of power theoretically makes them the authority on the network. This means that every client on the network believes the attacker's hashed transaction block.

Address A GoByte address is used to *Send/Receive a Payment* on the GoByte network. It contains a string of alphanumeric characters, but can also be represented as a scannable QR code. A GoByte address is also the public key in the pair of keys used by GoByte holders to digitally sign transactions (see Public key).

Algorithm In mathematics and computer science, an **algorithm** is a self-contained step-by-step set of operations to be performed. Algorithms perform calculation, data processing, and/or automated reasoning tasks.

Altcoin Since Bitcoin was the first cryptocurrency and has the largest market capitalization, it is considered as the reference. An altcoin, or alternative coin, is any cryptocurrency other than Bitcoin.

AML Anti-Money Laundering techniques are used to stop people from making illegally obtained funds appear as though they have been earned legally. AML mechanisms can be legal or technical in nature. Regulators frequently apply AML techniques to GoByte exchanges.

API In computer programming, an **application programming interface (API)** is a set of routines, protocols, and tools for building software and applications.

An API expresses a software component in terms of its operations, inputs, outputs, and underlying types, defining functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. A good API makes it easier to develop a program by providing all the building blocks, which are then put together by the programmer.

ASIC An application-specific integrated circuit (ASIC), is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. For example, a chip designed to run in a digital voice recorder or for *high-efficiency GoByte mining* is an ASIC.

ATM / BTM A GoByte ATM is a physical machine that allows a customer to buy GoByte with cash. There are many manufacturers, some of which enable users to sell GoByte for cash. They are also sometimes called ‘BTMs’ or ‘GoByte AVMS.’ GoByte is supported on several *ATMs*.

Backlog Backlog generally refers to an accumulation over time of work waiting to be done or orders to be fulfilled.

Backup The process of making copies of a computer file to ensure its integrity in case of loss, theft, or damage. GoByte allows users to *make backup copies* of their digital wallets. This protects against losing one’s money in the event of a computer crashing or losing one’s mobile device. This would be the equivalent of being able to backup the cash in your wallet, so that if you lost it, you could restore the cash from a backup.

Bitcoin 2.0 This is a term explaining the next new level of Bitcoin projects which started as a fork of Bitcoin but extended their code into the next level of Blockchain Projects (Smart Contracts, Decentralised Voting, ...)

Blockchain A **blockchain** is a distributed database that maintains a continuously-growing list of data records hardened against tampering and revision. It consists of data structure blocks — which exclusively hold data in initial blockchain implementations, and both data and programs in some of the more recent implementations — with each block holding batches of individual transactions and the results of any blockchain executables. Each block contains a timestamp and information linking it to a previous block.

Blocks Transactions on the Blockchain are collected in “**blocks**” which record and confirm when and in what sequence transactions enter and are logged in the block chain. Blocks are created by users known as “miners” who use specialized software or equipment designed specifically to create blocks.

Budget System / DGBB The development of GoByte and the GoByte ecosystem is self-funded by the network. Each time a block is discovered, 25% of the block reward goes to miners and 65% goes to masternodes. Ten percent is withheld by the network and used to fund projects that are approved by the masternode network. This process is known as *Decentralized Governance by Blockchain* (DGBB). For a fee, anybody can submit a proposal to the network, and will be paid directly by the blockchain if approved by the masternodes. The Budget System is sometimes called the Treasury System; the two terms are interchangeable.

ChainLock Defined in **DIP8**, ChainLocks are a method of using an LLMQ to threshold sign a block immediately after it is propagated by the miner in order to enforce the first-seen rule. This is a powerful method of mitigating 51% mining attacks, which are associated with double spending.

Cloud Wallet Third parties that will store your GoByte on their servers for you, so that you can access your funds from any device connected to the internet. If their website is hacked or if their servers are damaged, you run the risk of losing your GoByte. Any online wallets should be secured with strong passphrases and 2FA. You cannot make backup copies of your online wallet, because you do not have access to the private keys. We do not recommend that you store large quantities of funds in online wallets.

Coinbase transaction The first transaction in a block. Always created by a miner, it includes a single input which constitutes the block reward. This is split between the miner and a deterministically chosen masternode.

Cold Storage A method of generating and storing private keys completely offline. One could use a desktop or laptop computer disconnected from the internet, a dedicated hardware wallet, a USB stick, or a *paper wallet*.

Confirm(ed) Transaction When a GoByte transaction is made, a miner must verify that the transaction is valid. When the inputs and outputs are verified, the transaction is included in a block in the blockchain. The transaction can then be considered complete and irreversible. The confirmation number increases as more blocks are added to the blockchain.

Confirmation Number The number of confirmations for a specific GoByte transaction. Zero confirmations means that the **transaction is unconfirmed**. One confirmation means that the transaction is included in the latest block in the blockchain. Two confirmations means the transaction is included in two blocks, three confirmations for three blocks, and so on. The probability of a transaction being reversed (double spent) diminishes exponentially with every block and subsequent confirmation. Six confirmations is usually considered “safe” and irreversible.

Confirmed Transactions Transactions that are processed by miners and considered irreversible, usually after six confirmations. In the case of InstantSend, funds can be considered irreversible after a few seconds, but must still be written to the blockchain (and thus “confirmed”).

CPU A **central processing unit (CPU)** is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions. The term has been used in the computer industry at least since the early 1960s. Traditionally, the term “CPU” refers to a processor, more specifically to its processing unit and control unit (CU), distinguishing these core elements of a computer from external components such as main memory and I/O circuitry.

Cryptocurrency A **cryptocurrency** (or crypto currency or crypto-currency) is a medium of exchange using cryptography to secure the transactions and to control the creation of new units.

Cryptography Cryptography or cryptology (from Greek *κρυπτός* *kryptós*, “hidden, secret”; and *γραφειν* *graphein*, “writing,” or *-λογία* *-logia*, “study,” respectively) is the practice and study of techniques for secure communication in the presence of third parties called adversaries. More generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages; various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

DAP Decentralized Application Protocol. This term describes an application running on top of the GoByte DAPI platform.

DAP Client An HTTP Client that connects to DAPI and enables GoByte blockchain users to read and write data to their DAP Space.

DAP Schema A GoByte Schema document extending the GoByte System Schema to define consensus data and rules within a DAP contract.

DAP Space The part of a DAP State that is owned by a specific blockchain user. Data in a DAP Space can only be changed by the owner.

DAP State The total set of data stored in a DAP. This data consists of user DAP Spaces.

DAPI Decentralized Application Programming Interface. See above for a definition of API. DAPI will perform the same functions as an API, but with quorums of masternodes acting as the endpoints for API communication.

Dark Gravity Wave In concept, *Dark Gravity Wave (DGW)* is similar to *Kimoto Gravity Well*, adjusting the difficulty levels every block (instead of every 2016 blocks like Bitcoin) by using statistical data of the last blocks found. In this way block issuing times can remain consistent despite fluctuations in hashpower. However it doesn't suffer from the time-warp exploit.

GoByte GoByte is a portmanteau of “Digital Cash.” GoByte is an open source peer-to-peer cryptocurrency that solves many of Bitcoin's problems. GoByte's features include PrivateSend, InstantSend, Decentralized Governance by Blockchain (DGBB), a 2nd tier network (referred to as the masternode network). See the [Features](#) page for a full list of GoByte's features.

GoByteDrive GoByte network data storage backend service used by masternodes for off-chain data relating to Evolution. GoByteDrive implements [IPFS](#), a type of distributed file storage system.

GoByte Client GoByte clients are software programs used to interface with the GoByte network. They store the private keys needed to conduct GoByte transactions as well as a copy of the entire blockchain. A GoByte client connects to the GoByte network and becomes a node in the network. A node shares and propagates new transactions with the rest of the network, creating a robust decentralized infrastructure.

GoByte Core Wallet The *GoByte Core Wallet* (known also as the QT wallet) is the “official” GoByte wallet that is compiled by the GoByte Core Team and allows both PrivateSend and InstantSend. The GoByteCore wallet will download the entire blockchain and serve it over the internet to any peers who request it.

GoByte Evolution This is a 3 tier network GoByte/Dash developers are presently building. It will make GoByte/Dash as easy to use as PayPal, while still remaining decentralized. See the [Evolution](#) page for more information.

GoByte Schema A JSON-based language specification for defining and validating consensus data in Evolution.

DDoS A distributed denial of service attack uses large numbers of computers under an attacker's control to drain the resources of a central target. They often send small amounts of network traffic across the Internet to tie up computing and bandwidth resources at the target, which prevents it from providing services to legitimate users. GoByte exchanges have sometimes been hit with DDoS attacks.

Decentralized [Decentralized computing](#) is the allocation of resources, both hardware and software, to each individual workstation or office location. In contrast, centralized computing exists when the majority of functions are carried out or obtained from a remote centralized location. Decentralized computing is a trend in modern-day business environments. This is the opposite of centralized computing, which was prevalent during the early days of computers. A decentralized computer system has many benefits over a conventional centralized network. Desktop computers have advanced so rapidly that their potential performance far exceeds the requirements of most business applications. This results in most desktop computers remaining nearly idle most of the time. A decentralized system can use the potential of these systems to maximize efficiency. However, it is debatable whether these networks increase overall effectiveness.

Desktop Wallet A wallet is a piece of software that stores your GoByte. There are many different wallet options, but it is imperative to choose a secure one. We recommend any of the following: [GoByte Core Wallet](#) / [GoByte Electrum Wallet](#) / [Hardware Wallets](#)

Difficulty This number determines how difficult it is to hash a new block. It is related to the maximum allowed number in a given numerical portion of a transaction block's hash. The lower the number, the more difficult it is to produce a hash value that fits it. Difficulty varies based on the amount of computing power used by miners on the GoByte network. If large numbers of miners leave a network, the difficulty would decrease. GoByte's increasing popularity and the availability of specialized FPGA miners have caused the difficulty to increase over time.

Digital Wallet See [this link](#) for full documentation on wallets.

A digital wallet is similar to a physical wallet except that it is used to hold **digital currency**. A GoByte wallet holds your private keys, which allow you to spend your GoByte. You are also able to make backups of your

wallet in order to ensure that you never lose access to your GoByte. Digital wallets can exist in many different forms and on many devices:

- **Desktop Wallet** (*GoByte Electrum Wallet, GoByte Core Wallet*): Wallet programs that you install on a laptop or desktop computer. You are solely responsible for protecting the wallet file and the private keys it contains. Make backup copies of your wallet files to ensure that you don't lose access to your funds.
- **Mobile Wallet** (*Android, iOS*): These wallets can be downloaded through Google Play or Apple (iTunes) App Stores. Mobile wallets allow you to use GoByte on-the-go by scanning a QR code to send payment. Make backup copies of your mobile wallet files to ensure that you don't lose access to your funds. Due to security issues with mobile phones, it is advised that you don't store large amounts of funds on these wallets.
- **Online/Cloud/Web Wallet** (*Exodus, MyGoByteWallet*): Third parties that will store your GoByte on their servers for you or provide an interface to access your GoByte with you providing the keys, so that you can access your GoByte from any device connected to the internet. If their website is hacked or if their servers are damaged, you run the risk of losing your GoByte. Any online wallets should be secured with strong passphrases and 2FA. You cannot make backup copies of your online wallet, because you do not have access to the private keys. We strongly urge that you NEVER store large amounts of GoByte in any online wallet or cryptocurrency exchange.
- **Hardware Wallets** (*Trezor, KeepKey, Ledger, Nano*): A hardware wallet is a specialized, tamper-proof, hardware device that stores your private keys. This device is able to sign transactions with your private key without being connected to the internet. However, you must have an internet connection to send the transaction to the GoByte network. This allows your private keys to be accessed easily while still keeping them securely protected. This is widely regarded to be the safest form of storage for your GoByte.
- **Offline/Cold Storage** (*Paper wallet*): A special wallet that is created offline and is never exposed to the internet. Accomplished by using software to generate a public and private key offline and then recording the generated keys. They keys can be printed out on paper or even laser-etched in metal. Copies can be made and stored in a personal safe or bank deposit box. This is an extremely secure way to store GoByte. There is no risk of using software wallet files, which can become corrupt, or web wallets, which can be hacked. NOTE: USB sticks are not safe for long-term (multi-year) storage because they degrade over time.

DKG Defined in [DIP6](#), Distributed Key Generation (**DKG**) is a method of generating a BLS key pair for use in an LLMQ to perform threshold signing on network messages. It is based on BLS M-of-N Threshold Scheme and Distributed Key Generation, which is an implementation of Shamir's Secret Sharing.

Digital Signature A digital signature is a mathematical mechanism that allows someone to prove their identity or ownership of a digital asset. When your digital wallet signs a transaction with the appropriate private key, the whole network can see that the signature matches the address of the GoByte being spent, without the need to reveal the private key to the network. You can also digitally sign messages using your private key, to prove for instance that you are the owner of a certain GoByte address.

Electrum Wallet *GoByte Electrum Wallet* is a lightweight wallet that does not require you to download or sync the entire blockchain, making the wallet lighter and faster. However, it is missing certain features such as PrivateSend and InstantSend.

Encryption In cryptography, [encryption](#) is the process of encoding messages or information in such a way that only authorized parties can read it. Encrypted messages which are intercepted by a third-party are indecipherable gibberish without the private key. In an encryption scheme, the *plaintext* message is encrypted using an encryption algorithm, generating *ciphertext* that can only be read if decrypted by the intended recipient. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. Increases in computing power have "broken" many past encryption algorithms, but a well-designed modern system such as AES-256 is considered essentially "uncrackable."

Escrow Services An [escrow](#) is:

- a contractual arrangement in which a third party receives and disburses money or documents for the primary transacting parties, with the disbursement dependent on conditions agreed to by the transacting parties; or
- an account established by a broker for holding funds on behalf of the broker's principal or some other person until the consummation or termination of a transaction; or
- a trust account held in the borrower's name to pay obligations such as property taxes and insurance premiums.

A trusted escrow service is often used when purchasing cryptocurrency or other goods/services over the internet. Both the buyer and seller will choose a trusted third-party, the seller will send the item (or currency) to the escrow agent, and the buyer will send the purchasing funds to the escrow agent as well. Once the escrow agent is satisfied that both parties have satisfied the terms of the agreement, he/she will forward the funds and the product (or currency) being purchased to the appropriate party.

Evan Duffield Founder and first Lead Developer of Dash. Inventor of X11, InstantSend and PrivateSend. Before creating Dash, Evan was a financial advisor and holds a Series 65 license.

Antonio "Moratti" Mladin Co-Founder of GoByte and GoByte Nation DAO Founder. Before creating GoByte, Antonio was a Payment Processing/Balancing Manager and Back-Office Specialist.

Exchange The current price of one GoByte compared to the price of other currencies, like the US dollar, Yen, Euro, or Bitcoin. Because most trading volume takes place on the BTC/GBX markets, price is often quoted in fractions of a bitcoin. For instance, the price of one GoByte at the end of July 2020 was 0.00000315 (bitcoins per GoByte). An excellent site for following the exchange rate of GoByte is [CoinMarketCap](#). Businesses wishing to reduce the risk of holding a volatile digital currency can avoid that risk altogether by having a payment processor do an instant exchange at the time of each transaction.

Faucet Faucets are a reward system, in the form of a website or app, that dispenses rewards in the form of a microGBX or gbit, which is a hundredth of a millionth GoByte, for visitors to claim in exchange for completing a captcha or task as described by the website.

Fiat Gateway [Fiat money](#) has been defined variously as:

- Any money declared by a government to be legal tender.
- State-issued money which is neither convertible by law to any other thing, nor fixed in value in terms of any objective standard.
- Intrinsically valueless money used as money because of government decree.

Examples include the US dollar, the Euro, the Yen, and so forth.

Fintech [Financial technology](#), also known as FinTech, is an economic industry composed of companies that use technology to make financial services more efficient. Financial technology companies are generally startups trying to make financial processes more efficient or eliminate middle-men. Recently many fintech companies have begun utilizing blockchain technology, which is the same technology that underpins GoByte and Bitcoin.

Fork When the blockchain diverges or splits, with some clients recognizing one version of the blockchain as valid, and other clients believing that a different version of the blockchain is valid. Most forks resolve themselves without causing any problems, because the longest blockchain is always considered to be valid. In time, one version of the blockchain will usually "win" and become universally recognized as valid. Forks can, however, be extremely dangerous and should be avoided if possible.

Forking is most likely to occur during software updates to the network. GoByte uses a Multi-Phased Fork ("[Spork](#)") system for greater flexibility and safety.

Full Nodes Any GoByte client that is serving a full version of the blockchain to peers. This can be a user running a GoByte Core wallet on his/her desktop, or it could be a [masternode](#). Full nodes promote decentralization by allowing any user to double check the validity of the blockchain.

Fungible Every unit of the currency is worth the same as any other unit.

Genesis Block The very first block in the block chain.

GPU A [graphics processing unit \(GPU\)](#), also occasionally called visual processing unit (VPU), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing, and their highly parallel structure makes them more efficient than general-purpose CPUs for algorithms where the processing of large blocks of data is done in parallel. In a personal computer, a GPU can be present on a video card, or it can be embedded on the motherboard or — in certain CPUs — on the CPU die. Certain cryptocurrencies use mining algorithms which are most efficiently run on GPUs.

Hardware Wallet [Hardware wallets](#) are among the safest type of wallet for storing your GoByte. Your private key is protected inside a piece of hardware, and is never exposed to the internet. You are still able to sign transactions as normal, making it both safe and convenient.

Hash A mathematical process that takes a variable amount of data and produces a shorter, fixed-length output. A hashing function has two important characteristics. First, it is mathematically difficult to work out what the original input was by looking at the output. Second, changing even the tiniest part of the input will produce an entirely different output.

Hashrate The number of hashes that can be performed by a GoByte miner in a given period of time (usually a second).

Insight Blockchain information server used to power block explorers and respond to transaction queries.

InstantX See InstantSend

InstantSend [InstantSend](#) technology uses the masternode network to “lock” transaction inputs, preventing GoByte from being double-spent. Unlike Bitcoin, where it takes an hour or longer for transactions to fully confirm, transactions using InstantSend are “locked” and irreversible after only a few seconds.

Liquidity The ability to buy and sell an asset easily, with pricing that stays roughly similar between trades. A suitably large community of buyers and sellers is important for liquidity. The result of an illiquid market is price volatility, and the inability to easily determine the value of an asset.

LLMQ Defined in [DIP6](#), A Long- Living Masternode Quorum (LLMQ) is a deterministic subset of the global deterministic masternode list. Such a quorum is formed with the help of a distributed key generation (DKG) protocol and is supposed to be active for a long time (e.g. days). Multiple quorums are kept alive at the same time, allowing load balancing between these quorums. The main task of a LLMQ is to perform threshold signing of consensus related messages.

Masternode A [masternode](#) is special type of full node that performs services for the network and is paid a portion of the block reward. Masternodes require proof of ownership of 1000 GBX.

Masternodes serve as the second tier of the GoByte network, and power InstantSend, PrivateSend, the Budget System.

Mining [Miners](#) process transactions on the GoByte network and publish them on the blockchain. As a reward for doing this, miners are paid 25% of the block reward.

Mobile Wallet These are wallets available on mobile devices (iOS + Android).

MultiSig Multi-signature addresses provide additional security by requiring multiple people to sign a transaction with their private key before the transaction can be sent. For example, in [2 of 3 multisig](#), two out of three possible signatories have to sign a transaction for it to be processed. Multi-signature addresses are commonly used by exchanges and other organizations that are in possession of large sums of cryptocurrency, since it makes theft much more difficult.

Node A node is any device running GoByte wallet software. Full nodes are software clients that have downloaded the entire blockchain and serve it to other clients on GoByte’s peer-to-peer network.

OTC Over the counter (OTC) trades are trades that occur off exchanges. In an OTC trade, a buyer and seller trade with each other directly, or through an intermediary. OTC trading is useful when a person wants to either buy or sell a large amount of cryptocurrency and is afraid that a large buy or sell order will move the price (called “slippage”).

P2P Peer-to-peer. Decentralized interactions that happen between at least two parties in a highly interconnected network. An alternative system to a ‘hub-and-spoke’ arrangement, in which all participants in a transaction deal with each other through a single mediation point.

Paper Wallet *Paper wallets* are offline wallets, printed on paper for safety. If properly secured and stored they are considered the safest way to store cryptocurrency.

Privacy *Privacy* is the ability of an individual or group to seclude themselves, or information about themselves, and thereby express themselves selectively. The boundaries and content of what is considered private differ among cultures and individuals, but share common themes. When something is private to a person, it usually means that something is inherently special or sensitive to them. The domain of privacy partially overlaps security (confidentiality), which can include the concepts of appropriate use, as well as protection of information. GoByte includes PrivateSend, which allows users to maintain financial privacy.

Private Key A *private key* is a long alphanumeric passcode that allows GoByte to be spent. Every GoByte wallet contains one or more private keys which are saved in the wallet file. The private keys are mathematically related to all GoByte addresses generated for the wallet. Because the private key is the “ticket” that allows someone to spend GoByte, it is important that these are kept secure and secret.

PrivateSend *PrivateSend* obscures the source of funds in order to maintain financial privacy between users. It can be turned on or off at the users’ discretion.

Proof of Service - PoSe Consensus mechanism used in GoByte to verify that a masternode has provided uninterrupted service meeting a minimum quality level to the network. Maintaining this service allows a masternode to enter and move up through the global list and eventually into the selection pool to receive payment.

Proof of Stake - PoS Consensus mechanism that relies on ownership of a cryptocurrency to maintain the blockchain. In Proof of Stake systems, each owner of the currency can use their wallet to “stake,” and there’s a small chance that they will be chosen to create the next block and add it to the chain. In this way consensus is maintained across all nodes. Proof of Stake saves electricity and does not require specialized computer hardware. It does however suffer from several pitfalls, including the “nothing at stake” problem. Since no electricity is consumed, in the event of an attack it is actually beneficial for Proof of Stake nodes to “vote” to accept both the legitimate chain and the attacker’s chain.

Proof of Work - PoW Consensus mechanism that keeps all nodes honest by requiring computational power to be expended in order to create new blocks. Miners must use expensive equipment and burn electricity to add blocks to the blockchain. Without a consensus mechanism of some sort, any node could add blocks to the chain and the network’s nodes would never agree on which chain was valid.

Public Key The *public key* is derived from the private key but is not secret and can be revealed to anybody. When a private key is used to sign messages, the public key is used to verify that the signature is valid.

Pump and dump Inflating the value of a financial asset that has been produced or acquired cheaply, often using aggressive publicity and misleading statements. The publicity causes others to acquire the asset, forcing up its value. When the value is high enough, the perpetrator sells their assets, cashing in and flooding the market, which causes the value to crash. This is particularly common in markets with low liquidity, such as some altcoins.

Quorum Group of masternodes signing or voting on some action, with the formation of the group determined by some determination algorithm.

QR Code A two-dimensional graphical block containing a monochromatic pattern representing a sequence of data. QR codes are designed to be scanned by cameras, including those found in mobile phones, and are frequently used to encode GoByte addresses.

Satoshi Nakamoto *Satoshi Nakamoto* is the name used by the person or people who designed Bitcoin and created its original reference implementation.

SDK Software Development Kit. A set of tools, code and documentation used by developers to create apps targeting a specific hardware or software platform.

Signaling An indication, flag, or signal of support for a feature or fork. The term signaling is most often used in the context of miners delivering this indication of support or agreement. The message is generally delivered through their adoption of updated software in support of a particular protocol and/or by setting a specific version bit within discovered blocks.

State View The current state of all data objects once all changes from state transitions have been applied. Used in Evolution to determine what should be displayed in a given social wallet, for example.

Spork The Dash development team created a mechanism known as a “*spork*” by which updated code is released to the network, but not immediately made active (or “enforced”), mechanism adopted by GoByte. Communication is sent out to users informing them of the change and the need for them to update their clients. Those who update their clients run the new code, but in the event of errors occurring with that new code, the client’s blocks are not rejected by the network and unintended forks are avoided. Data about the error can then be collected and forwarded to the development team. Once the development team is satisfied with the new code’s stability in the mainnet environment – and once acceptable network consensus is attained – enforcement of the updated code can be activated remotely. Should problems arise, the code can be deactivated in the same manner, without the need for a network-wide rollback or client update.

Tainted Coins Taint is a measure of correlation between two (wallet) addresses. It is only important if the user is trying to remain anonymous.

tGBX Test GoByte, used on *testnet*.

Testnet *Testnet* is a network only for testing (parallel to the mainnet), test wallets, test coins, test masternodes, test miners, and test users all simulate their mainnet counterparts in a safe environment where errors or forks are not harmful.

Tor An anonymous routing protocol used by people wanting to hide their identity online.

Transaction Some movement of data on the distributed blockchain ledger. Transactions may be divided into classical and special transactions. Similar to Bitcoin, classical transactions move balances between addresses on the blockchain. Special transactions contain an extra payload in the format defined by [DIP2](#), and can be used to manage blockchain users, for example.

Transaction Block A collection of transactions on the GoByte network, gathered into a block that can then be hashed and added to the blockchain.

Transaction Fee A *small fee* imposed on some transactions sent across the GoByte network. The transaction fee is awarded to the miner that successfully hashes the block containing the relevant transaction.

Unconfirmed Transactions Transactions that are not yet processed by miners or held via InstantSend are “unconfirmed on the blockchain.” Unconfirmed transactions can be reversed and should not be considered as “final.”

Vanity Address A GoByte address with a desirable pattern, such as a name.

Virgin GoByte GoByte received as a reward for mining a block or running a masternode. These have not yet been spent anywhere and are “virgin.”

Volatility The measurement of price movements over time for a traded financial asset (including GoByte).

Wallet A method of storing GoByte for later use. A wallet holds the private keys associated with GoByte addresses. The blockchain is the record of the GoByte balances (and transactions) associated with those addresses.

Whitepaper A *white paper* is an authoritative report or guide that informs readers concisely about a complex issue and presents the issuing body’s philosophy on the matter. It is meant to help readers understand an issue, solve a problem, or make a decision.

NeoScript *NeoScript* is a hashing algorithm created by Feathercoin Core developers.

Zero Confirmations This is a transaction without any confirmations from the blockchain. It is technically reversible (unless InstantSend was used).

vin A transaction (tx) consists of one or more inputs and one or more outputs. The vin is the list of inputs to the transaction, and vout is the list of outputs. Masternodes require a 1000 GBX vin (exactly that amount) in order to work.

VMN Virtual Masternode - a standalone masternode emulator in JavaScript that simulates Layer 1-3 Evolution functions for DAP design, development and testing.

1.6 Wallets

Whenever you are storing objects with a market value, security is necessary. This applies to barter systems as well as economies using currency as a medium of exchange. While banks store balances on a private ledger, cryptocurrencies store balances under unique addresses on a distributed public ledger. The cryptographic private keys to access the balance stored on each public address are therefore the object of value in this system. This section of the documentation discusses different practical methods of keeping these keys safe in wallets, while still remaining useful for day-to-day needs.

For safety, it is not recommended to store significant funds on exchanges or software wallets. If you are holding cryptocurrency worth more than the device you use to store it, you should purchase a *hardware wallet*.

1.6.1 GoByte Core Wallet

GoByte Core Wallet is the full official release of GoByte, and supports all GoByte features as they are released, including InstantSend and PrivateSend, as well as an RPC console and governance features. GoByte Core Wallet (sometimes known as the QT wallet, due to the QT software framework used in development) is a professional or heavy wallet which downloads the full blockchain (several GB in size) and can operate as both a full node or masternode on the network. Because of the requirement to hold a full copy of the blockchain, some time is required for synchronisation when starting the wallet. Once this is done, the correct balances will be displayed and the functions in the wallet can be used. GoByte Core Wallet is available for macOS, Linux, Raspberry Pi and Windows.

Features:

- PrivateSend
- InstantSend
- Wallet encryption
- Coin control and fee control
- QR code generation and address book
- Masternode commands and voting
- Automated backup
- Debug console

Available documentation:

Installation

Installing GoByte Core is as simple as going to <https://www.gobyte.network/> and downloading the appropriate file for your system, then following the appropriate installation steps for your system. Detailed guides are available for Linux, macOS and Windows operating systems below.

It is also possible to *compile GoByte Core from source code*.

Linux Installation Guide

This guide describes how to download, verify, install and encrypt the GoByte Core wallet for Linux. The guide is written for Ubuntu 20.04 LTS, but the steps should be similar for other Linux distributions.

Downloading the GoByte Core wallet

Visit <https://www.gobyte.network/downloads> to download the latest GoByte Core wallet. In most cases, the website will properly detect which version you need. Click the GoByte Core button to download the package directly.



Fig. 3: The website properly detects the wallet appropriate for your system

If detection does not work, you will need to manually choose your operating system and whether you need a 32 or 64 bit version. If you are unsure whether your version of Linux is 32 or 64 bit, you can check in Ubuntu under the **Settings > About > OS Type**. For details on how to check this in other versions of Linux, see [here](#).

If you have a 32-bit system, download **GoByte Core x86**. If you have a 64-bit system, download **GoByte Core x64**. Once you know which version you need, download the GoByte Core TGZ file to your computer from <https://www.gobyte.network/downloads> and save it to your Downloads folder.

Verifying GoByte Core

This step is optional, but recommended to verify the authenticity of the file you downloaded. This is done by checking its detached signature against the public key published by the GoByte Core development team. To download the detached signature, click the **Signature** button on the wallet download page and save it to the same folder as the downloaded binary.

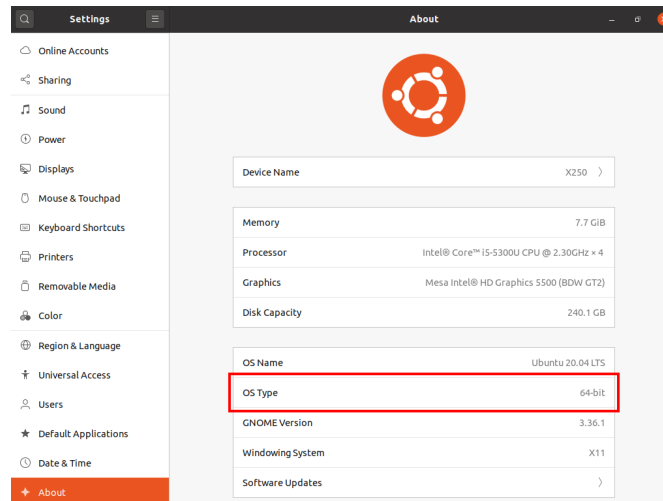


Fig. 4: Ubuntu System Overview. This is a 64 bit system.

All releases of GoByte are signed using GPG by Antonio Moratti (or Hisyam Nasir) with the key 9ED8 A2B0 A016 6C55, [verifiable here on Keybase](#). Open a terminal, import the key and verify the authenticity of your download as follows:

```
curl https://keybase.io/antoniomoratti/pgp_keys.asc | gpg --import
gpg --verify gobytecore-0.13.0.0-x86_64-linux-gnu.tar.gz.asc
```

If you see the message Good signature from "Antonio Moratti <antoniom@gobyte.network>" [unknown] then you have an authentic copy of GoByte Core for Linux.

Extracting GoByte Core

GoByte Core for Linux is distributed as a compressed archive and not an installer. This is because this same archive also contains other files built for running a masternode on a server, for example. In this guide, we will extract the executable file with a graphical user interface (GUI) designed for use by end users as a wallet.

Extract GoByte Core as follows:

```
tar xzf gobytecore-0.13.0.0-x86_64-linux-gnu.tar.gz
```

This will create a folder named gobytecore-0.13.0 in the current working directory. We will now install the executable binaries to /usr/local/bin using the install command:

```
sudo install -m 0755 -o root -g root -t /usr/local/bin gobytecore-0.13.0/bin/*
```

Start GoByte Core from the terminal with the following command:

```
gobyte-qt
```

The first time the program is launched, you will be offered a choice of where you want to store your blockchain and wallet data. Choose a location with enough free space, as the blockchain can reach 30GB+ in size. It is recommended to use the default data folder if possible.

GoByte Core will then start up. This will take a little longer than usual the first time you run it, since GoByte Core needs to generate cryptographic data to secure your wallet.



Fig. 5: Downloading the PGP key and verifying the signed binary



Fig. 6: Choosing the GoByte Core data folder



Fig. 7: Starting GoByte Core

Synchronizing GoByte Core to the GoByte network

Once GoByte Core is successfully installed and started, you will see the wallet overview screen. You will notice that the wallet is “out of sync”, and the status bar at the bottom of the window will show the synchronization progress.



Fig. 8: GoByte Core begins synchronizing with the GoByte network

During this process, GoByte Core will download a full copy of the GoByte blockchain from other nodes to your device. Depending on your internet connection, this may take a long time. If you see the message “No block source available”, check your internet connection. When synchronization is complete, you will see a small blue tick in the lower right corner.

You can now begin to use your wallet to send and receive funds.

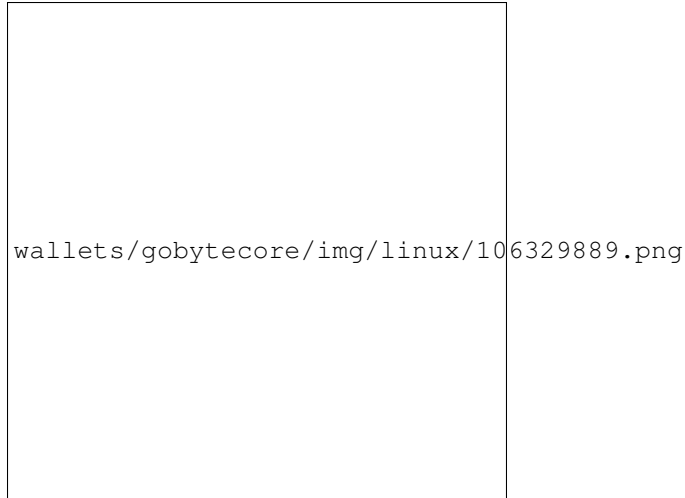


Fig. 9: GoByte Core synchronization is complete

Encrypting your GoByte wallet

After your wallet has synchronized with the GoByte network, it is strongly advised to encrypt the wallet with a password or passphrase to prevent unauthorized access. You should use a strong, new password that you have never used somewhere else. Take note of your password and store it somewhere safe or you will be locked out of your wallet and lose access to your funds.

To encrypt your wallet, click **Settings > Encrypt wallet**.



Fig. 10: Encrypting the GoByte wallet with a password

You will be asked to enter and verify a password.

When the encryption process is complete, you will see a warning that past backups of your wallet will no longer be usable, and be asked to shut down GoByte Core. When you restart GoByte Core, you will see a small blue lock in the lower right corner.



Fig. 11: Entering a password



Fig. 12: Confirm you want to encrypt your wallet



Fig. 13: Fully encrypted and synchronized GoByte Core wallet

Using the Ubuntu Repository to install GoByte Core

Ubuntu allows you to add third-party repositories to install and update software using the apt command line utility. GoByte Core team maintains such a repository, although the software version included here may be older than what is available on the website. To install GoByte Core from the repository, open the Terminal and enter the following commands:

```
sudo add-apt-repository ppa:gobytecoin/gobyte
sudo apt update
sudo apt install gobyted gobyte-qt
```

macOS Installation Guide

This guide describes how to download, install and encrypt the GoByte Core wallet for macOS. The guide is written for macOS Sierra, but the steps should be similar for other versions.

Downloading the GoByte Core wallet

Visit <https://www.gobyte.network/downloads> to download the latest GoByte Core wallet. In most cases, the website will properly detect which version you need. Click **Download Installer** to download the installer directly.

If detection does not work, you will need to manually choose your operating system. Go to <https://www.gobyte.network/downloads> and select the **macOS** tab, then click **Download Installer**. Save the file you downloaded to your Downloads folder.

Verifying GoByte Core

This step is optional, but recommended to verify the authenticity of the file you downloaded. This is done by checking its detached signature against the public key published by the GoByte Core development team. To download the detached signature, click the **Installer Signature** button on the wallet download page and save it to the same folder as the downloaded binary.



Fig. 14: The website properly detects the wallet appropriate for your system

All releases of GoByte are signed using GPG by Antonio Moratti (or Hisyam Nasir) with the key 9ED8 A2B0 A016 6C55, [verifiable here on Keybase](#). Open a terminal, import the key and verify the authenticity of your download as follows:

```
curl https://keybase.io/antoniomoratti/pgp_keys.asc | gpg --import  
gpg --verify gobytecore-0.13.0.0-x86_64-linux-gnu.tar.gz..asc
```

If you see the message Good signature from "Antonio Moratti <antoniom@gobyte.network>" [unknown] then you have an authentic copy of GoByte Core for macOS.

Installing GoByte Core

Open Finder and browse to your Downloads folder. Then double-click on the .dmg file you downloaded to decompress it. A window appears showing the contents of the file.

Drag the GoByte Core application file into your Applications folder to install GoByte Core.

Running GoByte Core for the first time

To run GoByte Core for the first time, either open Launchpad or browse to your **Applications** folder in Finder. Double-click **GoByte Core** or **GoByte-Qt** to start the application. You may see a warning about opening an app from an unidentified developer. To resolve this problem, simply Control-click the app icon and choose **Open** from the shortcut menu, then click **Open** again in the dialog box. The app is saved as an exception to your security settings, and you can open it in the future by double-clicking it just as you can any registered app.



Fig. 15: Downloading the PGP key and verifying the signed binary



Fig. 16: Opening the GoByte Core .dmg file

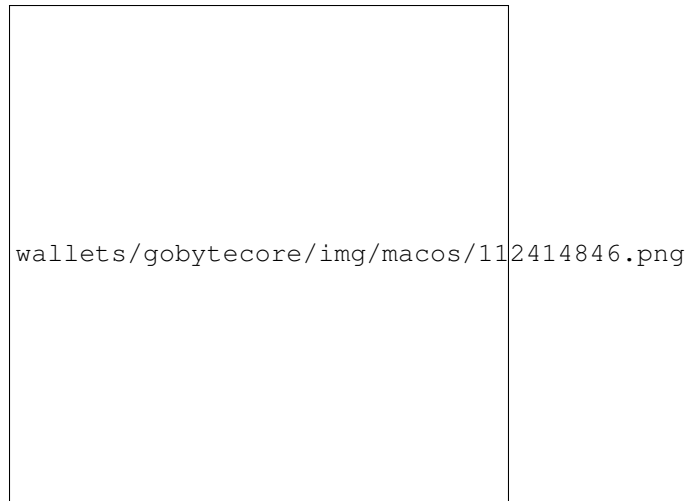


Fig. 17: Installing GoByte Core



The first time the program is launched, you will be offered a choice of where you want to store your blockchain and wallet data. Choose a location with enough free space, as the blockchain can reach 30GB+ in size. It is recommended to use the default data folder if possible.

GoByte Core will then start up. This will take a little longer than usual the first time you run it, since GoByte Core needs to generate cryptographic data to secure your wallet.

Synchronizing GoByte Core to the GoByte network

Once GoByte Core is successfully installed and started, you will see the wallet overview screen. You will notice that the wallet is “out of sync”, and the status bar at the bottom of the window will show the synchronization progress.

During this process, GoByte Core will download a full copy of the GoByte blockchain from other nodes to your device. Depending on your internet connection, this may take a long time. If you see the message “No block source available”, check your internet connection. When synchronization is complete, you will see a small blue tick in the lower right corner.



Fig. 18: Unblocking macOS from running GoByte Core

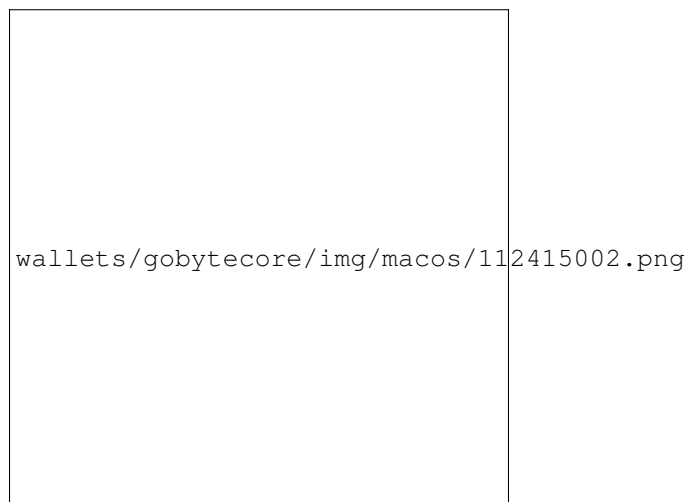


Fig. 19: Choosing the GoByte Core data folder

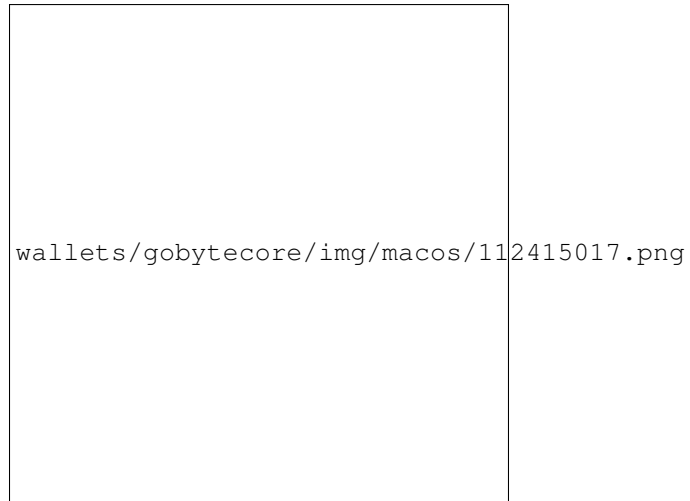


Fig. 20: Starting GoByte Core



Fig. 21: GoByte Core begins synchronizing with the GoByte network



Fig. 22: GoByte Core synchronization is complete

You can now begin to use your wallet to send and receive funds.

Encrypting your GoByte wallet

After your wallet has synchronized with the GoByte network, it is strongly advised to encrypt the wallet with a password or passphrase to prevent unauthorized access. You should use a strong, new password that you have never used somewhere else. Take note of your password and store it somewhere safe or you will be locked out of your wallet and lose access to your funds.

To encrypt your wallet, click **Settings > Encrypt Wallet**.

You will be asked to enter and verify a password.

When the encryption process is complete, you will see a warning that past backups of your wallet will no longer be usable, and be asked to shut down GoByte Core. When you restart GoByte Core, you will see a small blue lock in the lower right corner.

You can now begin to use your wallet to safely send and receive funds.

Windows Installation Guide

This guide describes how to download, install and encrypt the GoByte Core wallet for Windows. The guide is written for Windows 10, but the steps should be similar for Windows XP, Vista, 7 and 8.



Fig. 23: Encrypting the GoByte wallet with a password



Fig. 24: Enter a password



Fig. 25: Confirm you want to encrypt your wallet



Fig. 26: Fully encrypted and synchronized GoByte Core wallet

Downloading the GoByte Core wallet

Visit <https://www.gobyte.network/downloads> to download the latest GoByte Core wallet. In most cases, the website will properly detect which version you need. Click **Download Installer** to download the installer directly.



Fig. 27: The website properly detects the wallet appropriate for your system

If detection does not work, you will need to manually choose your operating system and whether you need a 32 or 64 bit version. If you are unsure whether your version of Windows is 32 or 64 bit, you can check in Windows 10 under **Start > Settings > System > About**. For details on how to check this in other versions of Windows, see [here](#).

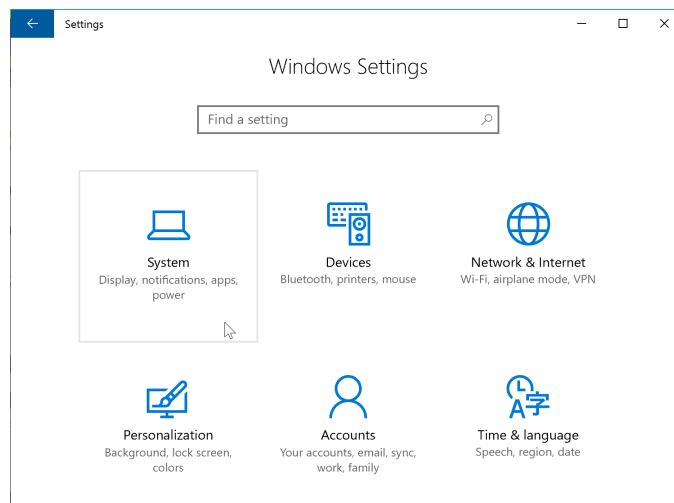


Fig. 28: In Windows Settings, click System

Once you know which version you need, download the GoByte Core Installer to your computer from <https://www.gobyte.network/downloads> and save the file you downloaded to your Downloads folder.

Verifying GoByte Core

This step is optional, but recommended to verify the authenticity of the file you downloaded. This is done by checking its detached signature against the public key published by the GoByte Core development team. To download the

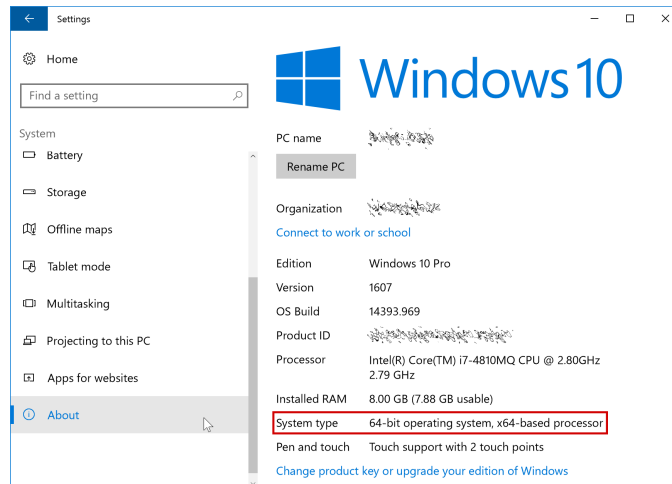


Fig. 29: Under the System section, click About to view the System type. This is a 64 bit system.

detached signature, click the **Installer Signature** button on the wallet download page and save it to the same folder as the downloaded binary.

All releases of GoByte are signed using GPG by Antonio Moratti (or Hisyam Nasir) with the key 9ED8 A2B0 A016 6C55, [verifiable here on Keybase](#). Install [Gpg4win](#) if it is not already available on your computer. Once it is installed, open the **Kleopatra** certificate manager and click **Lookup on Server**. Type Antonio Moratti in the **Find** field and click **Search**. Verify the Key-ID matches the ID above, then click **Import**.

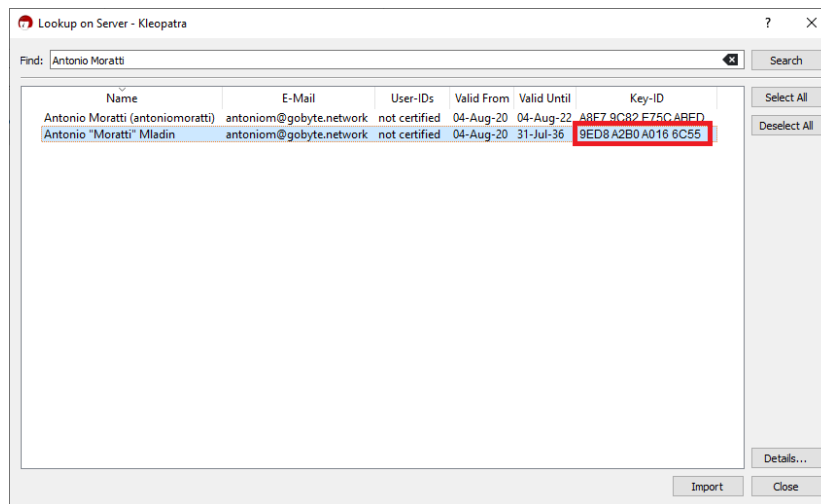


Fig. 30: Importing antoniomoratti's GPG public key

Skip any requests to certify the certificate with your own key. Next, click **Decrypt/Verify...** and select the detached signature file named `gobytecore-qt.exe.asc` in the same folder as the downloaded installer.

If you see the first line of the message reads `Verified gobytecore-qt.exe with gobytecore-qt.exe.asc` then you have an authentic copy of GoByte Core for Windows.

Running the GoByte Core installer

Double-click the file to start installing GoByte Core.

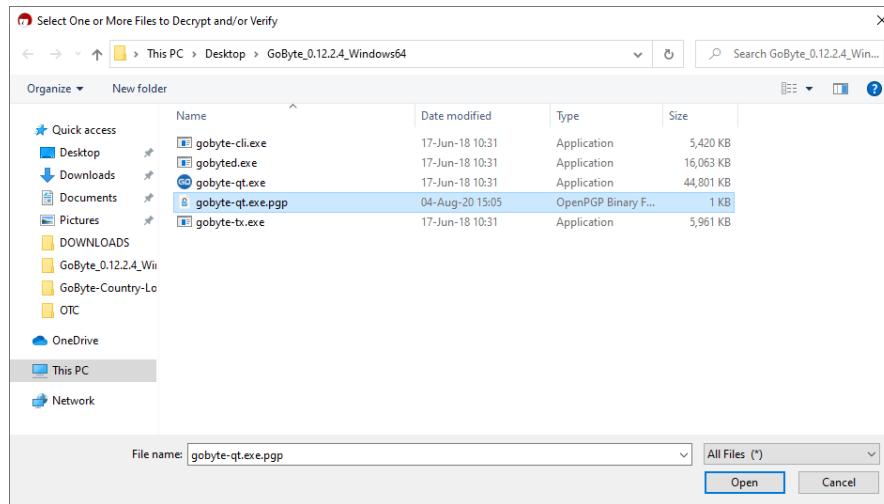


Fig. 31: Selecting the signature file for verification

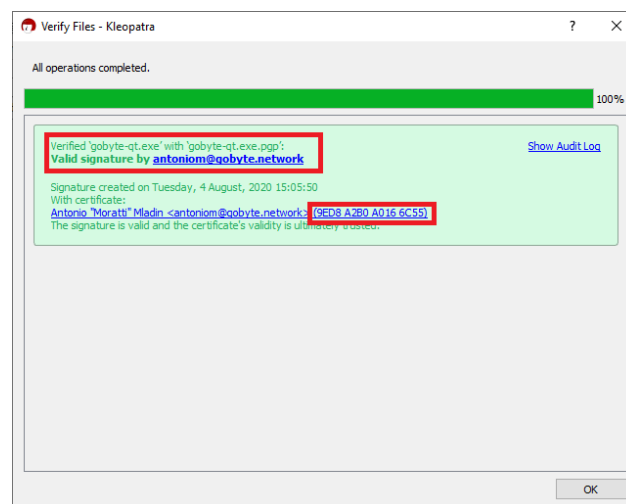
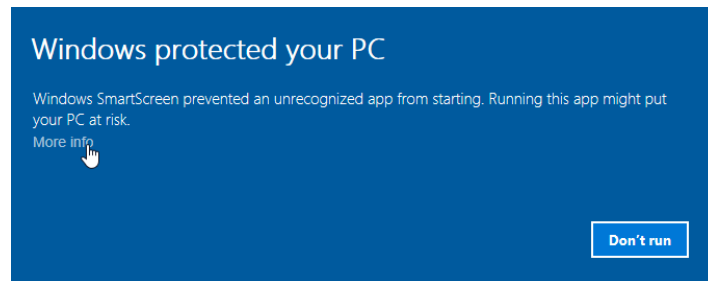


Fig. 32: The binary installer has been verified



Fig. 33: The GoByte Core installer in the Downloads folder

At this point, you may see a warning from Windows SmartScreen that the app is unrecognized. You can safely skip past this warning by clicking **More info**, then **Run anyway**.



The installer will then guide you through the installation process.

Click through the following screens. All settings can be left at their default values unless you have a specific reason to change something.

Running GoByte Core for the first time

Once installation is complete, GoByte Core will start up immediately. If it does not, click **Start > GoByte Core > GoByte Core** to start the application. The first time the program is launched, you will be offered a choice of where you want to store your blockchain and wallet data. Choose a location with enough free space, as the blockchain can reach 30GB+ in size. It is recommended to use the default data folder if possible.

GoByte Core will then start up. This will take a little longer than usual the first time you run it, since GoByte Core needs to generate cryptographic data to secure your wallet.

Synchronizing GoByte Core to the GoByte network

Once GoByte Core is successfully installed and started, you will see the wallet overview screen. You will notice that the wallet is “out of sync”, and the status bar at the bottom of the window will show the synchronization progress.

During this process, GoByte Core will download a full copy of the GoByte blockchain from other nodes to your device. Depending on your internet connection, this may take a long time. If you see the message “No block source



Fig. 34: Bypassing Windows SmartScreen to run the app. This warning is known as a “false positive”.



Fig. 35: The GoByte Core installer welcome screen



Fig. 36: Select the installation location



Fig. 37: Select the Start menu folder



Fig. 38: GoByte Core is being installed



Fig. 39: Installation is complete

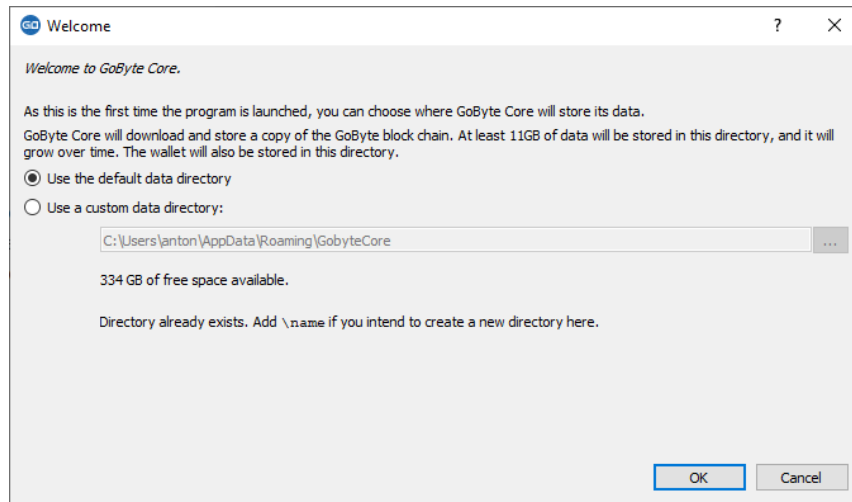


Fig. 40: Choosing the GoByte Core data folder



GoByte Core
Version v0.12.3.4-b4e3f66
© 2017-2019 The GoByte Core developers
© 2009-2019 The Bitcoin Core developers

Fig. 41: Starting GoByte Core

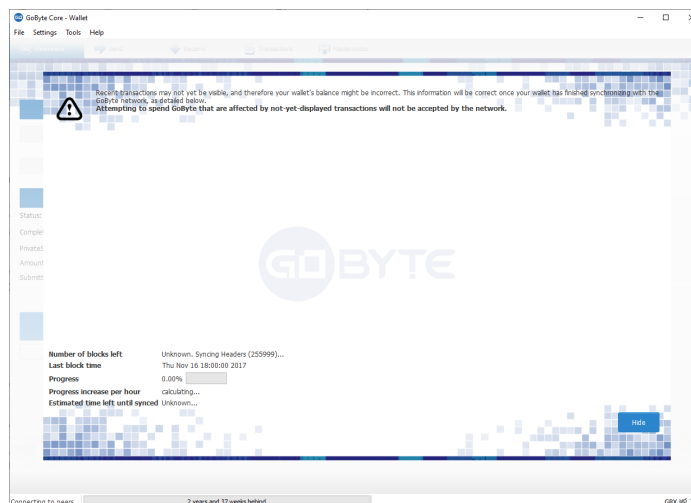


Fig. 42: GoByte Core begins synchronizing with the GoByte network

available”, check your internet connection. When synchronization is complete, you will see a small blue tick in the lower right corner.

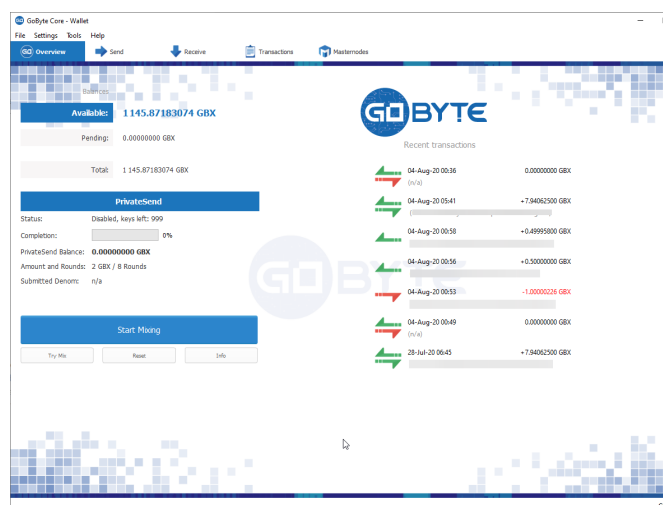


Fig. 43: GoByte Core synchronization is complete

You can now begin to use your wallet to send and receive funds.

Encrypting your GoByte wallet

After your wallet has synchronized with the GoByte network, it is strongly advised to encrypt the wallet with a password or passphrase to prevent unauthorized access. You should use a strong, new password that you have never used somewhere else. Take note of your password and store it somewhere safe or you will be locked out of your wallet and lose access to your funds.

To encrypt your wallet, click **Settings > Encrypt Wallet**.

You will be asked to enter and verify a password.

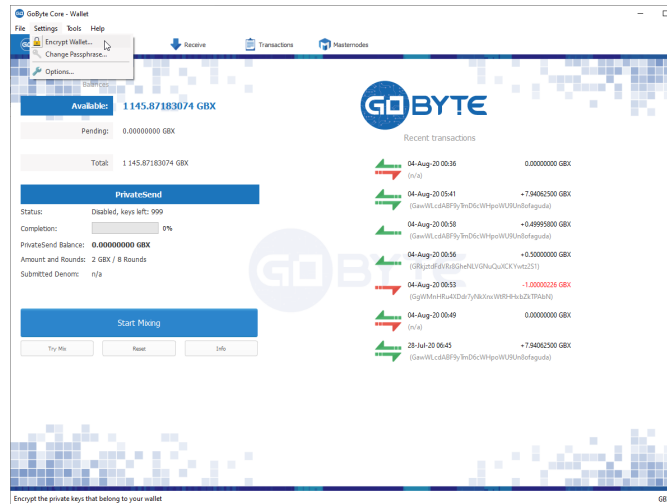


Fig. 44: Encrypting the GoByte wallet with a password

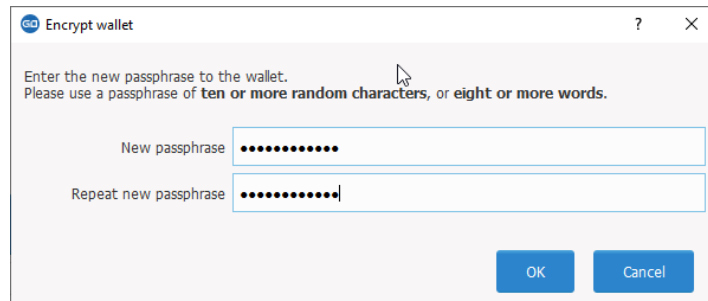


Fig. 45: Enter a password

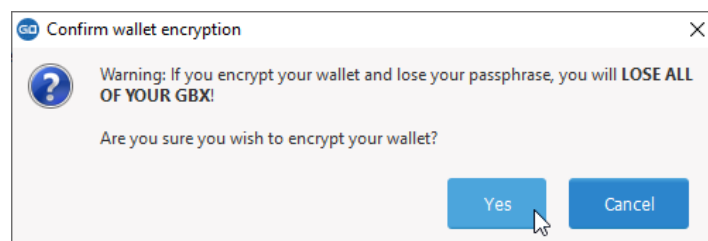


Fig. 46: Confirm you want to encrypt your wallet

When the encryption process is complete, you will see a warning that past backups of your wallet will no longer be usable, and be asked to shut down GoByte Core. When you restart GoByte Core, you will see a small blue lock in the lower right corner.

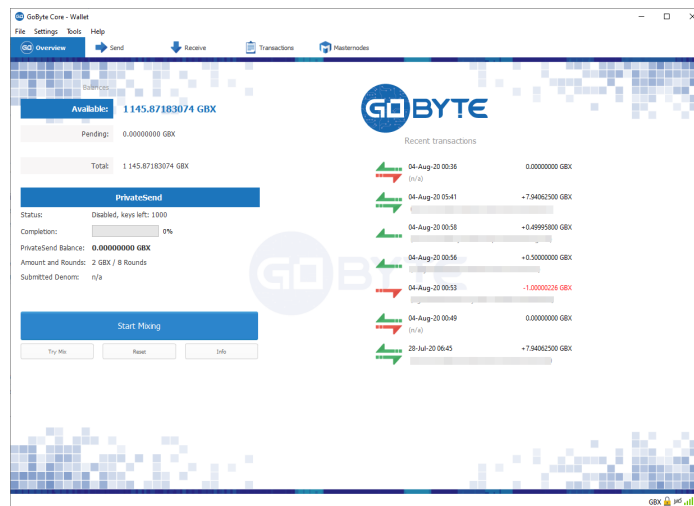


Fig. 47: Fully encrypted and synchronized GoByte Core wallet

You can now begin to use your wallet to safely send and receive funds.

Interface

The GoByte Core Wallet is an application that runs on your computer and allows you to make transactions on the GoByte network. Most transactions are for sending or receiving GoByte, but it is also possible to create signed messages or control a masternode, for example. The GoByte Core Wallet interface is described in detail in the following sections.

The Main Window

The GoByte Core window is broken up into several areas:

- The menu bar
- The tab bar
- The main area
- The status bar

The Menu Bar

The menu bar provides access to all functions of GoByte Core. There are four menus available:

File The File menu is used to manage your wallet, messages and addresses.

Settings The Settings menu provides access to wallet encryption options and general software settings.

Tools The Tools menu provides information on the network, allows you modify masternode configuration files and other advanced functions.

Help The Help menu links to documentation, guides and legal statements relating to GoByte Core.

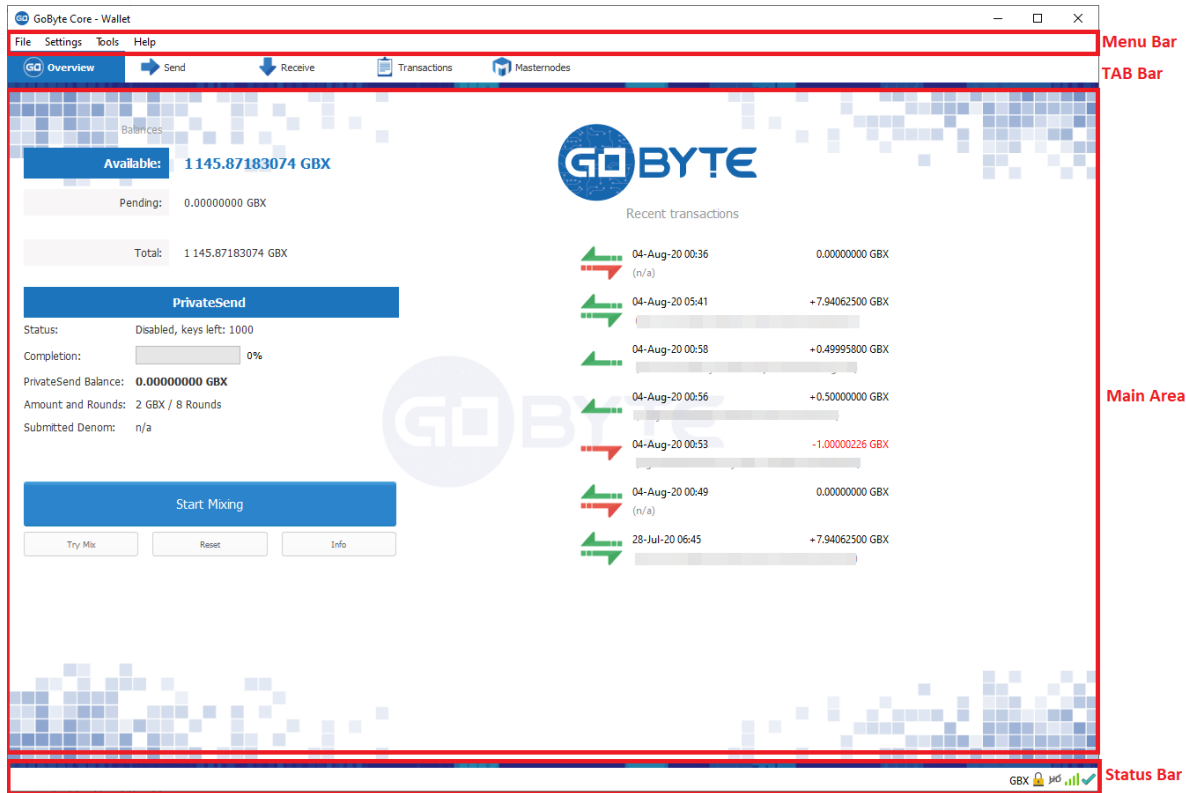


Fig. 48: The GoByte Core Wallet

The Tab Bar

The tab bar is used to quickly switch between the main areas of the GoByte Core. The content in the main area of GoByte Core changes depending on which tab you have selected. The following tabs are available:

The Overview tab

The overview tab offers quick access to your balance and most recent transactions, as well as the PrivateSend feature and options for coin mixing.

The left part of the main area is divided into two areas. The upper area shows your balances:

Available This shows your current liquid balance. This is the amount of GoByte you can spend now.

Pending This shows funds waiting for a transaction to complete.

Immature This shows funds from masternode or mining payments which have not yet reached the required number of confirmations.

Total This is simply your available and pending funds added together.

The lower area shows the status of PrivateSend and allows you to mix your funds using the GoByte Masternode Network.

The right part of the screen shows your recent transactions. These are identified by icons as follows:

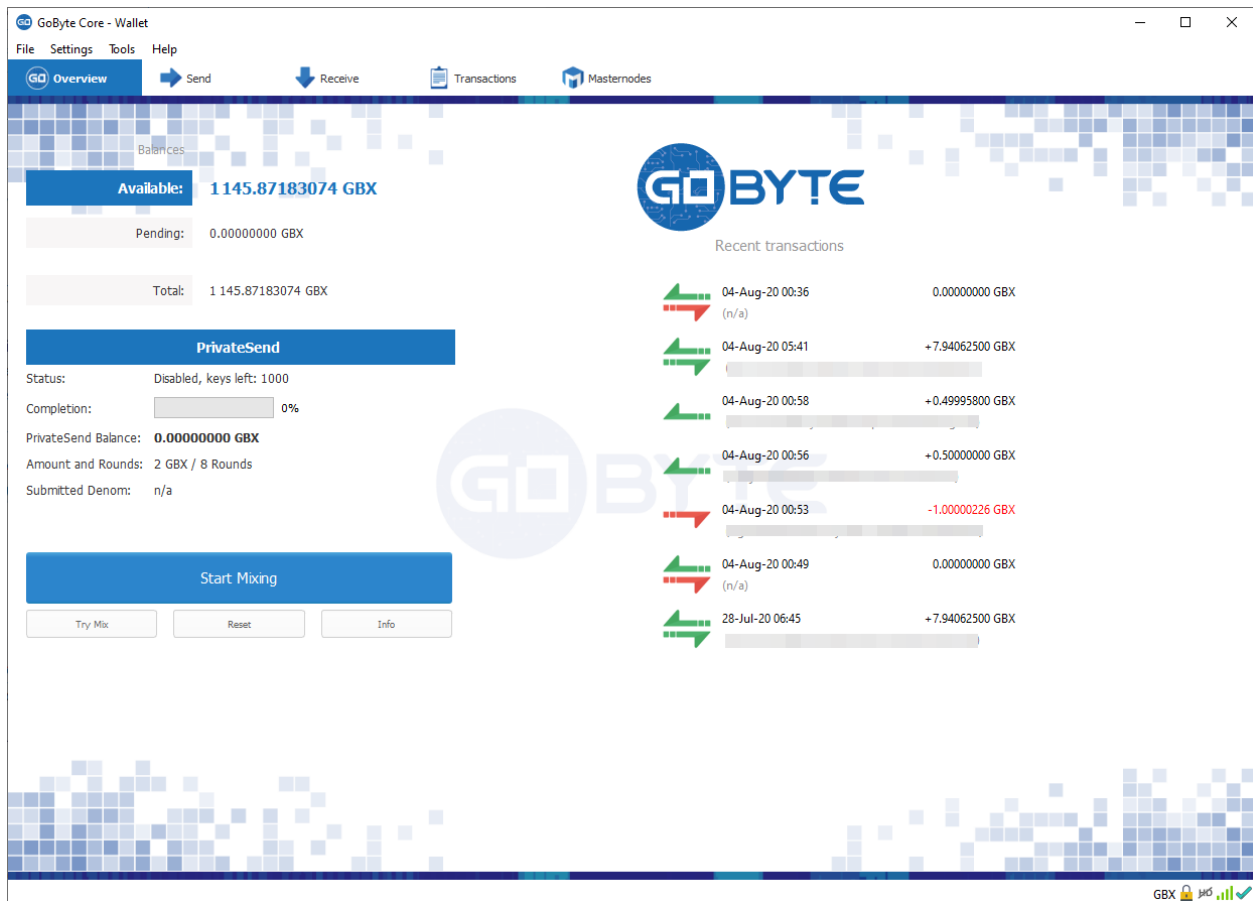
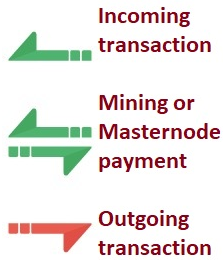


Fig. 49: The GoByte Core Overview tab



- Recent incoming transactions appear black, prefixed by a + sign
- Recent outgoing transactions appear red, prefixed by a – sign
- Incoming mining or masternode payments also appear black

For more details on your transaction history, see the Transactions tab.

The Send Tab

The Send tab allows you to send funds to another address on the GoByte network. It contains fields to enter the recipient's address, a label for the address, and the amount of GoByte you wish to send. Options related to the transaction fee, InstantSend and PrivateSend are also available. A quick view of your total balance is also available in the lower right corner.

The Receive Tab

The Receive tab allows you to create addresses to receive GoByte. You can create a request for a specific amount of GoByte or include a specific message, and send it to another user as a link or QR code.

The Transactions Tab

The transactions tab shows the entire transaction history for all addresses associated with your wallet. This appears as a table showing the time, type, label and amount of GoByte for each transaction. You can also export the transaction history as a CSV file by clicking the Export button in the bottom right corner of the window.

The icons in the leftmost column indicate the status of the transaction. A tick indicates that the recommended number of confirmations has been passed, while a clock indicates that the transaction has yet to reach six confirmations.

The Status Bar

The status bar shows a synchronization progress bar and a row of status icons which indicate the status of your connection to the GoByte network.

The Synchronization Bar

This bar shows the synchronization status of GoByte Core with the GoByte network. Each time you open GoByte Core, it will begin downloading the blocks which have been created on the blockchain in the time since you last opened the app. These blocks are downloaded from other GoByte users and masternodes. If you have never opened the app before, this could mean several years' worth of blocks need downloading. The following statuses are possible:

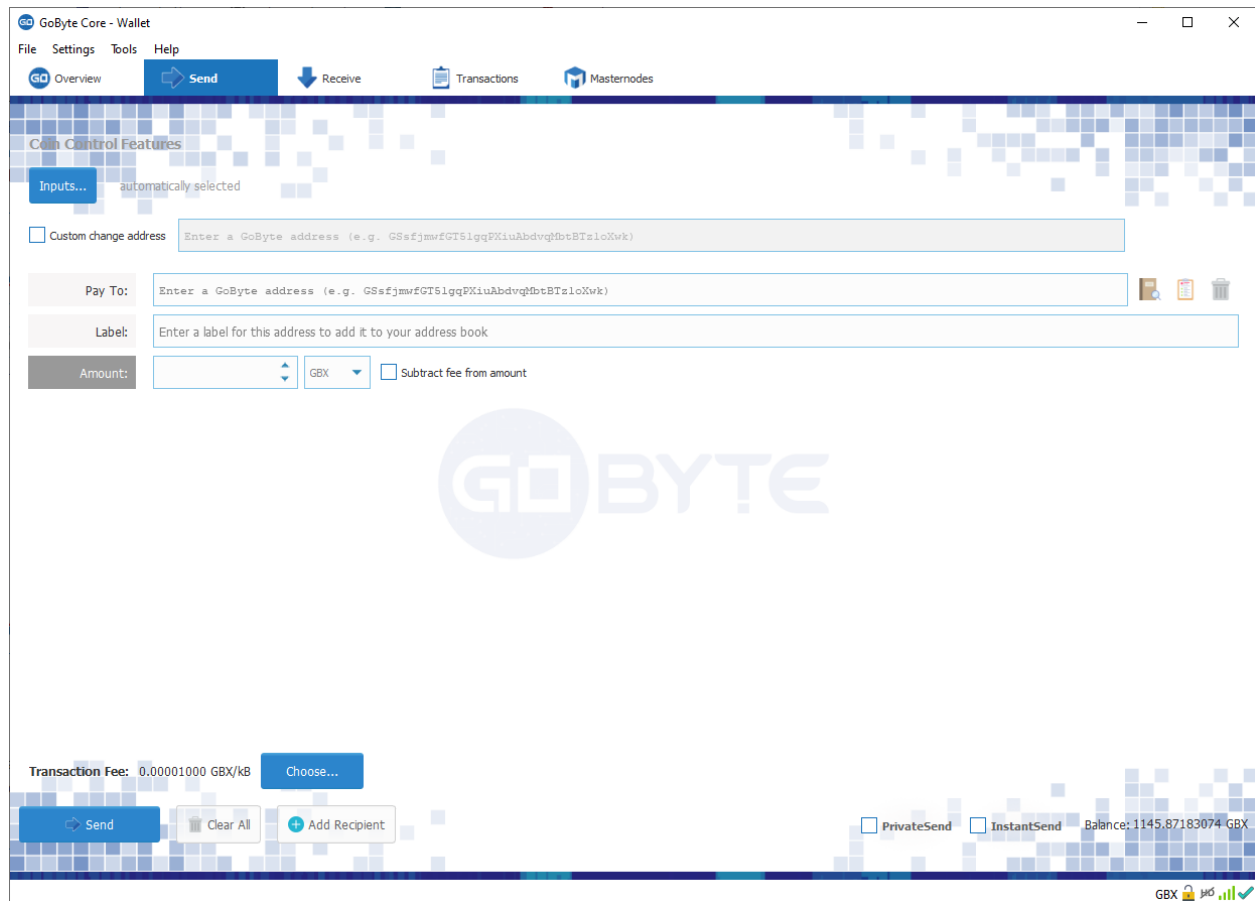


Fig. 50: The Send tab

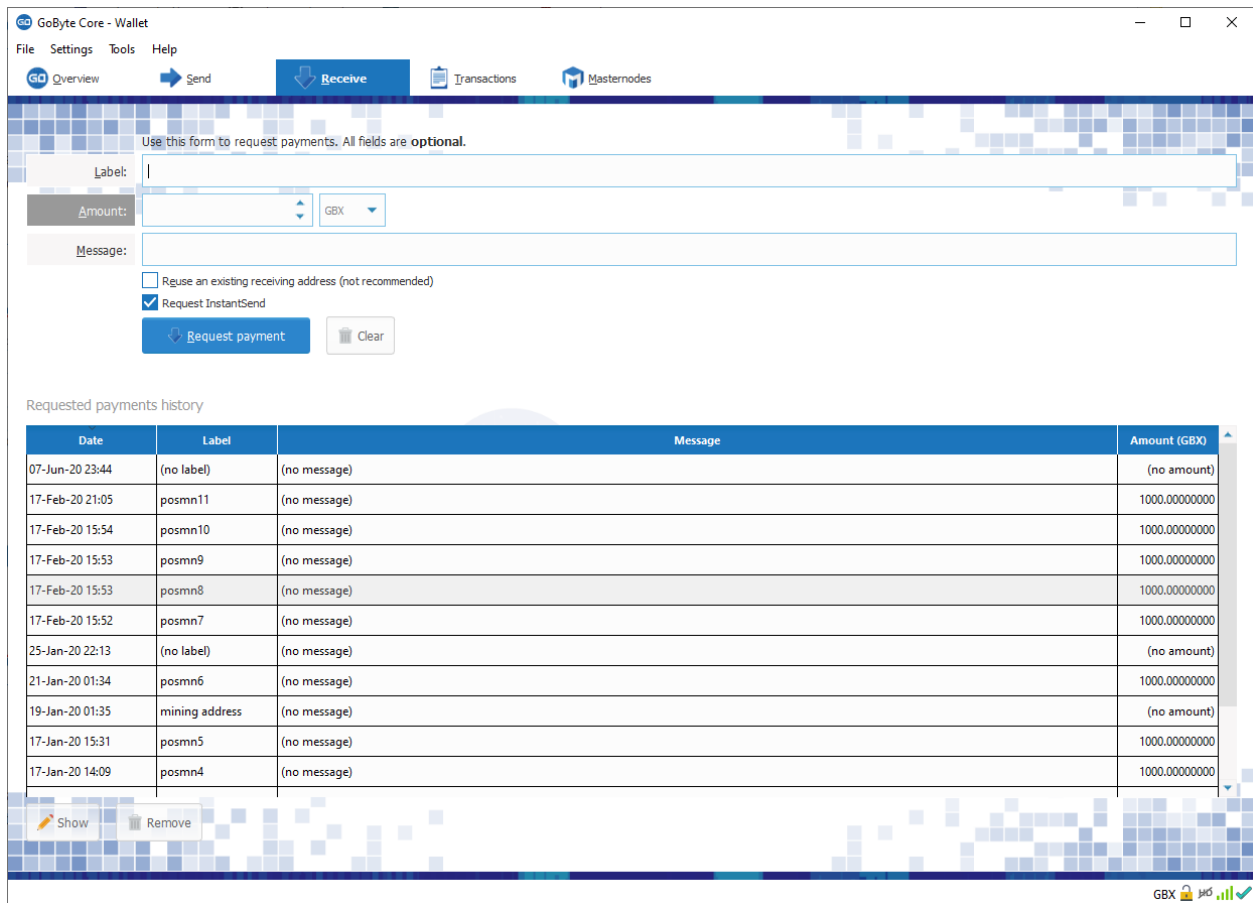


Fig. 51: The Receive tab

GoByte Core - Wallet

File Settings Tools Help

Overview Send Receive Transactions Masternodes

All Most Common Enter address or label to search Min amount

Date	Type	Address / Label	Amount (GBX)
04-Aug-20 00:36	Payment to yourself	(n/a)	0.00000000
04-Aug-20 05:41	Mined		7.94062500
04-Aug-20 00:58	Received with		0.49995800
04-Aug-20 00:56	Received with		0.50000000
04-Aug-20 00:53	Sent to		-1.00000226
04-Aug-20 00:49	Payment to yourself		0.00000000
28-Jul-20 06:45	Mined		7.94062500
27-Jul-20 22:38	Mined		7.94062500
24-Jul-20 13:39	Sent to		-5.00000000
21-Jul-20 04:51	Mined		7.94062500
15-Jul-20 14:56	Mined		7.94062500
15-Jul-20 06:23	Mined		7.94062500
13-Jul-20 14:53	Mined		7.94062500
13-Jul-20 12:14	Mined		7.94062500
13-Jul-20 03:36	Mined		7.94062500
07-Jul-20 18:23	Mined		7.94062500
07-Jul-20 13:17	Mined		7.94062500
07-Jul-20 08:45	Mined		7.94062500
01-Jul-20 23:08	Mined		7.94062500
26-Jun-20 16:08	Mined		7.94062500
26-Jun-20 13:36	Mined		7.94062500

Selected amount: Export

GBX

Fig. 52: The transactions tab

No block source available This occurs if your internet connection is down, or if the ports required by GoByte Core are blocked by a firewall.

Synchronizing with network GoByte Core is downloading blocks from the network.

Synchronizing masternodes/masternode payments/governance objects GoByte Core is synchronizing other data with the second layer network.

Once synchronization is complete, the progress bar will disappear and a tick will appear on the right of the status bar.

The Status Icons



The lock icons indicate the status of your wallet: either locked or unlocked. You need to unlock your wallet to send funds or perform certain other actions.



These icons indicate the quality of your connection to the GoByte network. If you cannot connect because of network problems, you will see the icon on the left. More bars indicate more connections to your peers on the network.



These icons show the synchronization status of GoByte Core with the network. Once synchronization is complete, the refresh icon will become a blue tick.



These icons indicate whether your wallet is running in hierarchical deterministic (HD) mode or standard mode.

The Options Dialog

This documentation describes the functionality of the GoByte Core Options dialog, available under the **Settings > Options** menu in GoByte Core.

Main tab

The Main tab of the Options dialog contains settings related to startup and performance of the GoByte Core app.

Start GoByte Core on system login This option causes GoByte Core to start automatically when the current user logs in. On most computers there is only one main user account which is logged in automatically when the computer turns on, so this option is effectively the same as starting GoByte Core together with the operating system.

Size of database cache This option specifies the size of the database cache in memory. A higher value will result in increased performance when adding new blocks at the cost of higher memory usage. The default value is 100MB and it should not be set lower than this level.

Number of script verification threads This option sets the number of script verification threads, ranging from -4 to 16. [Script verification](#) is the process of following instructions recorded in the blockchain to ensure the transactions are valid. 0 means automatic and will allow script verification to scale to the number of cores available on your processor. Setting a positive number specifies that GoByte Core should use that number of processor cores, while setting a negative number will leave that number of processor cores free.

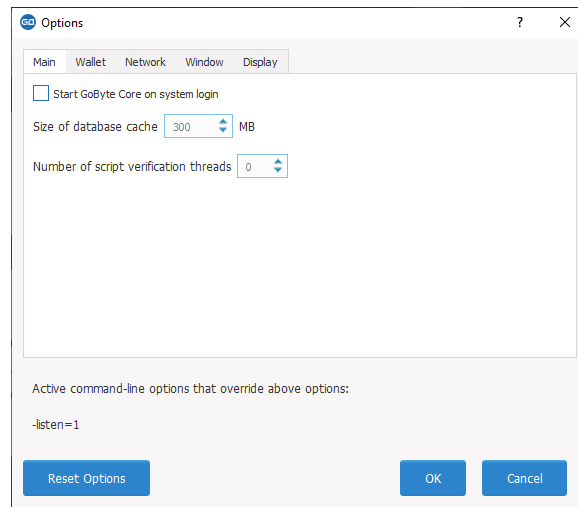


Fig. 53: The Main tab of the GoByte Core Options dialog

Wallet tab

The Wallet tab of the Options dialog contains settings related to how addresses are managed in the GoByte Core app. The first time you run GoByte Core, it will generate a new wallet containing 1000 unique GoByte addresses. This tab allows you to configure how these addresses are used as inputs with the Coin Control, PrivateSend and Masternode features.

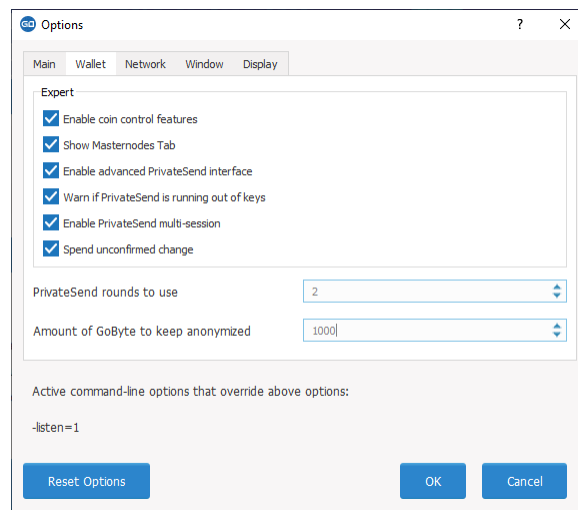


Fig. 54: The Wallet tab of the GoByte Core Options dialog

Enable coin control features Your GoByte Core wallet balance is actually the sum total of all addresses holding balance that are associated with your wallet. When you spend GoByte, GoByte Core will withdraw from as many inputs as necessary to make up the desired amount of GoByte to be transferred. This behavior may be undesirable if you want to keep a certain balance on one address. The most common use case is the requirement to maintain 1000 GoByte on a single address as collateral for a masternode. Enabling this option will add a button labelled **Inputs** on the **Send** tab. This provides access to the **Coin selection** dialog, which can be used to lock, unlock and prioritize different addresses in your wallet. See [here](#) for a more detailed explanation of Coin Control.

Show Masternodes tab Enabling this option causes GoByte Core to display an additional Masternodes tab to the right of the Transactions tab. This option requires you to restart the GoByte Core app. The Masternodes tab can be used to manage interactions (start, stop, check status, etc.) with masternodes controlled by this wallet. This tab is an advanced feature not required by users that do not operate a masternode on the GoByte network.

Enable advanced PrivateSend interface Enabling this option changes the PrivateSend mixing interface on the Overview tab of the GoByte Core wallet to include more options, such as Try Mix and percentage completion. See [here](#) for a full explanation of how to use PrivateSend.

Warn if PrivateSend is running out of keys Enabling this option will cause GoByte Core to display a warning when your original set of 1000 addresses is running out, which may affect PrivateSend mixing. Every time a mixing event happens, up to 9 of your addresses are used up. This means those 1000 addresses last for about 100 mixing events. When 900 of them are used, your wallet must create more addresses. It can only do this, however, if you have automatic backups enabled. Consequently, users who have backups disabled will also have PrivateSend disabled.

Enable PrivateSend multi-session Normally PrivateSend mixing is completed in several consecutive rounds, each using a single masternode. Enabling this option allows multi-session, which means you can use multiple masternode servers at the same time, greatly increasing the speed of the mixing process at the cost of creating more addresses and thus requiring more frequent wallet backups. This feature is experimental as of GoByte Core 12.1.5.

Spend unconfirmed change When this option is enabled, the GoByte Core wallet permits you to immediately spend change from previous transactions that has been transferred internally between addresses associated with the same wallet. This is possible even if the transaction has not yet been confirmed because the wallet knows it will eventually be confirmed since it created the internal transaction itself. Leaving this option enabled allows you to create new transactions even if previous transactions have not yet been confirmed.

PrivateSend rounds to use Use this option to control the number of rounds of PrivateSend mixing to be carried out for your chosen balance. Each round of mixing uses a new masternode. The higher the number of rounds, the more difficult it becomes to trace the GoByte to its original address. This is at the expense of more time required for mixing and potentially higher fees. See [here](#) for a full explanation of how to use PrivateSend.

Amount of GoByte to keep mixed This option allows you to specify how much GoByte should be kept on balance in a ready-to-use mixed state, meaning it has already passed through the PrivateSend mixing process. If you do not have sufficient GoByte available in your balance of unlocked inputs, the amount will be automatically reduced to the available balance and shown in red in the PrivateSend interface on the Overview tab.

Network tab

This tab includes options related to how your connection to the GoByte network is made.

Map port using UPnP This option causes GoByte Core to automatically attempt to open and map the client port on your router using [UPnP](#) (Universal Plug and Play). This feature is supported by most modern home routers and will allow you to connect to the GoByte network without making any special settings on your router.

Allow incoming connections This option causes your client to accept external connections. Since GoByte is a peer-to-peer network and GoByte Core is considered a full client because it stores a copy of the blockchain on your device, enabling this option helps other clients synchronize the blockchain and network through your node.

Connect through SOCKS5 proxy (default proxy) These options allow users on an intranet requiring a proxy to reach the broader internet to specify the address of their proxy server to relay requests to the internet. Contact your system administrator or check out the network settings in your web browser if you are unable to connect and suspect a proxy may be the source of the problem.

Use separate SOCKS5 proxy to reach peers via Tor hidden services These options allow you to specify an additional proxy server designed to help you connect to peers on the Tor network. This is an advanced option for increased privacy and requires a Tor proxy on your network. For more information about Tor, see [here](#).

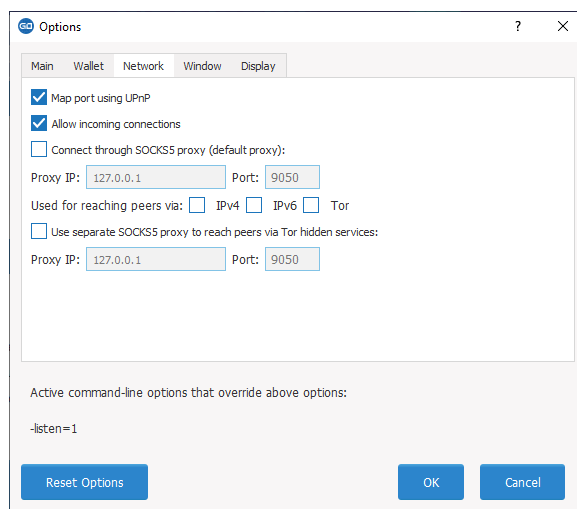


Fig. 55: The Network tab of the GoByte Core Options dialog

Window tab

This option contains options governing behavior of the GoByte Core app window under Microsoft Windows.

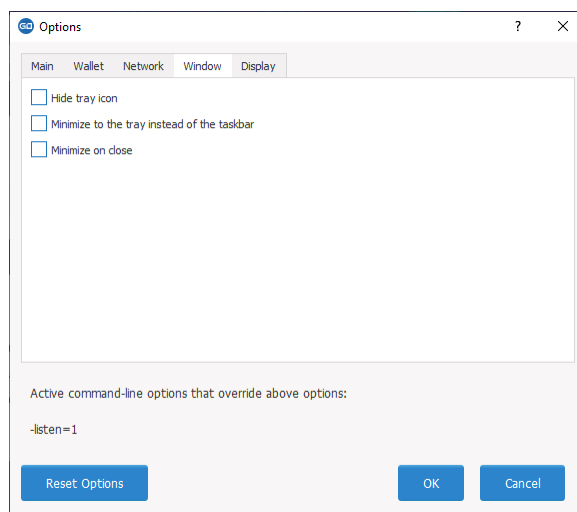


Fig. 56: The Window tab of the GoByte Core Options dialog

Hide tray icon When this option is enabled, GoByte Core will not display an icon in the system tray. This option cannot be selected at the same time as **Minimize to the tray instead of the taskbar**.

Minimize to the tray instead of the taskbar When this option is enabled and the GoByte Core window is minimized, it will no longer appear in your taskbar as a running task. Instead, GoByte Core will keep running in the background and can be re-opened from the GoByte icon in the system tray (the area next to your system clock). This option cannot be selected at the same time as **Hide tray icon**.

Minimize on close When this option is enabled, clicking the X button in the top right corner of the window will cause GoByte Core to minimize rather than close. To completely close the app, select **File > Exit**.

Display tab

This tab contains options relating to the appearance of the GoByte Core app window.

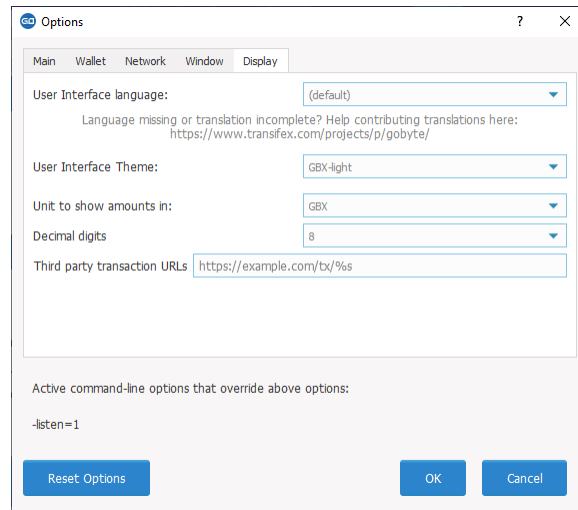


Fig. 57: The Display tab of the GoByte Core Options dialog

User interface language Select your preferred language from this drop-down menu. Changing the language requires you to restart the GoByte Core app.

User interface theme You can use this option to select a different theme governing the appearance of the GoByte Core window. All functionality is identical under the different themes, although the default GoByte-light theme is most recent and most likely to work without any display artifacts.

Unit to show amounts in This allows you to change the default unit of currency in GoByte Core from GBX to mGBX, μ GBX or gbits/duffs. Each unit shifts the decimal separator three places to the right. gBits/duffs are the smallest unit into which GoByte may be separated.

Decimal digits This option allows you to select how many decimal digits will be displayed in the user interface. This does not affect internal accounting of your inputs and balance.

Third party transaction URLs This option allows you to specify an external website to inspect a particular address or transaction on the blockchain. Several blockchain explorers are available for this. To use this feature, enter the URL of your favorite blockchain explorer, replacing the %s with the transaction ID. You will then be able to access this blockchain explorer directly from GoByte Core using the context menu of any given transaction.

The Tools Dialog

This documentation describes the functionality of the GoByte Core Tools dialog, available under the **Tools** menu in GoByte Core.

Information tab

General This section displays information on the name and version of the client and database, and the location of the current application data directory.

Network This section displays information and statistics on the network to which you are connected.

Block chain This section shows the current status of the blockchain.

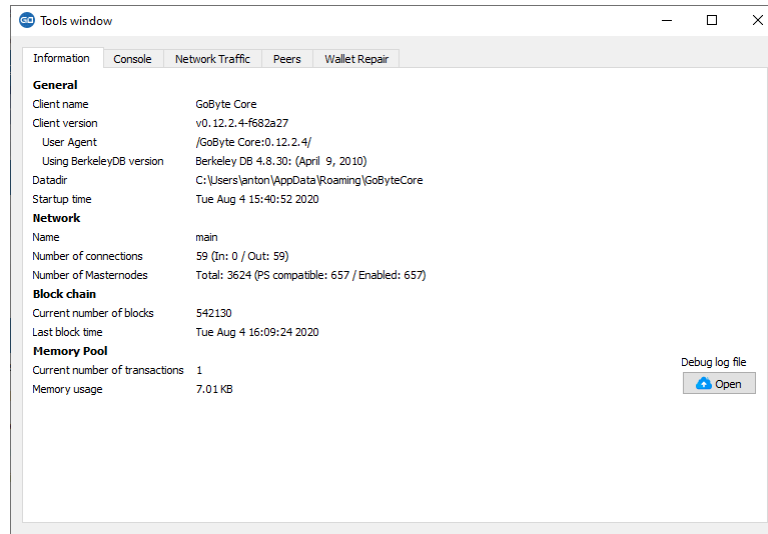


Fig. 58: The Information tab of the GoByte Core Tools dialog

Memory pool This section shows the status of the memory pool, which contains transactions that could not yet be written to a block. This includes both transactions created since the last block and transactions which could not be entered in the last block because it was full.

Open debug log file This button opens debug.log from the application data directory. This file contains output from GoByte Core which may help to diagnose errors.

Console tab

The Console tab provides an interface with the GoByte Core RPC (remote procedure call) console. This is equivalent to the `gobyte-cli` command on headless versions of GoByte, such as `gobyted` running on a masternode. Click the red `—` icon to clear the console, and see the detailed documentation on RPC commands to learn about the possible commands you can issue.

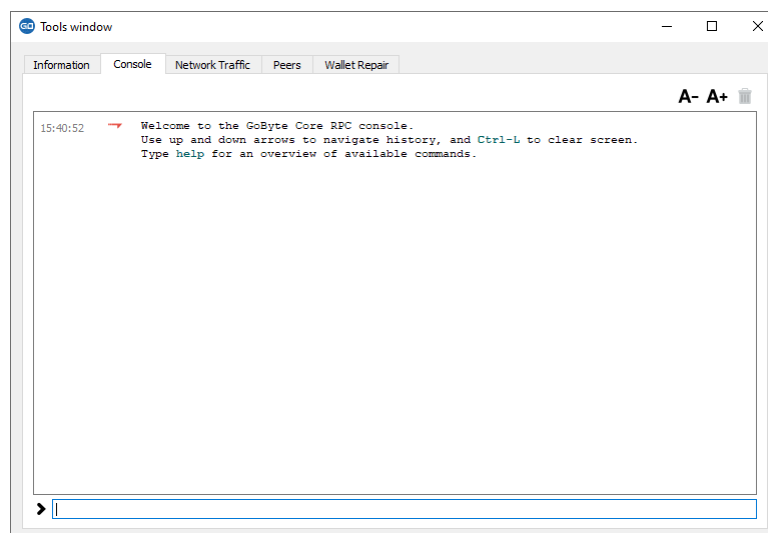


Fig. 59: The Console tab of the GoByte Core Tools dialog

Network Traffic tab

The Network Traffic tab shows a graph of traffic sent and received to peers on the network over time. You can adjust the time period using the slider or **Clear** the graph.

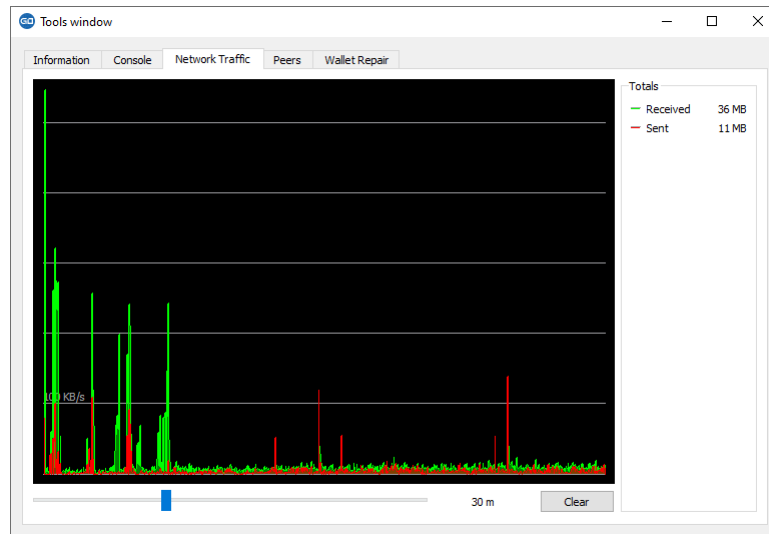


Fig. 60: The Network Traffic tab of the GoByte Core Tools dialog

Peers tab

The Peers tab shows a list of other full nodes connected to your GoByte Core client. The IP address, version and ping time are visible. Selecting a peer shows additional information on the data exchanged with that peer.

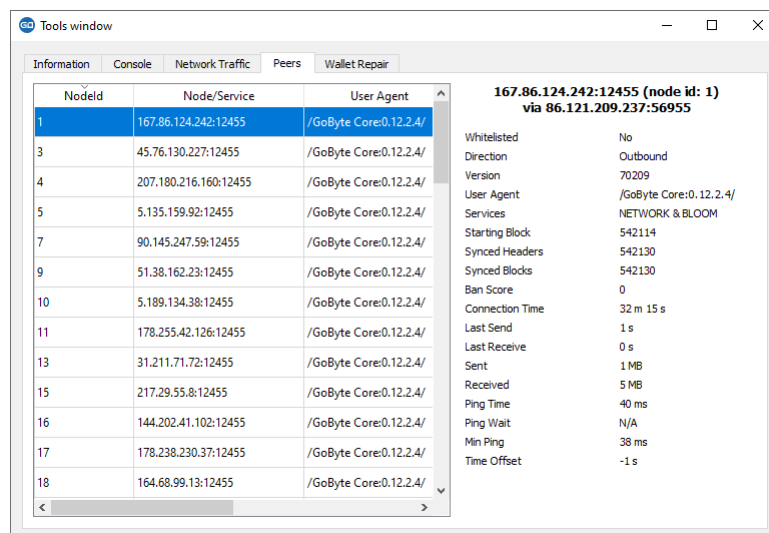


Fig. 61: The Peers tab of the GoByte Core Tools dialog

Wallet Repair tab

The Wallet Repair tab offers a range of startup commands to restore a wallet to a functional state. Selecting any of these commands will restart GoByte Core with the specified command-line option.

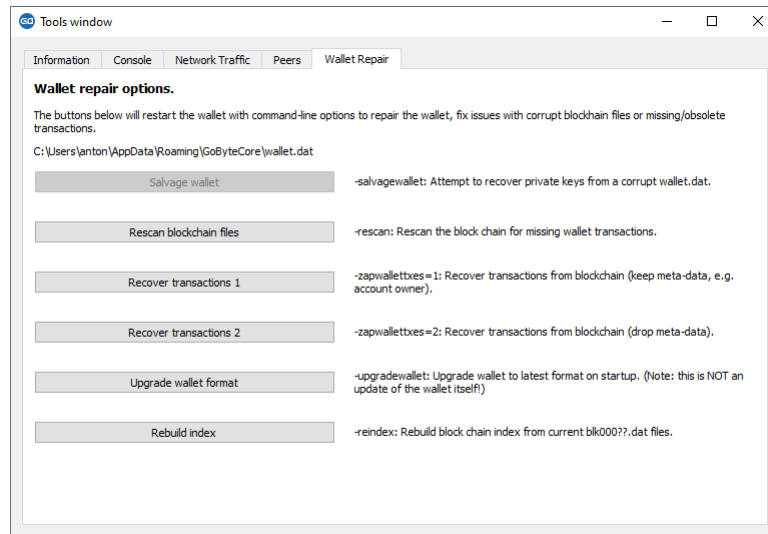


Fig. 62: The Wallet Repair tab of the GoByte Core Tools dialog

Salvage wallet Salvage wallet assumes wallet.dat is corrupted and cannot be read. It makes a copy of wallet.dat to wallet.<date>.bak and scans it to attempt to recover any private keys. Check your debug.log file after running salvage wallet and look for lines beginning with “Salvage” for more information on operations completed.

Rescan blockchain files Rescans the already downloaded blockchain for any transactions affecting accounts contained in the wallet. This may be necessary if you replace your wallet.dat file with a different wallet or a backup - the wallet logic will not know about these transactions, so a rescan is necessary to determine balances.

Recover transactions The recover transactions commands can be used to remove unconfirmed transactions from the memory pool. Your wallet will restart and rescan the blockchain, recovering existing transactions and removing unconfirmed transactions. Transactions may become stuck in an unconfirmed state if there is a conflict in protocol versions on the network during PrivateSend mixing, for example, or if a transaction is sent with insufficient fees when blocks are full.

Upgrade wallet format This command is available for very old wallets where an upgrade to the wallet version is required in addition to an update to the wallet software. You can view your current wallet version by running the `getwalletinfo` command in the console.

Rebuild index Discards the current blockchain and chainstate indexes (the database of unspent transaction outputs) and rebuilds it from existing block files. This can be useful to recover missing or stuck balances.

Sending and receiving

Your GoByte Core Wallet is associated with a number of unique addresses that can be used to send and receive GoByte. Each address holds its own balance, and the sum of all your balances is what appears on the **Overview** tab. When you send GoByte, your wallet will automatically transfer funds from as many of your addresses as necessary to the destination address, which is controlled by another GoByte user and associated with their wallet. You can control which addresses you use using the Coin Control feature.

When you confirm a transaction, GoByte Core will enter the transaction in a block, which will then be added to the blockchain for other clients to confirm. A transaction is generally considered confirmed once six blocks have been

added after the block containing your transaction, although masternode and mining payments are only released after 101 blocks. Note that a different process is used for InstantSend and PrivateSend transactions.

GoByte addresses are 34 characters long and begin with an uppercase G.

Sending GoByte

You can use GoByte Core to send GoByte from your balance to another user. The receiving user will provide you with a GoByte address to which you should send the funds. Click the **Send** tab in the tab bar and enter the destination address in the **Pay To** field.

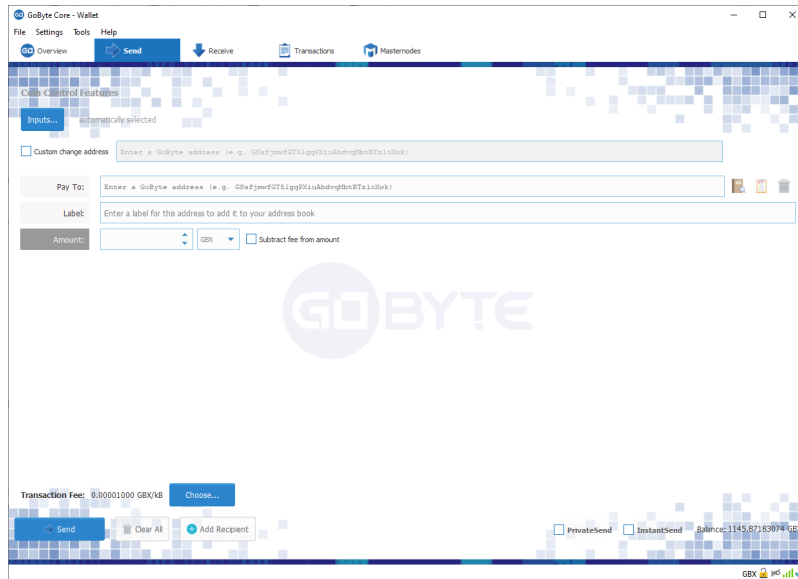



Fig. 63: The GoByte Core Send tab

You can also use the three icons  to the right of the **Pay To** field to select a previously used address, paste an address from the clipboard, or clear the current address. If this is a new address, you can enter a name for the address in the **Label** field to help identify it again later. Finally, enter the amount of GoByte you want to transfer in the **Amount** field.

The other options relate to fees and PrivateSend/InstantSend. You can choose if you want to pay the network fee in addition to the amount sent, or subtract it from the amount sent. You can also increase your fee to encourage nodes on the network to prioritize your transaction. Choosing **InstantSend** has a similar effect, but actually relies on a different mechanism in the second layer network to speed up the transaction time. Choosing **PrivateSend** will send GoByte from an address that has previously been mixed. You can find out more about PrivateSend and InstantSend [here](#).

Let's try an example. Say you have received an invoice which you now want to pay with GoByte. The writer of the invoice has included a GoByte address, which can be seen in the following window beginning with G. The invoice is for 2.45 GoByte, which you fill in the **Amount** field.

Once you have entered the destination address and the amount, click the **Send** button. If you have encrypted your wallet, you will now be required to enter your password to unlock the wallet.

Finally, you are given one final confirmation and chance to cancel your send transaction before GoByte Core processes the transaction on the blockchain.

If you respond with **Yes**, your transaction will be processed. Your operating system may display a notification, and the transaction will appear on the Transactions tab, where you can monitor its progress.

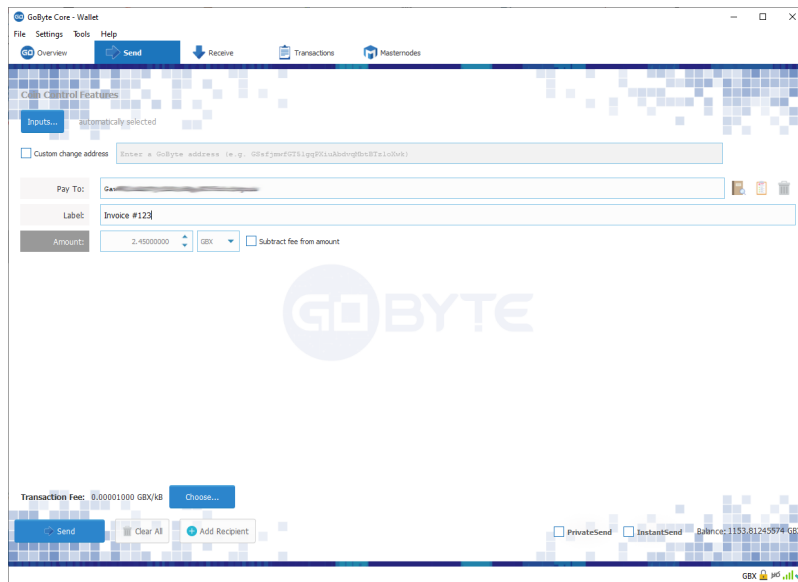


Fig. 64: The Send tab filled out for a transaction

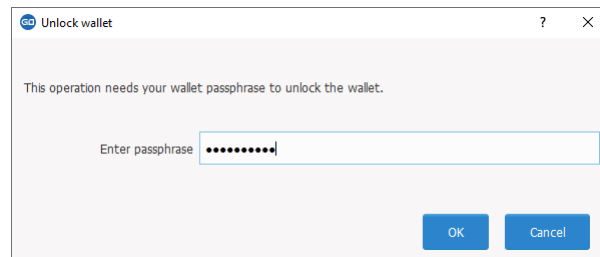


Fig. 65: Entering the password to unlock the wallet

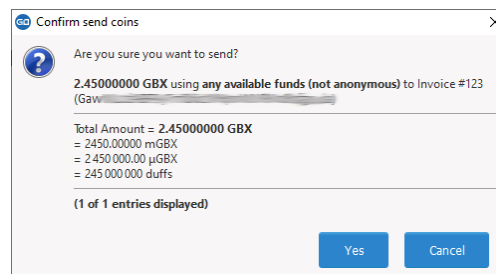


Fig. 66: Final confirmation window



Fig. 67: The Windows 10 sent transaction confirmation notification

Note that the amount of the transaction increased by .000045 GoByte. This is the transaction fee. In the next section, we will see what this procedure looks like from the receiving side.

Receiving GoByte

To receive GoByte, you must first create a receiving address to give to the sending party. To do this, click **File > Receiving addresses**. The **Receiving addresses** window appears.

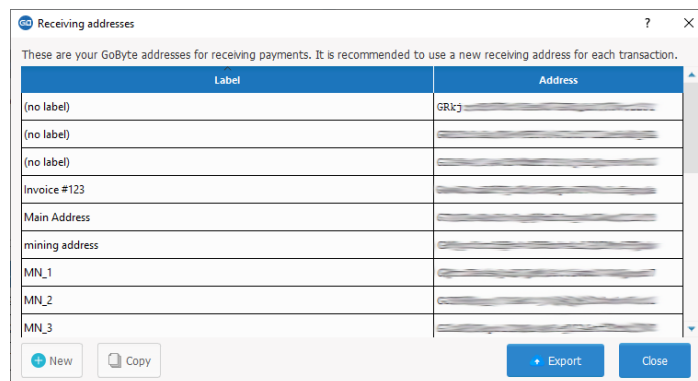


Fig. 68: The Receiving addresses window

Either copy an existing address by clicking on the address and then the **Copy** button, or create a new address by clicking the **New** button. You can also edit an existing address by right clicking and selecting **Edit** address from the context menu. Give this address to the person who will send you GoByte. Your wallet does not need to be open to receive funds, but if it is, you can watch the transaction arrive in real time. This is because your wallet constantly watches for new blocks on the blockchain when it is open, and will recognize a new transaction involving your receiving address when it occurs.

Once you have been paid, you can see the balance both on the **Overview** tab and on the **Transactions** tab.

How to Create New Receiving Addresses in GoByteQT

PrivateSend and InstantSend



Fig. 69: The Windows 10 received transaction confirmation notification

The screenshot shows the 'Gobyte Core - Wallet' application. The top navigation bar includes 'File', 'Settings', 'Tools', and 'Help'. Below this is a sub-navigation bar with 'Overview', 'Send', 'Receive', 'Transactions', and 'MasterNodes'. The 'Transactions' tab is active, displaying a table of wallet transactions. The table has columns for 'Date', 'Type', 'Address / Label', and 'Amount (GBYTE)'. The transactions listed include received payments, payments to yourself, sent payments, and mined blocks, with amounts ranging from 0.00000000 to 2.44999870 GBYTE.

Date	Type	Address / Label	Amount (GBYTE)
05-Aug-20 22:14	Received with	Invoices	2.44999870
05-Aug-20 22:13	Payment to yourself	[n/a]	0.00000000
05-Aug-20 22:10	Sent to	Invoice #123	-2.45000073
05-Aug-20 22:09	Payment to yourself	[n/a]	0.00000000
05-Aug-20 06:51	Mined	Invoice #123	7.94062500
04-Aug-20 00:36	Payment to yourself	[n/a]	0.00000000
04-Aug-20 05:41	Mined	Invoice #123	7.94062500
04-Aug-20 00:58	Received with	Invoice #123	0.49995800
04-Aug-20 00:56	Received with	[REDACTED]	0.50000000
04-Aug-20 00:53	Sent to	[REDACTED]	-1.00000026
04-Aug-20 00:49	Payment to yourself	[n/a]	0.00000000
28-Jul-20 06:45	Mined	Invoice #123	7.94062500

Fig. 70: The received transaction

PrivateSend

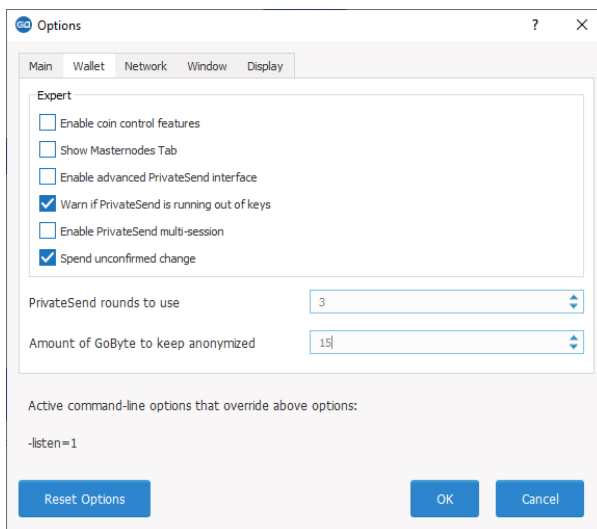
This documentation describes how to use GoByte Core to send GoByte privately. PrivateSend, released as DarkSend in RC4 of the DarkCoin client and rebranded to PrivateSend in May 2016 by Dash, is a trustless method of running a sequence of transactions (known as “mixing”) such that an external observer is unable to determine the source of funding when a PrivateSend transaction is created. This gives your GoByte the same privacy properties as cash withdrawn from an ATM, for example. The mixing and denomination process is seamless, automatic, and requires no intervention on the part of the user. The current implementation of PrivateSend in the GoByte Core wallet allows any amount of GoByte to be mixed for later use in PrivateSend transactions. PrivateSend is also available in the *GoByte Electrum* wallet.

Knowledge of the exact number of rounds of PrivateSend mixing used in any given PrivateSend transaction has a **quantifiable effect** on the confidence an adversary may have when attempting to guess the source of a PrivateSend transaction. For this reason, the recommended (and default) number of rounds of PrivateSend mixing is set to four.

You can read more about PrivateSend theory and processes [here](#).

Configuration

1. Open your GoByte Core wallet, go to **Settings** and select **Options**. Go to the **Wallet** tab.



2. Next to **PrivateSend rounds to use**, enter a value between 1-16. Each round of PrivateSend performs one denominated fund mixing transaction. Higher numbers of rounds increase your overall level of privacy while decreasing the chance of detection via node collusion. 16 is the highest number of rounds currently available.

NOTE: To prevent system abuse, an average of one in ten rounds of masternode mixing incurs a fee of .0001 GBX.

3. Enter a target value for **Amount of GoByte to keep mixed**. This value provides a lower boundary on the final amount of funds to be mixed. Depending on how the client splits your wallet balance, you may end up with denominated inputs whose sum total is greater than the target amount. In this case the client will use all existing denominated inputs in the PrivateSend process. The final mixed amount may be higher than your target, but should be close.
4. Click **OK** to save settings.
5. PrivateSend is disabled by default when you open the wallet. It will only start after you set the number of rounds and number of GoByte to mix under settings and click **Start Mixing** on the **Overview** tab of your wallet.

Starting Mixing

The PrivateSend process is initiated by clicking the **Start Mixing** button on the **Overview** tab of the GoByte Core wallet. Mixing is possible once the following conditions have been met:

- The wallet contains sufficient non-mixed funds to create the minimum required denominated values
- The user has not disabled PrivateSend in the Options dialog
- The target value for mixed Funds in the Options dialog is greater than zero

If your wallet is encrypted (highly recommended), you will be asked to enter your wallet passphrase. Enable the **Only for mixing via PrivateSend** checkbox to unlock the wallet for mixing only.

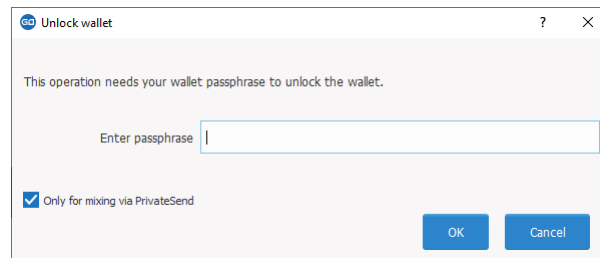


Fig. 71: Entering a password for PrivateSend mixing only

This will unlock your wallet, and the PrivateSend mixing process will begin. The wallet will remain unlocked until PrivateSend mixing is complete, at which point it will be locked automatically.

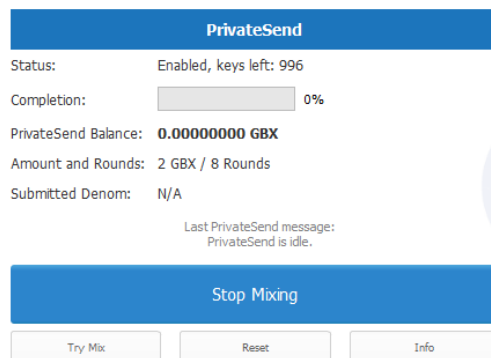


Fig. 72: PrivateSend interface after clicking the **Start Mixing** button. Note the **Status** is **Enabled**.

PrivateSend will begin creating transactions and your PrivateSend balance will gradually increase. This process can take some time, so be patient. You can monitor the process in more detail as described in the following section.

Any of the following actions will interrupt the mixing process. Because the transactions are atomic (they either take place completely, or do not take place at all), it should be possible to safely interrupt PrivateSend mixing at any time.

- Clicking the Stop Mixing button on the Overview tab
- Closing the client before PrivateSend mixing is completed
- Sending PrivateSend funds from the wallet before PrivateSend rounds are completed
- Disabling PrivateSend before the process is complete

Monitoring Mixing

If you want to monitor PrivateSend in more detail, you need to enable some advanced features of the wallet. Go to **Settings**, select **Options** and go to the **Wallet** tab. Check the boxes next to the **Enable coin control features** and **Enable advanced PrivateSend interface** options.

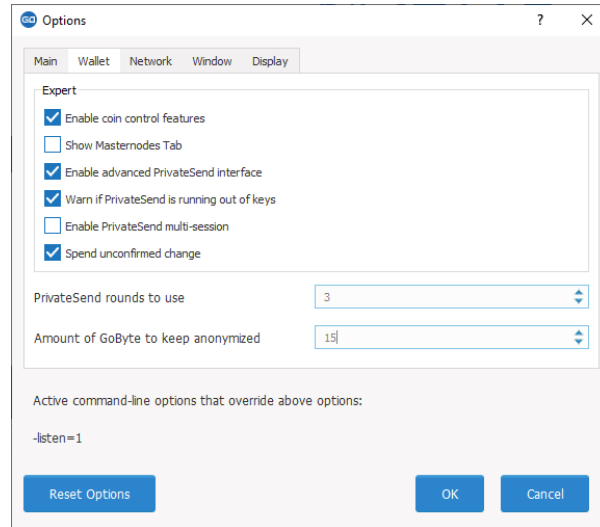


Fig. 73: Enabling advanced options for PrivateSend in the GoByte Core wallet settings

This will allow you to monitor progress and see which individual operations PrivateSend is carrying out in the background.

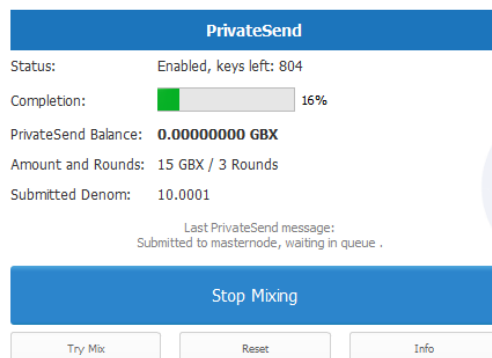


Fig. 74: Monitoring PrivateSend progress

Since PrivateSend mixing creates a lot of new address keys to send and receive the mixed denominations, you may receive a warning when the number of remaining keys runs low. This is nothing to be worried about, since the wallet will simply create more keys as necessary. However, these keys will not exist in any previous backups of your wallet. For this reason, it is important to backup your wallet again after mixing is complete.

You can also monitor PrivateSend progress by viewing the transactions created by the mixing process on the **Transactions** tab.

The following table describes the PrivateSend-related transactions displayed in the Type column of the **Transactions** tab:

Date	Type	Address / Label	Amount (GBX)
05-Aug-20 22:39	PrivateSend Create Denominations	fp (n/a)	-0.00003021
05-Aug-20 22:39	PrivateSend Create Denominations	fp (n/a)	-0.00000429
05-Aug-20 22:39	PrivateSend Create Denominations	fp (n/a)	-0.00000735
05-Aug-20 22:38	PrivateSend Create Denominations	fp (n/a)	-0.00001189
05-Aug-20 22:38	PrivateSend Create Denominations	fp (n/a)	-0.00000395
05-Aug-20 22:38	PrivateSend Create Denominations	fp (n/a)	-0.00000769
05-Aug-20 22:38	PrivateSend Create Denominations	fp (n/a)	-0.00001058
05-Aug-20 22:38	PrivateSend Create Denominations	fp (n/a)	-0.00000462
05-Aug-20 22:38	PrivateSend Create Denominations	fp (n/a)	-0.00000632
05-Aug-20 22:37	PrivateSend Create Denominations	fp (n/a)	-0.00001332
05-Aug-20 22:37	PrivateSend Create Denominations	fp (n/a)	-0.00000430
05-Aug-20 22:36	PrivateSend Create Denominations	fp (n/a)	-0.00000702
05-Aug-20 22:14	Received with	← Invoices	2.44998070
05-Aug-20 22:13	Payment to yourself	fp (n/a)	0.00000000
05-Aug-20 22:10	Sent to	→ Invoice #123	-2.45000373
05-Aug-20 22:09	Payment to yourself	fp (n/a)	0.00000000
05-Aug-20 06:51	Mined	fp Invoice #123	7.94062500
04-Aug-20 00:36	Payment to yourself	fp (n/a)	0.00000000
04-Aug-20 00:41	Mined	fp Invoice #123	7.94062500
04-Aug-20 00:58	Received with	← Invoice #123	0.49995800

Fig. 75: Transactions created by PrivateSend on the Transactions tab

PrivateSend Transaction Type	Transaction Description
PrivateSend Make Collateral Inputs (<i>Mixing</i>)	Wallet funds were moved to collateral inputs that will be used to make collateral payments. This is done to minimize traceability of collaterals.
PrivateSend Create Denominations (<i>Mixing</i>)	Wallet funds were broken into PrivateSend denominations (Step 1 here)
PrivateSend Denominate (<i>Mixing</i>)	A transaction was sent to a masternode in order to participate in a mixing session (Step 3 here)
PrivateSend Collateral Payment (<i>Mixing</i>)	The mixing session collateral was claimed. This fee is charged in ~10% of mixing sessions to prevent spam attacks.
PrivateSend (<i>Spending</i>)	Mixed funds were used to send a payment to someone. Note: Unlike the previous 4 transaction types, this is not a mixing process transaction.

You can also use the coin control feature to view which addresses hold mixed denominations ready to be used for PrivateSend transactions. Go to the **Send** tab of your wallet and click **Inputs** to view the possible input addresses for your transactions. You can see how each address holds given denominations of mixed GoByte, and how many rounds of mixing have been completed. This is to ensure that an efficient combination of addresses can be used as inputs in PrivateSend transactions without too much change, since amount in a PrivateSend transaction must be rounded up to completely spend all inputs. The current minimum balance for an input used in a PrivateSend transaction is 0.00100010 GBX.

Paying with PrivateSend

You can only use PrivateSend for payments once you have mixed enough GoByte to make up the amount you are trying to send. Because the mixing process takes time, it must be done in advance before you create the send transaction. A PrivateSend transaction is effectively the same as any other transaction on the blockchain, but it draws only from input addresses where the denomination has previously been mixed to ensure privacy of funds. Because several input addresses are usually required to make up the amount you are trying to send, a PrivateSend transaction will usually take up more space (in kilobytes) on the blockchain, and therefore will be charged a slightly higher fee.

To send a payment using PrivateSend, go to the **Send** tab of the GoByte Core wallet and enable the **PrivateSend**

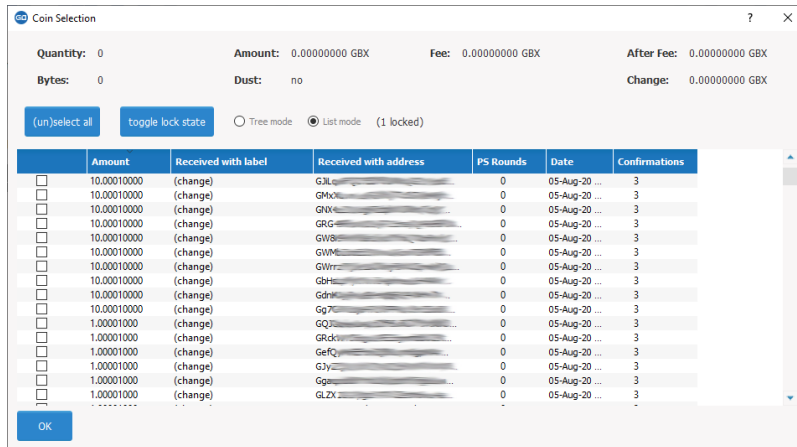


Fig. 76: Coin Selection dialog showing addresses holding PrivateSend mixed balances in different denominations

option. The balance displayed will change to show your PrivateSend balance instead of the total balance. You can then enter the **Pay To** address, **Label**, **Amount** and click **Send** as usual. Your payment will be rounded up to completely spend the lowest possible denomination of mixed balance available (currently to the nearest 0.001 GBX). You will be prompted to enter your password and receive a detailed breakdown of the fee structure for PrivateSend before sending.

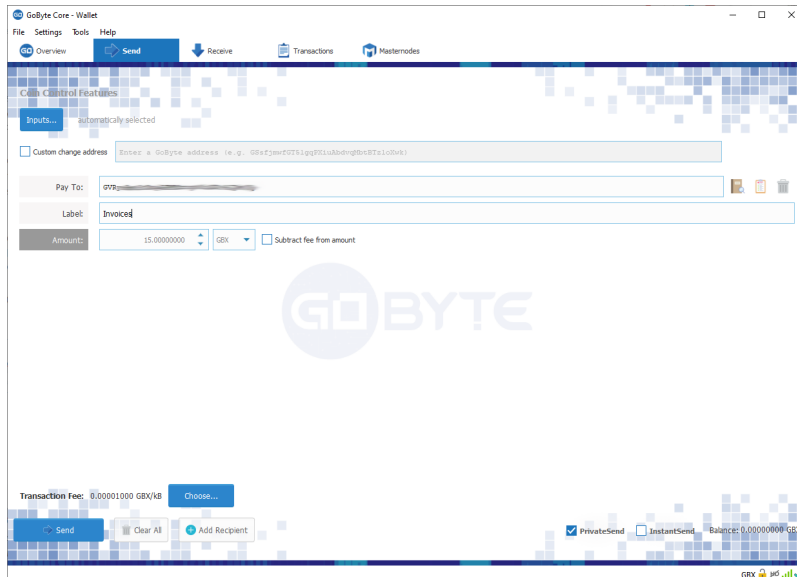


Fig. 77: GoByte Core ready to send a PrivateSend transaction. Note PrivateSend is enabled and the amount to be sent is less than the available PrivateSend balance

InstantSend

Introduction

This documentation describes how to use InstantSend to instantly send funds to any other GoByte user around the world. Since the release of InstantSend based on Long Living Masternode Quorums (LLMQ-IS) in GoByte 0.14, the GoByte network itself will attempt to generate an InstantSend lock for almost every transaction processed by the network, according to these rules. Unlike the prior implementation of InstantSend, which required a higher fee,

LLMQ-IS locks transactions without any action taken by the user. For this reason, only the recipient needs to monitor the network for the transaction lock in order to be able to receive funds and process transactions with immediate settlement.

GoByte InstantSend is supported by many wallets and vendors, including (but not limited to) the following:

- GoByte Core Wallet
- GoByte Android Wallet
- GoByte iOS Wallet
- My GoByte Wallet
- and many more...

You can read more about InstantSend theory and processes [here](#).

InstantSend Transactions

Since GoByte 0.14, all user-facing interface options to create an InstantSend transaction have been removed, because all transactions are effectively InstantSend transactions. As before, the recipient is responsible for monitoring the network for the InstantSend lock and implementing user-facing logic and interfaces to continue with transaction processing after payment has been received. See the InstantSend Integration documentation for more information on how to monitor for InstantSend locks.

Wallet backup and restore

Backup

This documentation describes how to safely back up your wallet file for safe storage in case your computer or laptop is damaged or lost. GoByte Core stores all data necessary to control your GoByte addresses in a single file called *wallet.dat*. This wallet is in the Berkeley DB format and stores the pairs of private/public cryptographic keys used to manage your balances on the GoByte blockchain. GoByte Core makes most of these operations transparent and even generates automatic backups of your wallet file in case it is corrupted, but the user is responsible for ensuring that these backups are stored in a safe place. **If you lose access to your wallet file, you will permanently lose access to your GoByte.**

It is important to consider that if you have not encrypted your wallet using the **Settings > Encrypt Wallet** menu item, anyone with access to the backed up wallet.dat file will immediately have full access to your GoByte. If you do choose to encrypt your wallet, do not store the password in the same place as the wallet.dat file, particularly if you are saving the backup to the cloud.

Backup from GoByte Core

Firstly, never copy your wallet.dat file while GoByte Core is open. Always use the **File > Backup Wallet** menu if the wallet is open. When you select this menu item, a dialog box will appear to specify where the file should be saved. Enter a name for the file, select a location and click **Save**. The example below shows saving the file to a USB stick. Keep this file in a physically separate location to your computer.

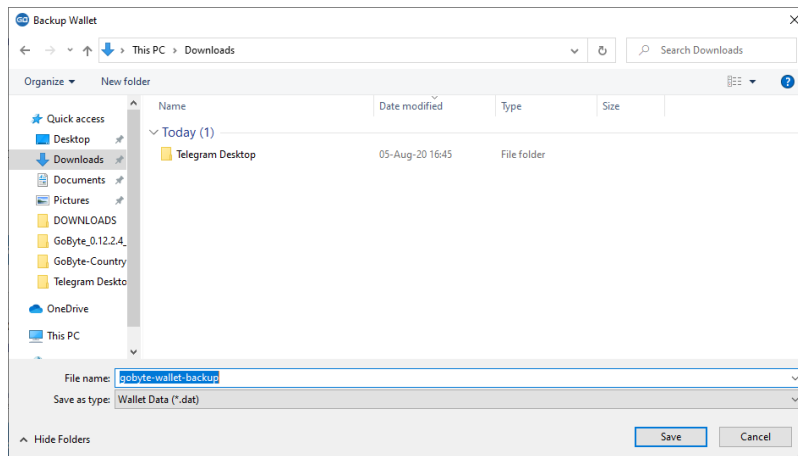
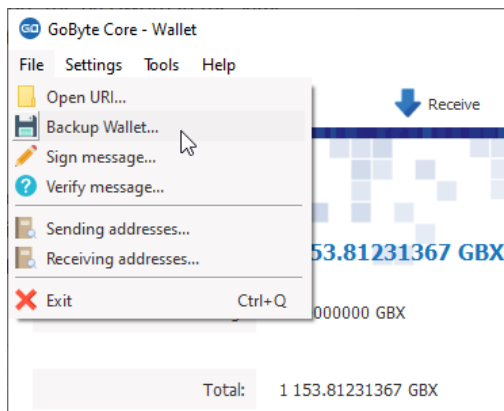


Fig. 78: Backing up the GoByte Core wallet from the File menu

Backup by copying wallet.dat

If GoByte Core is not running, you can also backup your wallet by simply copying the *wallet.dat* file to another location. This file is located in the *GoByteCore* data folder. You were given the option to specify the location of this folder during installation, but by default the folder is in the following locations on different operating systems:

- **Windows**

```
C:\Users\YourUserName\AppData\Roaming\GoByteCore
```

You can access this folder directly by **Windows Key + R** and typing `%APPDATA%\GoByteCore`

- **Linux**

```
/home/YourUserName/.gobytecore
```

You can access this folder directly by typing `cd ~/.gobytecore` at the terminal or `~/.gobytecore` in the path bar using the **Go > Enter Location...** menu item in Files

- **macOS**

```
/Users/YourUserName/Library/Application Support/GoByteCore
```

You can access this folder by typing `cd ~/Library/Application Support/GoByteCore` at the terminal or `~/Library/Application Support/GoByteCore` in dialog at the **Go > Go To Folder** menu item in Finder

Ensure GoByte Core is not running, then simply copy the *wallet.dat* file from this folder to another folder in the normal way for your operating system. The example below shows copying the file to a USB stick using simple drag and drop while holding down **Ctrl** on a Windows system. On most operating systems, you can also right click on the file and select **Copy**, then select **Paste** in the target folder. Keep this file in a physically separate location to your computer. Be careful to copy (not move) the file!

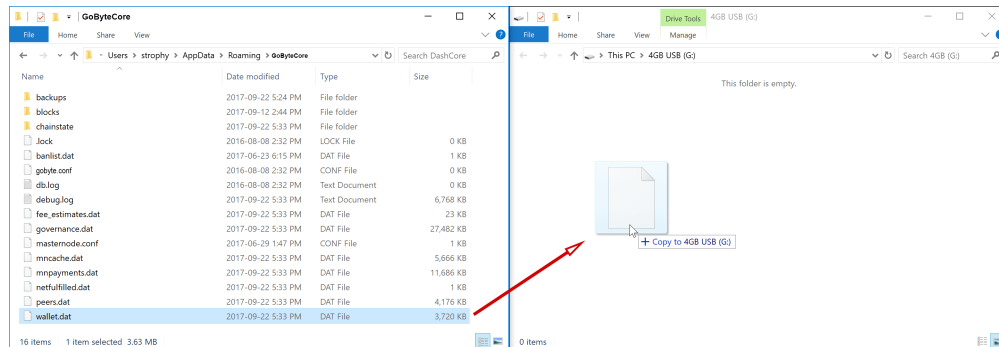


Fig. 79: Backing up wallet.dat by copying to another folder

Automatic backups

Every time you open GoByte Core, it will automatically create a backup copy of *wallet.dat* in the *gobytecore/backups* folder. Up to 10 backups can be kept here by default, and the oldest backup will be deleted as each additional new backup is created. You can modify the number of backups kept here using the `-createwalletbackups=n` parameter at the command line or in *gobyte.conf*. Setting this value to 0 completely disables backups.

You can view the automatic backups folder by browsing to *GoByteCore* folder at the location specified above for *wallet.dat* and opening the *backups* folder, or by selecting **Tools > Show Automatic Backups** from the menu in GoByte Core. Since these files are not active when GoByte Core is running, you can safely copy them at any time. They are also a handy backup if the original files in the *GoByteCore* folder become corrupted due to improper shutdown of the GoByte Core app.

Restore

To restore a backup, install GoByte Core on the target system (or stop it, if already installed) and rename the existing *wallet.dat* file in the *GoByteCore* folder.

Then copy the backup wallet file to the *GoByteCore* folder and ensure it is named *wallet.dat*. Now, when you start GoByte Core again, it will load the new wallet. Do not replace *wallet.dat* while GoByte Core is running, since this will result in data corruption!

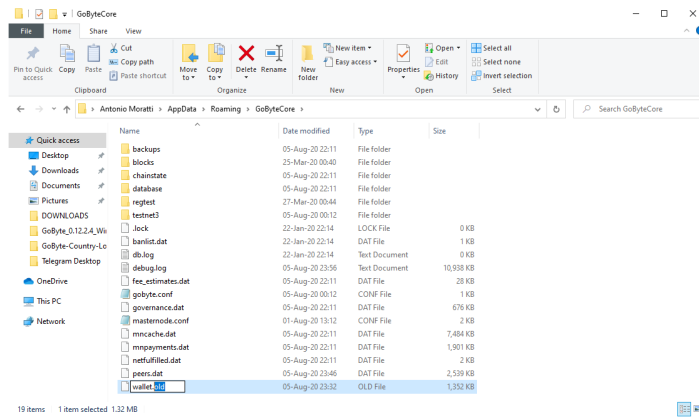


Fig. 80: Renaming the old wallet.dat file to wallet.old in the GoByteCore folder

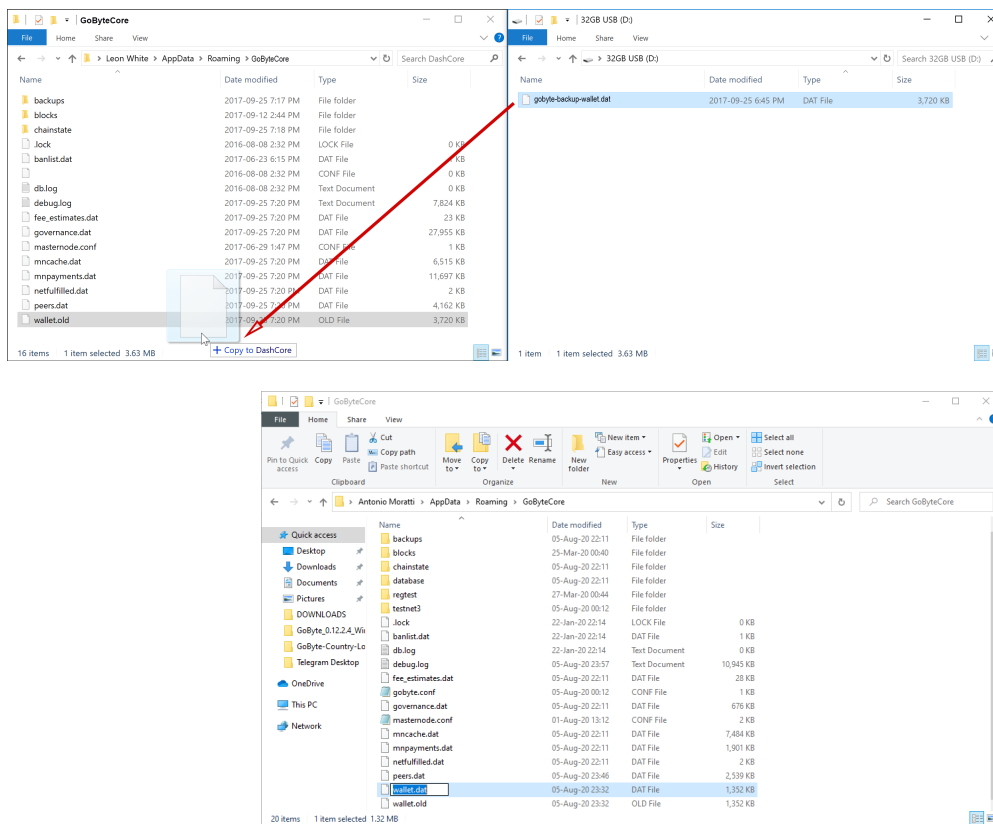


Fig. 81: Copying the backup file into the GoByteCore folder and renaming it to wallet.dat

Backup storage considerations

Any backup depends to some extent on the existence of software capable of reading the data at some future point in time. As such, it is good practice to store a copy of the software used to create the backup together with the backup file itself. In this case, this would be a copy of the version of GoByte Core you used to create the backup.

The *wallet.dat* file itself should be encrypted using a password set from the **Settings > Encrypt Wallet** menu item. However, this only prevents someone with access to the file from creating transactions, not from opening the file. You

could additionally store the file in another encrypted container, such as a USB stick using [BitLocker](#) in Windows, [LUKS](#) in Linux or [FileVault](#) on macOS. It is also possible to create [disk images](#) or [zip files](#) with password encryption - the choice is yours. For further reading on encrypted containers, see [here](#).

Where you store this file is then up to you. It could be in your home, in a safe deposit box at a bank, a waterproof or fireproof container, or on cloud storage such as Google Drive, Dropbox or iCloud. Consider where you will store any passwords needed to restore access to the wallet (in your head, on paper, in a password manager, etc.) and who may need access to the password in the future.

Finally it is important to understand that *wallet.dat* itself is a relatively dangerous way to store large amounts of funds - it is simply a database file storing private keys. While the convenience of storing a wallet file directly is fine for smaller holdings, it is more secure to store large amounts of GoByte on a single predefined address in a way that you are guaranteed access through any software supporting the protocol, rather than a specific implementation of that protocol. If you are interested in this, read more about paper wallets, where the private key can be printed directly or also encrypted using BIP38 for safe storage.

Verifying backups

There is no fixed procedure to verify your backup, but you should test restoring it at least once to make sure it works. If you have a simple copy of the file, try to restore it to your current *GoByteCore* folder and start GoByte Core to make sure it opens without any errors. If you decided to store the file in an encrypted zip file, make sure you can unzip it and that it opens correctly in GoByte Core. In short, make sure that you (or the person you are trusting to eventually go through this process for you) can actually reverse your backup process to get access to your GoByte, instead of relying on the fact that this process should theoretically be reversible.

Arguments and commands

All command-line options (except for `-datadir` and `-conf`) may be specified in a configuration file, and all configuration file options may also be specified on the command line. Command-line options override values set in the configuration file. The configuration file is a list of `setting=value` pairs, one per line, with optional comments starting with the `#` character.

The configuration file is not automatically created; you can create it using your favorite plain-text editor. By default, `gobyte-qt` (or `gobyted`) will look for a file named `gobyte.conf` in the gobyte data directory, but both the data directory and the configuration file path may be changed using the `-datadir` and `-conf` command-line arguments.

Platform	Path to data folder	Typical path to configuration file
Linux	~/	/home/username/.gobytecore/gobyte.conf
macOS	~/Library/Application Support/	/Users/username/Library/Application Support/GoByteCore/gobyte.conf
Windows	%APPDATA%	(Vista-10) C:\Users\username\AppData\Roaming\GoByteCore\gobyte.conf (2000-XP) C:\Documents and Settings\username\Application Data\GoByteCore\gobyte.conf

Note: if running GoByte in testnet mode, the sub-folder `testnet3` will be appended to the data directory automatically.

Command line arguments

These commands are accurate as of GoByte Core version 0.12.2.1.

- *gobyted*
- *gobyte-qt*
- *gobyte-cli*
- *gobyte-tx*

gobyted

GoByte Core Daemon

Usage

gobyted [**options**] Start GoByte Core Daemon

Options

--help	This help message
--version	Print version and exit
--alerts	Receive and display P2P network alerts (default: 1)
--alertnotify=<cmd>	Execute command when a relevant alert is received or we see a really long fork (%s in cmd is replaced by message)
--blocknotify=<cmd>	Execute command when the best block changes (%s in cmd is replaced by block hash)
--assumevalid=<hex>	If this block is in the chain assume that it and its ancestors are valid and potentially skip their script verification (0 to verify all, default: 00000000000000b4181bbdddbae464ce11fede5d0292fb63fdede1e7c8ab21c, testnet: 00000ce22113f3eb8636e225d6a1691e132fdd587aed993e1bc9b07a0235eea4)
--conf=<file>	Specify configuration file (default: gobyte.conf)
--daemon	Run in the background as a daemon and accept commands
--datadir=<dir>	Specify data directory
--dbcache=<n>	Set database cache size in megabytes (4 to 16384, default: 100)
--loadblock=<file>	Imports blocks from external blk000???.dat file on startup
--maxorphantx=<n>	Keep at most <n> unconnectable transactions in memory (default: 100)
--maxmempool=<n>	Keep the transaction memory pool below <n> megabytes (default: 300)
--mempoolexpiry=<n>	Do not keep transactions in the mempool longer than <n> hours (default: 72)
--par=<n>	Set the number of script verification threads (-1 to 16, 0 = auto, <0 = leave that many cores free, default: 0)
--pid=<file>	Specify pid file (default: gobyted.pid)
--prune=<n>	Reduce storage requirements by pruning (deleting) old blocks. This mode is incompatible with -txindex and -rescan. Warning: Reverting this setting requires re-downloading the entire blockchain. (default: 0 = disable pruning blocks, >945 = target size in MiB to use for block files)

--reindex-chainstate	Rebuild chain state from the currently indexed blocks
--reindex	Rebuild chain state and block index from the blk*.dat files on disk
--sysperms	Create new files with system default permissions, instead of umask 077 (only effective with disabled wallet functionality)
--txindex	Maintain a full transaction index, used by the getrawtransaction rpc call (default: 1)
--addressindex	Maintain a full address index, used to query for the balance, txids and unspent outputs for addresses (default: 0)
--timestampindex	Maintain a timestamp index for block hashes, used to query blocks hashes by a range of timestamps (default: 0)
--spentindex	Maintain a full spent index, used to query the spending txid and input index for an outpoint (default: 0)

Connection options

--addnode=<ip>	Add a node to connect to and attempt to keep the connection open
--banscore=<n>	Threshold for disconnecting misbehaving peers (default: 100)
--bantime=<n>	Number of seconds to keep misbehaving peers from reconnecting (default: 86400)
--bind=<addr>	Bind to given address and always listen on it. Use [host]:port notation for IPv6
--connect=<ip>	Connect only to the specified node(s)
--discover	Discover own IP addresses (default: 1 when listening and no -externalip or -proxy)
--dns	Allow DNS lookups for -addnode, -seednode and -connect (default: 1)
--dnsseed	Query for peer addresses via DNS lookup, if low on addresses (default: 1 unless -connect)
--externalip=<ip>	Specify your own public address
--forcednsseed	Always query for peer addresses via DNS lookup (default: 0)
--listen	Accept connections from outside (default: 1 if no -proxy or -connect)
--listenonion	Automatically create Tor hidden service (default: 1)
--maxconnections=<n>	Maintain at most <n> connections to peers (temporary service connections excluded) (default: 125)
--maxreceivebuffer=<n>	Maximum per-connection receive buffer, <n>*1000 bytes (default: 5000)
--maxsendbuffer=<n>	Maximum per-connection send buffer, <n>*1000 bytes (default: 1000)
--onion=<ip:port>	Use separate SOCKS5 proxy to reach peers via Tor hidden services (default: -proxy)
--onlynet=<net>	Only connect to nodes in network <net> (ipv4, ipv6 or onion)
--permitbaremultisig	Relay non-P2SH multisig (default: 1)
--peerbloomfilters	Support filtering of blocks and transaction with bloom filters (default: 1)
--port=<port>	Listen for connections on <port> (default: 12455 or testnet: 13455)

--proxy=<ip:port>	Connect through SOCKS5 proxy
--proxyrandomize	Randomize credentials for every proxy connection. This enables Tor stream isolation (default: 1)
--seednode=<ip>	Connect to a node to retrieve peer addresses, and disconnect
--timeout=<n>	Specify connection timeout in milliseconds (minimum: 1, default: 5000)
--torcontrol=<ip:port>	Tor control port to use if onion listening enabled (default: 127.0.0.1:9051)
--torpassword=<pass>	Tor control port password (default: empty)
--upnp	Use UPnP to map the listening port (default: 0)
--whitebind=<addr>	Bind to given address and whitelist peers connecting to it. Use [host]:port notation for IPv6
--whitelist=<netmask>	Whitelist peers connecting from the given netmask or IP address. Can be specified multiple times. Whitelisted peers cannot be DoS banned and their transactions are always relayed, even if they are already in the mempool, useful e.g. for a gateway
--whitelistrelay	Accept relayed transactions received from whitelisted peers even when not relaying transactions (default: 1)
--whitelistforcerelay	Force relay of transactions from whitelisted peers even they violate local relay policy (default: 1)
--maxuploadtarget=<n>	Tries to keep outbound traffic under the given target (in MiB per 24h), 0 = no limit (default: 0)

Wallet options

--disablewallet	Do not load the wallet and disable wallet RPC calls
--keypool=<n>	Set key pool size to <n> (default: 1000)
--fallbackfee=<amt>	A fee rate (in GBX/kB) that will be used when fee estimation has insufficient data (default: 0.0002)
--mintxfee=<amt>	Fees (in GBX/kB) smaller than this are considered zero fee for transaction creation (default: 0.0001)
--paytxfee=<amt>	Fee (in GBX/kB) to add to transactions you send (default: 0.00)
--rescan	Rescan the block chain for missing wallet transactions on startup
--salvagewallet	Attempt to recover private keys from a corrupt wallet.dat on startup
--sendfreetransactions	Send transactions as zero-fee transactions if possible (default: 0)
--spendzeroconfchange	Spend unconfirmed change when sending transactions (default: 1)
--txconfirmtarget=<n>	If paytxfee is not set, include enough fee so transactions begin confirmation on average within n blocks (default: 2)
--maxtxfee=<amt>	Maximum total fees (in GBX) to use in a single wallet transaction; setting this too low may abort large transactions (default: 0.20)
--usehd	Use hierarchical deterministic key generation (HD) after bip39/bip44. Only has effect during wallet creation/first start (default: 0)

- mnemonic** User defined mnemonic for HD wallet (bip39). Only has effect during wallet creation/first start (default: randomly generated)
- mnemonicpassphrase** User defined mnemonic passphrase for HD wallet (bip39). Only has effect during wallet creation/first start (default: empty string)
- hdseed** User defined seed for HD wallet (should be in hex). Only has effect during wallet creation/first start (default: randomly generated)
- upgradewallet** Upgrade wallet to latest format on startup
- wallet=<file>** Specify wallet file (within data directory) (default: wallet.dat)
- walletbroadcast** Make the wallet broadcast transactions (default: 1)
- walletnotify=<cmd>** Execute command when a wallet transaction changes (%s in cmd is replaced by TxID)
- zapwallettxes=<mode>** Delete all wallet transactions and only recover those parts of the blockchain through -rescan on startup (1 = keep tx meta data e.g. account owner and payment request information, 2 = drop tx meta data)
- createwalletbackups=<n>** Number of automatic wallet backups (default: 10)
- walletbackupsdir=<dir>** Specify full path to directory for automatic wallet backups (must exist)
- keepass** Use KeePass 2 integration using KeePassHttp plugin (default: 0)
- keepassport=<port>** Connect to KeePassHttp on port <port> (default: 19455)
- keepasskey=<key>** KeePassHttp key for AES encrypted communication with KeePass
- keepassid=<name>** KeePassHttp id for the established association
- keepassname=<name>** Name to construct url for KeePass entry that stores the wallet passphrase

ZeroMQ notification options

- zmqpubhashblock=<address>** Enable publish hash block in <address>
- zmqpubhashtx=<address>** Enable publish hash transaction in <address>
- zmqpubhashtxlock=<address>** Enable publish hash transaction (locked via InstantSend) in <address>
- zmqpubhashgovernancevote=<address>** Enable publish hash of governance votes in <address>
- zmqpubhashgovernanceobject=<address>** Enable publish hash of governance objects (like proposals) in <address>
- zmqpubhashinstantsenddoublepend=<address>** Enable publish transaction hashes of attempted InstantSend double spend in <address>
- zmqpubrawblock=<address>** Enable publish raw block in <address>
- zmqpubrawtx=<address>** Enable publish raw transaction in <address>
- zmqpubrawtxlock=<address>** Enable publish raw transaction (locked via InstantSend) in <address>
- zmqpubrawinstantsenddoublepend=<address>** Enable publish raw transactions of attempted InstantSend double spend in <address>

Debugging/Testing options

- uacomment=<cmt>** Append comment to the user agent string
- debug=<category>** Output debugging information (default: 0, supplying <category> is optional). If <category> is not supplied or if <category> = 1, output all debugging information. <category> can be: addrman, alert, bench, coindb, db, http, libevent, lock, mempool, mempoolrej, net, proxy, prune, rand, reindex, rpc, selectcoins, tor, zmq, gobyte (or specifically: gobject, instantsend, keepass, masternode, mnpayments, mnsync, privatesend, spork).
- gen** Generate coins (default: 0)
- genproclimit=<n>** Set the number of threads for coin generation if enabled (-1 = all cores, default: 1)
- help-debug** Show all debugging options (usage: --help -help-debug)
- logips** Include IP addresses in debug output (default: 0)
- logtimestamps** Prepend debug output with timestamp (default: 1)
- minrelaytxfee=<amt>** Fees (in GBX/kB) smaller than this are considered zero fee for relaying, mining and transaction creation (default: 0.0001)
- printtoconsole** Send trace/debug info to console instead of debug.log file
- printtodebuglog** Send trace/debug info to debug.log file (default: 1)
- shrinkdebugfile** Shrink debug.log file on client startup (default: 1 when no -debug)

Chain selection options

- testnet** Use the test chain
- litemode=<n>** Disable all GoByte specific functionality (Masternodes, PrivateSend, InstantSend, Governance) (0-1, default: 0)

Masternode options

- masternode=<n>** Enable the client to act as a masternode (0-1, default: 0)
- mnconf=<file>** Specify masternode configuration file (default: masternode.conf)
- mnconflock=<n>** Lock masternodes from masternode configuration file (default: 1)
- masternodeprivkey=<n>** Set the masternode private key

PrivateSend options

- enableprivatesend=<n>** Enable use of automated PrivateSend for funds stored in this wallet (0-1, default: 0)
- privatesendmultisession=<n>** Enable multiple PrivateSend mixing sessions per block, experimental (0-1, default: 0)
- privatesendrounds=<n>** Use N separate masternodes for each denominated input to mix funds (2-16, default: 2)

- privatesendamount=<n>** Keep N GBX anonymized (default: 1000)
- liquidityprovider=<n>** Provide liquidity to PrivateSend by infrequently mixing coins on a continual basis (0-100, default: 0, 1=very frequent, high fees, 100=very infrequent, low fees)

InstantSend options

- enableinstant send=<n>** Enable InstantSend, show confirmations for locked transactions (0-1, default: 1)
- instant senddepth=<n>** Show N confirmations for a successfully locked transaction (0-9999, default: 5)
- instant sendnotify=<cmd>** Execute command when a wallet InstantSend transaction is successfully locked (%s in cmd is replaced by TxID)

Node relay options

- bytespersigop** Minimum bytes per sigop in transactions we relay and mine (default: 20)
- datacarrier** Relay and mine data carrier transactions (default: 1)
- datacarriersize** Maximum size of data in data carrier transactions we relay and mine (default: 83)
- mempoolreplacement** Enable transaction replacement in the memory pool (default: 0)

Block creation options

- blockminsize=<n>** Set minimum block size in bytes (default: 0)
- blockmaxsize=<n>** Set maximum block size in bytes (default: 750000)
- blockprioritysize=<n>** Set maximum size of high-priority/low-fee transactions in bytes (default: 10000)

RPC server options

- server** Accept command line and JSON-RPC commands
- rest** Accept public REST requests (default: 0)
- rpcbind=<addr>** Bind to given address to listen for JSON-RPC connections. Use [host]:port notation for IPv6. This option can be specified multiple times (default: bind to all interfaces)
- rpccookiefile=<loc>** Location of the auth cookie (default: data dir)
- rpcuser=<user>** Username for JSON-RPC connections
- rpcpassword=<pw>** Password for JSON-RPC connections
- rpcauth=<userpw>** Username and hashed password for JSON-RPC connections. The field <userpw> comes in the format: <USERNAME>:<SALT>\${HASH}. A canonical python script is included in share/rpcuser. This option can be specified multiple times

- rpcport=<port>** Listen for JSON-RPC connections on <port> (default: 9998 or testnet: 19998)
- rpccallowip=<ip>** Allow JSON-RPC connections from specified source. Valid for <ip> are a single IP (e.g. 1.2.3.4), a network/netmask (e.g. 1.2.3.4/255.255.255.0) or a network/CIDR (e.g. 1.2.3.4/24). This option can be specified multiple times
- rpcthreads=<n>** Set the number of threads to service RPC calls (default: 4)

gobyte-qt

GoByte Core QT GUI, use same command line options as gobyted with additional options for UI as described below.

Usage

gobyte-qt [command-line options] Start GoByte Core QT GUI

Wallet options

- windowtitle=<name>** Wallet window title

Debugging/Testing options

- debug=<category>** Output debugging information (default: 0, supplying <category> is optional). If <category> is not supplied or if <category> = 1, output all debugging information. <category> can be: addrman, alert, bench, coindb, db, http, libevent, lock, mempool, mempoolrej, net, proxy, prune, rand, reindex, rpc, selectcoins, tor, zmq, gobyte (or specifically: gobject, instantsend, keepass, masternode, mnpayments, mnsync, privatesend, spork), qt.

UI options

- choosedatadir** Choose data directory on startup (default: 0)
- lang=<lang>** Set language, for example “de_DE” (default: system locale)
- min** Start minimized
- rootcertificates=<file>** Set SSL root certificates for payment request (default: -system-)
- splash** Show splash screen on startup (default: 1)
- resetguisettings** Reset all settings changed in the GUI

gobyte-cli

GoByte Core RPC client

Usage

gobyte-cli [options] <command> [params] Send command to GoByte Core

gobyte-cli [options] **help** List commands

gobyte-cli [options] **help** <command> Get help for a command

Options

--help	This help message
--conf=<file>	Specify configuration file (default: gobyte.conf)
--datadir=<dir>	Specify data directory

Chain selection options

--testnet	Use the test chain
--regtest	Enter regression test mode, which uses a special chain in which blocks can be solved instantly. This is intended for regression testing tools and app development.
--rpconnect=<ip>	Send commands to node running on <ip> (default: 127.0.0.1)
--rpcport=<port>	Connect to JSON-RPC on <port> (default: 9998 or testnet: 19998)
--rpcwait	Wait for RPC server to start
--rpcuser=<user>	Username for JSON-RPC connections
--rpcpassword=<pw>	Password for JSON-RPC connections
--rpcclienttimeout=<n>	Timeout during HTTP requests (default: 900)

gobyte-tx

GoByte Core gobyte-tx utility

Usage

gobyte-tx [options] <hex-tx> [commands] Update hex-encoded gobyte transaction

gobyte-tx [options] **-create** [commands] Create hex-encoded gobyte transaction

Options

--help	This help message
--create	Create new, empty TX.
--json	Select JSON output
--txid	Output only the hex-encoded transaction id of the resultant transaction.

Chain selection options

--testnet	Use the test chain
--regtest	Enter regression test mode, which uses a special chain in which blocks can be solved instantly. This is intended for regression testing tools and app development.

Commands

delin=N Delete input N from TX

delout=N Delete output N from TX

in=TXID:VOUT Add input to TX

locktime=N Set TX lock time to N

nversion=N Set TX version to N

outaddr=VALUE:ADDRESS Add address-based output to TX

outdata=[VALUE:]DATA Add data-based output to TX

outscript=VALUE:SCRIPT Add raw script output to TX

sign=SIGHASH-FLAGS Add zero or more signatures to transaction. This command requires JSON registers: `prevtxs=JSON object`, `privatekeys=JSON object`. See `signrawtransaction` docs for format of sighash flags, JSON objects.

Register Commands

load=NAME:FILENAME Load JSON file `FILENAME` into register `NAME`

set=NAME:JSON-STRING Set register `NAME` to given `JSON-STRING`

RPC commands

This documentation lists all available RPC commands as of GoByte version 0.12.2.1, and limited documentation on what each command does. For full documentation of arguments, results and examples, type `help ("command")` to view full details at the console. You can enter commands either from **Tools > Debug** console in the QT wallet, or using *gobyte-cli* for headless wallets and *gobyted*.

Addressindex

getaddressbalance Returns the balance for an address(es) (requires addressindex to be enabled).

getaddressdeltas Returns all changes for an address (requires addressindex to be enabled).

getaddressmempool Returns all mempool deltas for an address (requires addressindex to be enabled).

getaddresstxids Returns the txids for an address(es) (requires addressindex to be enabled).

getaddressutxos Returns all unspent outputs for an address (requires addressindex to be enabled).

Blockchain

getbestblockhash Returns the hash of the best (tip) block in the longest block chain.

getblock “hash” (verbose) If verbose is false, returns a string that is serialized, hex-encoded data for block ‘hash’. If verbose is true, returns an Object with information about block <hash>.

getblockchaininfo Returns an object containing various state info regarding block chain processing.

getblockcount Returns the number of blocks in the longest block chain.

getblockhash index Returns hash of block in best-block-chain at index provided.

getblockhashes timestamp Returns array of hashes of blocks within the timestamp range provided.

getblockheader “hash” (verbose) If verbose is false, returns a string that is serialized, hex-encoded data for block-header ‘hash’. If verbose is true, returns an Object with information about blockheader <hash>.

getblockheaders “hash” (count verbose) Returns an array of items with information about <count> blockheaders starting from <hash>. If verbose is false, each item is a string that is serialized, hex-encoded data for a single blockheader. If verbose is true, each item is an Object with information about a single blockheader.

getchaintips (count branchlen) Return information about all known tips in the block tree, including the main chain as well as orphaned branches.

getdifficulty Returns the proof-of-work difficulty as a multiple of the minimum difficulty.

getmempoolinfo Returns details on the active state of the TX memory pool.

getrawmempool (verbose) Returns all transaction ids in memory pool as a json array of string transaction ids.

getspentinfo Returns the txid and index where an output is spent.

gettxout “txid” n (includemempool) Returns details about an unspent transaction output.

gettxoutproof [“txid”,...] (blockhash) Returns a hex-encoded proof that “txid” was included in a block.

gettxoutsetinfo Returns statistics about the unspent transaction output set. Note this call may take some time.

verifychain (checklevel numblocks) Verifies blockchain database.

verifytxoutproof “proof” Verifies that a proof points to a transaction in a block, returning the transaction it commits to and throwing an RPC error if the block is not in our best chain.

Control

debug (0 | 1 | addrman | alert | bench | coindb | db | lock | rand | rpc | selectcoins | mempool | mempoolrej | net | proxy | prune | h
Change debug category on the fly. Specify single category or use comma to specify many.

getinfo Deprecated. Returns an object containing various state info.

help (“command”) List all commands, or get help for a specified command.

stop Stop GoByte Core server.

GoByte

getgovernanceinfo Returns an object containing governance parameters.

getpoolinfo Returns an object containing mixing pool related information.

getsuperblockbudget index Returns the absolute maximum sum of superblock payments allowed.

gobject “command”... Manage governance objects. Available commands:

- check** Validate governance object data (proposal only)
- prepare** Prepare governance object by signing and creating tx
- submit** Submit governance object to network
- deserialize** Deserialize governance object from hex string to JSON
- count** Count governance objects and votes
- get** Get governance object by hash
- getvotes** Get all votes for a governance object hash (including old votes)
- getcurrentvotes** Get only current (tallying) votes for a governance object hash (does not include old votes)
- list** List governance objects (can be filtered by signal and/or object type)
- diff** List differences since last diff
- vote-alias** Vote on a governance object by masternode alias (using masternode.conf setup)
- vote-conf** Vote on a governance object by masternode configured in gobyte.conf
- vote-many** Vote on a governance object by all masternodes (using masternode.conf setup)

masternode “command”... Set of commands to execute masternode related actions. Available commands:

- count** Print number of all known masternodes (optional: ‘ps’, ‘enabled’, ‘all’, ‘qualify’)
- current** Print info on current masternode winner to be paid the next block (calculated locally)
- genkey** Generate new masternodeprivkey
- outputs** Print masternode compatible outputs
- start-alias** Start single remote masternode by assigned alias configured in masternode.conf
- start-<mode>** Start remote masternodes configured in masternode.conf (<mode>: ‘all’, ‘missing’, ‘disabled’)
- status** Print masternode status information
- list** Print list of all known masternodes (see masternodelist for more info)
- list-conf** Print masternode.conf in JSON format
- winner** Print info on next masternode winner to vote for
- winners** Print list of masternode winners

masternodebroadcast “command”... Set of commands to create and relay masternode broadcast messages. Available commands:

- create-alias** Create single remote masternode broadcast message by assigned alias configured in masternode.conf
- create-all** Create remote masternode broadcast messages for all masternodes configured in masternode.conf
- decode** Decode masternode broadcast message
- relay** Relay masternode broadcast message to the network

masternodelist (“mode” “filter”) Get a list of masternodes in different modes

mnsync [status|next|reset] Returns the sync status, updates to the next step or resets it entirely.

privatesend “command” Available commands:

start Start mixing
stop Stop mixing
reset Reset mixing

sentinelping version Sentinel ping.

spork <name> [<value>] <name> is the corresponding spork name, or ‘show’ to show all current spork settings, active to show which sporks are active<value> is a epoch datetime to enable or disable spork. Requires wallet passphrase to be set with `walletpassphrase` call.

voteraw <masternode-tx-hash> <masternode-tx-index> <governance-hash> <vote-signal> [yes|no|abstain] <time> <vote-sig>
Compile and relay a governance vote with provided external signature instead of signing vote internally.

Generating

generate numblocks Mine blocks immediately (before the RPC call returns).

getgenerate Return if the server is set to generate coins or not. The default is false. It is set with the command line argument `-gen` (or `gobyte.conf` setting `gen`). It can also be set with the `setgenerate` call.

setgenerate generate (genproclimit) Set ‘generate’ true or false to turn generation on or off. Generation is limited to ‘genproclimit’ processors, -1 is unlimited. See the `getgenerate` call for the current setting.

Mining

getblocktemplate (“jsonrequestobject”) If the request parameters include a ‘mode’ key, that is used to explicitly select between the default ‘template’ request or a ‘proposal’. It returns data needed to construct a block to work on.

getmininginfo Returns a json object containing mining-related information.

getnetworkhashps (blocks height) Returns the estimated network hashes per second based on the last n blocks. Pass in [blocks] to override # of blocks, -1 specifies since last difficulty change. Pass in [height] to estimate the network speed at the time when a certain block was found.

prioritisetransaction <txid> <priority delta> <fee delta> Accepts the transaction into mined blocks at a higher (or lower) priority.

submitblock “hexdata” (“jsonparametersobject”) Attempts to submit new block to network. The ‘jsonparametersobject’ parameter is currently ignored. See https://en.bitcoin.it/wiki/BIP_0022 for full specification.

Network

addnode “node” “add|remove|onetry” Attempts add or remove a node from the addnode list. Or try a connection to a node once.

clearbanned Clear all banned IPs.

disconnectnode “node” Immediately disconnects from the specified node.

getaddednodeinfo dummy (“node”) Returns information about the given added node, or all added nodes (note that onetry addnodes are not listed here).

getconnectioncount Returns the number of connections to other nodes.

getnettotals Returns information about network traffic, including bytes in, bytes out, and current time.

getnetworkinfo Returns an object containing various state info regarding P2P networking.

getpeerinfo Returns data about each connected network node as a json array of objects.

listbanned List all banned IPs/Subnets.

ping Requests that a ping be sent to all other nodes, to measure ping time. Results provided in `getpeerinfo`, `pingtime` and `pingwait` fields are decimal seconds. Ping command is handled in queue with all other commands, so it measures processing backlog, not just network ping.

setban “ip(/netmask)” “add/remove” (bantime) (absolute) Attempts add or remove a IP/Subnet from the banned list.

setnetworkactive true/false Disable/enable all p2p network activity.

Rawtransactions

createrawtransaction [{"txid": "id", "vout": n}, ...] [{"address": amount, "data": "hex"}, ...] (locktime) Create a transaction spending the given inputs and creating new outputs. Outputs can be addresses or data. Returns hex-encoded raw transaction. Note that the transaction's inputs are not signed, and it is not stored in the wallet or transmitted to the network.

decoderawtransaction “hexstring” Return a JSON object representing the serialized, hex-encoded transaction.

decodescript “hex” Decode a hex-encoded script.

fundrawtransaction “hexstring” includeWatching Add inputs to a transaction until it has enough in value to meet its out value. This will not modify existing inputs, and will add one change output to the outputs.

getrawtransaction “txid” (verbose) Return the raw transaction data. If verbose=0, returns a string that is serialized, hex-encoded data for 'txid'. If verbose is non-zero, returns an Object with information about 'txid'.

sendrawtransaction “hexstring” (allowhighfees instantsend) Submits raw transaction (serialized, hex-encoded) to local node and network. Also see `createrawtransaction` and `signrawtransaction` calls.

signrawtransaction “hexstring” ([{"txid": "id", "vout": n, "scriptPubKey": "hex", "redeemScript": "hex"}], ...] [“privatekey1”, ...] Sign inputs for raw transaction (serialized, hex-encoded). The second optional argument (may be null) is an array of previous transaction outputs that this transaction depends on but may not yet be in the block chain. The third optional argument (may be null) is an array of base58-encoded private keys that, if given, will be the only keys used to sign the transaction.

Util

createmultisig nrequired [“key”, ...] Creates a multi-signature address with n signature of m keys required. It returns a json object with the address and `redeemScript`.

estimatefee nblocks Estimates the approximate fee per kilobyte needed for a transaction to begin confirmation within nblocks blocks.

estimatepriority nblocks Estimates the approximate priority a zero-fee transaction needs to begin confirmation within nblocks blocks.

estimatesmartfee nblocks WARNING: This interface is unstable and may disappear or change! Estimates the approximate fee per kilobyte needed for a transaction to begin confirmation within nblocks blocks if possible and return the number of blocks for which the estimate is valid.

estimatesmartpriority nblocks WARNING: This interface is unstable and may disappear or change! Estimates the approximate priority a zero-fee transaction needs to begin confirmation within nblocks blocks if possible and return the number of blocks for which the estimate is valid.

validateaddress “gobyteaddress” Return information about the given gobyte address.

verifymessage “gobyteaddress” “signature” “message” Verify a signed message.

Wallet

abandontransaction “txid” Mark in-wallet transaction <txid> as abandoned. This will mark this transaction and all its in-wallet descendants as abandoned which will allow for their inputs to be respend.

addmultisigaddress nrequired [“key”,...] (“account”) Add a nrequired-to-sign multisignature address to the wallet. Each key is a GoByte address or hex-encoded public key. If ‘account’ is specified (DEPRECATED), assign address to that account.

backupwallet “destination” Safely copies wallet.dat to destination, which can be a directory or a path with filename.

dumphdinfo Returns an object containing sensitive private info about this HD wallet.

dumpprivkey “gobyteaddress” Reveals the private key corresponding to ‘gobyteaddress’. Then the importprivkey can be used with this output

dumpwallet “filename” Dumps all wallet keys in a human-readable format.

encryptwallet “passphrase” Encrypts the wallet with ‘passphrase’. This is for first time encryption. After this, any calls that interact with private keys such as sending or signing will require the passphrase to be set prior the making these calls. Use the walletpassphrase call for this, and then walletlock call. If the wallet is already encrypted, use the walletpassphrasechange call. Note that this will shutdown the server.

getaccount “gobyteaddress” DEPRECATED. Returns the account associated with the given address.

getaccountaddress “account” DEPRECATED. Returns the current GoByte address for receiving payments to this account.

getaddressesbyaccount “account” DEPRECATED. Returns the list of addresses for the given account.

getbalance (“account” minconf addlockconf includeWatchonly) If account is not specified, returns the server’s total available balance. If account is specified (DEPRECATED), returns the balance in the account. Note that the account “” is not the same as leaving the parameter out. The server total may be different to the balance in the default “” account.

getnewaddress (“account”) Returns a new GoByte address for receiving payments. If ‘account’ is specified (DEPRECATED), it is added to the address book so payments received with the address will be credited to ‘account’.

getrawchangeaddress Returns a new GoByte address, for receiving change. This is for use with raw transactions, NOT normal use.

getreceivedbyaccount “account” (minconf addlockconf) DEPRECATED. Returns the total amount received by addresses with <account> in transactions with specified minimum number of confirmations.

getreceivedbyaddress “gobyteaddress” (minconf addlockconf) Returns the total amount received by the given gobyteaddress in transactions with specified minimum number of confirmations.

gettransaction “txid” (includeWatchonly) Get detailed information about in-wallet transaction <txid>

getunconfirmedbalance Returns the server’s total unconfirmed balance.

getwalletinfo Returns an object containing various wallet state info.

importaddress “address” (“label” rescan p2sh) Adds a script (in hex) or address that can be watched as if it were in your wallet but cannot be used to spend.

importelectrumwallet “filename” index Imports keys from an Electrum wallet export file (.csv or .json)

importprivkey “gobyteprivkey” (“label” rescan) Adds a private key (as returned by dumpprivkey) to your wallet.

importpubkey “pubkey” (“label” rescan) Adds a public key (in hex) that can be watched as if it were in your wallet but cannot be used to spend.

importwallet “filename” Imports keys from a wallet dump file (see dumpwallet).

instantsendtoaddress “gobyteaddress” amount (“comment” “comment-to” subtractfeefromamount) Send an amount to a given address. The amount is a real and is rounded to the nearest 0.00000001

keepass <genkeyinit|setpassphrase> Keepass settings.

keypoolrefill (newsize) Fills the keypool.

listaccounts (minconf addlockconf includeWatchonly) DEPRECATED. Returns Object that has account names as keys, account balances as values.

listaddressgroupings Lists groups of addresses which have had their common ownership made public by common use as inputs or as the resulting change in past transactions.

listlockunspent Returns list of temporarily unspendable outputs. See the lockunspent call to lock and unlock transactions for spending.

listreceivedbyaccount (minconf addlockconf includeempty includeWatchonly) DEPRECATED. List balances by account.

listreceivedbyaddress (minconf addlockconf includeempty includeWatchonly) List balances by receiving address.

listsinceblock (“blockhash” target-confirmations includeWatchonly) Get all transactions in blocks since block [blockhash], or all transactions if omitted

listtransactions (“account” count from includeWatchonly) Returns up to ‘count’ most recent transactions skipping the first ‘from’ transactions for account ‘account’.

listunspent (minconf maxconf [“address”,...]) Returns array of unspent transaction outputs with between minconf and maxconf (inclusive) confirmations. Optionally filter to only include txouts paid to specified addresses.

lockunspent unlock [{“txid”:”txid”,”vout”:n},...] Updates list of temporarily unspendable outputs. Temporarily lock (unlock=false) or unlock (unlock=true) specified transaction outputs.

move “fromaccount” “toaccount” amount (minconf “comment”) DEPRECATED. Move a specified amount from one account in your wallet to another.

sendfrom “fromaccount” “togobyteaddress” amount (minconf addlockconf “comment” “comment-to”) DEPRECATED (use sendtoaddress). Sent an amount from an account to a gobyte address.

sendmany “fromaccount” {“address”:amount,...} (minconf addlockconf “comment” [“address”,...] subtractfeefromamount u
Send multiple times. Amounts are double-precision floating point numbers.

sendtoaddress “gobyteaddress” amount (“comment” “comment-to” subtractfeefromamount use_is use_ps)
Send an amount to a given address.

setaccount “gobyteaddress” “account” DEPRECATED. Sets the account associated with the given address.

settxfee amount Set the transaction fee per kB. Overwrites the paytxfee parameter.

signmessage “gobyteaddress” “message” Sign a message with the private key of an address.

walletlock Removes the wallet encryption key from memory, locking the wallet. After calling this method, you will need to call walletpassphrase again before being able to call any methods which require the wallet to be unlocked.

walletpassphrase “passphrase” timeout (mixingonly) Stores the wallet decryption key in memory for ‘timeout’ seconds. This is needed prior to performing transactions related to private keys such as sending gobytes

walletpassphrasechange “oldpassphrase” “newpassphrase” Changes the wallet passphrase from ‘oldpassphrase’ to ‘newpassphrase’.

Advanced topics

Coin Control

Coin Control allows users of the GoByte Core Wallet to specify which addresses and Unspent Transaction Outputs (UTXOs) should be used as inputs in transactions. This allows you to keep a specific balance on certain addresses in your wallet, while spending others freely. In GoByte Core Wallet, click **Settings > Options > Wallet > Enable coin control features**. Now, when you go to the Send tab in your wallet, a new button labelled **Inputs...** will appear. Click this button to select which UTXOs can be used as input for any transactions you create. The following window appears:

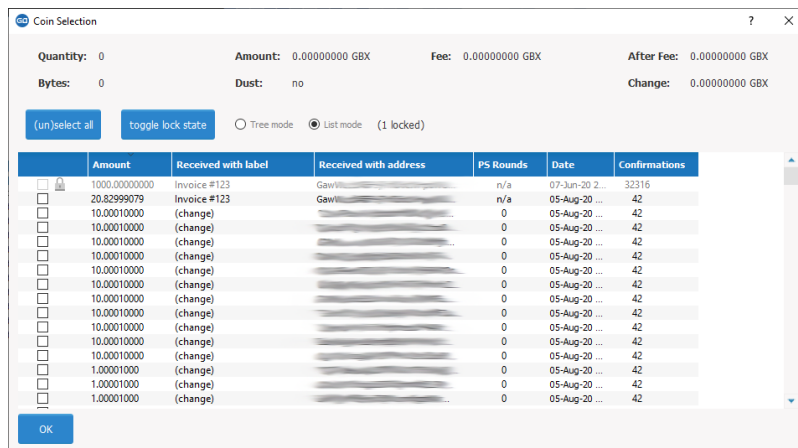
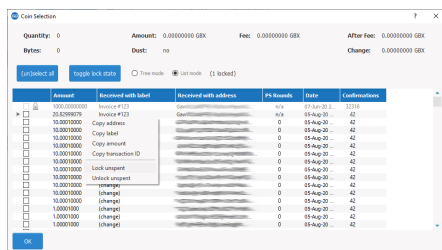


Fig. 82: Coin Selection window in GoByte Core wallet, showing two masternodes (testnet)

Right click on the transaction(s) you do not want to spend, then select **Lock unspent**. A small lock will appear next to the transaction. You can click the **Toggle lock state** button to invert the locked/unlocked state of all UTXOs. When you are ready to continue, click **OK**. You can now safely create transactions with your remaining funds without affecting the locked UTXOs.



HD Wallets

Since version 0.12.2.0, GoByte Core has included an implementation of BIP39/BIP44 compatible hierarchical deterministic (HD) key generation. This functionality is only available from the command line by specifying the `usehd` option when starting GoByte Core for the first time. Use this function with care, since the mnemonic seed and keys

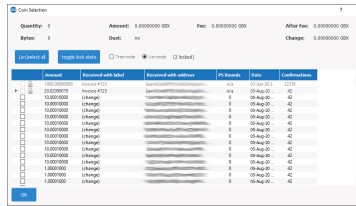


Fig. 83: Locking UTXOs in GoByte Core wallet

will be stored in plain text until you specify a wallet passphrase. Note that the wallet passphrase is different to the mnemonic passphrase, which is often also referred to as the “25th word” or “extension word”. The wallet passphrase encrypts the wallet file itself, while the mnemonic passphrase is used to specify different derivation branches from the same mnemonic seed.

We will use the Windows GUI wallet in this example, but the commands are similar if using `gobyte-qt` or `gobyted` on other operating systems. Enter the following command to get started with a randomly generated HD wallet seed and no mnemonic passphrase:

```
gobyte-qt.exe --usehd=1
```

A new HD wallet will be generated and GoByte Core will display a warning informing you that you must encrypt your wallet after verifying it works correctly. Open the console from **Tools -> Debug console** or issue the following RPC command from `gobyte-cli` to view the mnemonic seed:

```
dumphdinfo
```

GoByte Core will display the HD seed in both hexadecimal and as a BIP39 mnemonic. To restore an existing HD wallet, or define your own separately generated mnemonic and/or passphrase, ensure no `wallet.dat` file exists in the `datadir` and enter the following command:

```
gobyte-qt.exe --usehd=1 --mnemonic="enter mnemonic" --mnemonicpassphrase="optional_
↵mnemonic passphrase"
```

The HD wallet will be restored and your balance will appear once sync is complete.

Multisignature

This section presents a worked example to demonstrate multisig functionality in GoByte Core. While the transactions are no longer visible on the current testnet blockchain and some address formats or RPC responses may differ slightly from the version shown here, the principle and commands are the same. The example demonstrates how to set up a 2-of-3 multisig address and create a transaction. The example parties involved are a buyer, a seller and an arbiter. This example is based on:

- <https://people.xiph.org/~greg/escrowexample.txt>
- <https://gist.github.com/gavinandresen/3966071>
- <https://bitcoin.org/en/developer-examples#p2sh-multisig>

Step 1: Create three addresses

Seller:

```

seller@testnet03:~$ ./gobyte-cli getnewaddress
n18cPEtj4ZfToPZxRszUz2XPts4eGsxiPk
seller@testnet03:~$ ./gobyte-cli validateaddress n18cPEtj4ZfToPZxRszUz2XPts4eGsxiPk
{
  "isvalid" : true,
  "address" : "n18cPEtj4ZfToPZxRszUz2XPts4eGsxiPk",
  "ismine" : true,
  "isscript" : false,
  "pubkey" : "02a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e",
  "iscompressed" : true,
  "account" : ""
}
seller@testnet03:~$ ./gobyte-cli dumpprivkey n18cPEtj4ZfToPZxRszUz2XPts4eGsxiPk
cVQVgBr8sW4FTPyZ16BSCo1PcAfDhpJArgMPdLxKZQWcVFwMXXx

```

Buyer:

```

buyer@testnet03:~$ ./gobyte-cli getnewaddress
mp5orHuaFaHXCFSceYvUPL7H16JU8fKG6u
buyer@testnet03:~$ ./gobyte-cli validateaddress mp5orHuaFaHXCFSceYvUPL7H16JU8fKG6u
{
  "isvalid" : true,
  "address" : "mp5orHuaFaHXCFSceYvUPL7H16JU8fKG6u",
  "ismine" : true,
  "isscript" : false,
  "pubkey" : "0315617694c9d93f0ce92769e050a6868ffc74d229077379c0af8bfb193c3d351c",
  "iscompressed" : true,
  "account" : ""
}
buyer@testnet03:~$ ./gobyte-cli dumpprivkey mp5orHuaFaHXCFSceYvUPL7H16JU8fKG6u
cP9DFmEDb11waWbQ8eG1YUoZCGe59BBxJF3kk95PTMXuG9HzcxnU

```

Arbiter:

```

arbiter@testnet03:~$ ./gobyte-cli getnewaddress
n1cZSyQXhach5rrj2tm5wg6JC7uZ3qPNiN
arbiter@testnet03:~$ ./gobyte-cli validateaddress n1cZSyQXhach5rrj2tm5wg6JC7uZ3qPNiN
{
  "isvalid" : true,
  "address" : "n1cZSyQXhach5rrj2tm5wg6JC7uZ3qPNiN",
  "ismine" : true,
  "isscript" : false,
  "pubkey" : "0287ce6cf69b85593ce7db801874c9a2fb1b653dbe5dd9ebfa73e98b710af9e9ce",
  "iscompressed" : true,
  "account" : ""
}
arbiter@testnet03:~$ ./gobyte-cli dumpprivkey n1cZSyQXhach5rrj2tm5wg6JC7uZ3qPNiN
cUbDFL81a2w6urAGZf7ecGbdzM82pdHLeCaPXdp71s96SzDV49M

```

This results in three keypairs (public/private):

```

seller:      02a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e  /_
↪ cVQVgBr8sW4FTPyZ16BSCo1PcAfDhpJArgMPdLxKZQWcVFwMXXx
buyer:      0315617694c9d93f0ce92769e050a6868ffc74d229077379c0af8bfb193c3d351c  /_
↪ cP9DFmEDb11waWbQ8eG1YUoZCGe59BBxJF3kk95PTMXuG9HzcxnU
arbiter:    0287ce6cf69b85593ce7db801874c9a2fb1b653dbe5dd9ebfa73e98b710af9e9ce  /_
↪ cUbDFL81a2w6urAGZf7ecGbdzM82pdHLeCaPXdp71s96SzDV49M

```


Step 2: Create multisig address

The `createmultisig` command takes as variables the number `n` signatures of `m` keys (supplied as json array) required. In this example, 2 of 3 keys are required to sign the transaction.

Note: The address can be created by anyone, as long as the public keys and their sequence are known (resulting address and `redeemScript` are identical, see below).

Seller:

```
seller@testnet03:~$ ./gobyte-cli createmultisig 2 '[
↪ "02a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e",
↪ "0315617694c9d93f0ce92769e050a6868ffc74d229077379c0af8bfb193c3d351c",
↪ "0287ce6cf69b85593ce7db801874c9a2fb1b653dbe5dd9ebfa73e98b710af9e9ce"]'
{
  "address" : "2MuEQCZh7VB8pNrT4bj1CFZQh2oK7XZYLQf",
  "redeemScript" :
↪ "522102a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e210315617694c9d93f0ce92769e
↪ "
}
```

Buyer:

```
buyer@testnet03:~$ ./gobyte-cli createmultisig 2 '[
↪ "02a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e",
↪ "0315617694c9d93f0ce92769e050a6868ffc74d229077379c0af8bfb193c3d351c",
↪ "0287ce6cf69b85593ce7db801874c9a2fb1b653dbe5dd9ebfa73e98b710af9e9ce"]'
{
  "address" : "2MuEQCZh7VB8pNrT4bj1CFZQh2oK7XZYLQf",
  "redeemScript" :
↪ "522102a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e210315617694c9d93f0ce92769e
↪ "
}
```

Arbiter:

```
arbiter@testnet03:~$ ./gobyte-cli createmultisig 2 '[
↪ "02a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e",
↪ "0315617694c9d93f0ce92769e050a6868ffc74d229077379c0af8bfb193c3d351c",
↪ "0287ce6cf69b85593ce7db801874c9a2fb1b653dbe5dd9ebfa73e98b710af9e9ce"]'
{
  "address" : "2MuEQCZh7VB8pNrT4bj1CFZQh2oK7XZYLQf",
  "redeemScript" :
↪ "522102a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e210315617694c9d93f0ce92769e
↪ "
}
```

Step 3: Buyer funds the multisig address

This works the same as a usual transaction.

Buyer:

```
buyer@testnet03:~$ ./gobyte-cli sendtoaddress 2MuEQCZh7VB8pNrT4bj1CFZQh2oK7XZYLQf 777.
↪ 77
a8b3bf5bcace91a8dbbdfb9b7eb027efb9bd001792f043ecf7b558aaa3cb951
```

The seller/arbiter can trace the transaction by its txid in the block explorer. Or from the console as follows.

Buyer:

```
seller@testnet03:~$ ./gobyte-cli getrawtransaction
↪ a8b3bf5bcace91a8dbbdf9b7eb027efb9bd001792f043ecf7b558aaa3cb951 1
{
  "hex" : "010000001a2e514dd90f666e3de4cddd22682ae1ca7225988656369d98228c742482fee16b010000006b4830",
  "txid" : "a8b3bf5bcace91a8dbbdf9b7eb027efb9bd001792f043ecf7b558aaa3cb951",
  "version" : 1,
  "locktime" : 0,
  [...]
  "vout" : [
    {
      "value" : 777.77000000,
      "n" : 0,
      "scriptPubKey" : {
        "asm" : "OP_HASH160 15c85c2472f5941b60a49462a2cfd0d17ab49d1c OP_EQUAL",
        ↪ ",
        "hex" : "a91415c85c2472f5941b60a49462a2cfd0d17ab49d1c87",
        "reqSigs" : 1,
        "type" : "scripthash",
        "addresses" : [
          "2MuEQCZh7VB8pNrT4bj1CFZQh2oK7XZYLQf"
        ]
      }
    },
    [...]
  ],
  "blockhash" : "000000034def806f348cadf6a80660aed1cfc30ccbd1492a8ea87062800ea94d",
  "confirmations" : 3,
  "time" : 1409224896,
  "blocktime" : 1409224896
}
```

Step 4: Spending the multisig

Now we assume the deal is complete, the buyer got the goods and everyone is happy. Now the seller wants to get his GoByte. As a 2-of-3 multisig was used, the transaction must be signed by 2 parties (seller + buyer or arbiter). The seller creates a transaction (we will reuse his public address from above).

Seller:

```
seller@testnet03:~$ ./gobyte-cli createrawtransaction '[{"txid":
↪ "a8b3bf5bcace91a8dbbdf9b7eb027efb9bd001792f043ecf7b558aaa3cb951", "vout": 0}]' '[{
↪ "n18cPEtj4ZfToPZxRszUz2XPts4eGsxiPk": 777.77}]'
010000000151b93caa8a557bcf3e042f7901d09bfb7e02ebb7f9dbbdfba891ceca5bbfb3a80000000000ffffffffff0140d6de
```

And partially signs it, using the redeemScript, scriptPubKey and his private key

Seller:

```
seller@testnet03:~$ ./gobyte-cli signrawtransaction
↪ '010000000151b93caa8a557bcf3e042f7901d09bfb7e02ebb7f9dbbdfba891ceca5bbfb3a8000000000ffffffffff0140d6de
↪ ' [{"txid": "a8b3bf5bcace91a8dbbdf9b7eb027efb9bd001792f043ecf7b558aaa3cb951", "vout
↪ ": 0, "scriptPubKey": "a91415c85c2472f5941b60a49462a2cfd0d17ab49d1c87", "redeemScript":
↪ "522102a862b412ff9e3afd01a2873a02622897f6df92e3fc85597788b898309fec882e210315617694c9d93f0ce92769e
↪ }]' [{"cVQVgBr8sW4FTPyZl6BSColPcAfDhpJARgMPdLxKZQWcVFwMXRXx"}]
```

(continues on next page)

Daemon

GoByte can be run as a background process (or daemon) on Linux systems. This is particularly useful if you are running GoByte as a server instead of as a GUI node. This guide assumes you have installed GoByte Core for Linux as described in the [Linux Installation Guide](#).

1. Create a user and group to run the daemon:

```
sudo useradd -m gobyte -s /bin/bash
```

2. Create a data directory for GoByte in the new user's home directory:

```
sudo -u gobyte mkdir -p /home/gobyte/.gobytecore
```

3. Create a configuration file in the new GoByte data directory:

```
sudo -u gobyte nano /home/gobyte/.gobytecore/gobyte.conf
```

4. Paste the following basic configuration to your `gobyte.conf` file, replacing the password with a long and random password:

```
listen=1
server=1
daemon=1
```

5. Register the `gobyted` daemon as a system service by creating the following file:

```
sudo nano /etc/systemd/system/gobyted.service
```

6. Paste the following daemon configuration into the file:

```
[Unit]
Description=GoByte Core Daemon
After=syslog.target network-online.target

[Service]
Type=forking
User=gobyte
Group=gobyte
OOMScoreAdjust=-1000
ExecStart=/usr/local/bin/gobyted -pid=/home/gobyte/.gobytecore/gobyted.pid
TimeoutStartSec=10m
ExecStop=/usr/local/bin/gobyte-cli stop
TimeoutStopSec=120
Restart=on-failure
RestartSec=120
StartLimitInterval=300
StartLimitBurst=3

[Install]
WantedBy=multi-user.target
```

7. Register and start the daemon with `systemd`:

```
sudo systemctl daemon-reload
sudo systemctl enable gobyted
sudo systemctl start gobyted
```

GoByte is now installed as a system daemon. View the status as follows:

```
systemctl status gobyted
```

View logs as follows:

```
sudo journalctl -u gobyted
```

Tor

Tor is free and open-source software for enabling anonymous communication. The name derived from the acronym for the original software project name “The Onion Router”. Tor directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays to conceal a user’s location and usage from anyone conducting network surveillance or traffic analysis.

GoByte Core GUI

GoByte Core traffic can be directed to pass through Tor by specifying a running Tor service as a proxy. First install Tor by visiting <https://www.torproject.org/download/> and downloading the appropriate Tor Browser bundle for your system. Set up the Tor browser by following the documentation on [Installation](#) and [Running Tor Browser for the First Time](#).

Once Tor Browser is running, you have two options to configure GoByte Core to use Tor for network traffic.

1. **Using the GUI:** Start GoByte Core and go to **Settings > Options > Network** and enable the **Connect through SOCKS5 proxy** setting. Specify 127.0.0.1 for the **Proxy IP** and 9150 for the **Port**. Click **OK** and restart GoByte Core.
2. **Using gobyte.conf:** Ensure GoByte Core is not running and edit your `gobyte.conf` settings file. Add the line `proxy=127.0.0.1:9150`, save the file and start GoByte Core.

You are now connected through the Tor network. You will need to remember to start the Tor Browser each time before you start GoByte Core or you will not be able to sync.

Tor onion service

Tor onion services allows other users to connect to your GoByte node using an onion address, providing further anonymity by concealing your IP address. Follow these steps to set up an onion service under Ubuntu Linux:

1. Install tor:

```
sudo apt install tor
```

2. Add the following line to the `torrc` file:

```
sudo bash -c "echo -e 'ControlPort 9051\nCookieAuthentication_\n1\nCookieAuthFileGroupReadable 1' >> /etc/tor/torrc"
```

3. Restart Tor:

```
sudo systemctl restart tor
```

4. Determine the group Tor is running under (usually the last entry in your groups file):

```
tail /etc/group
```

The group is usually `debian-tor` under Debian-based Linux distributions.

5. Add the user running GoByte to the Tor group:

```
sudo usermod -aG debian-tor gobyte
```

6. Add the following two lines to `gobyte.conf`:

```
proxy=127.0.0.1:9050
torcontrol=127.0.0.1:9051
```

7. Restart GoByte and monitor `debug.log` for onion informatoin:

```
grep -i onion ~/.gobytecore/debug.log
```

You should see a line similar to the following:

```
2020-06-29 03:43:57 tor: Got service ID knup3fvr6fyvypu7, advertising service_
↪knup3fvr6fyvypu7.onion:19999
```

Your onion service is now available at the shown address.

Multiple wallets

It is possible to select between different GoByte wallets when starting GoByte Core by specifying the `wallet` argument, or even run multiple instances of GoByte Core simultaneously by specifying separate data directories using the `datadir` argument.

To begin, install the GoByte Core wallet for your system according to the [installation instructions](#). When you get to the step **Running GoByte Core for the first time**, you can decide whether you want to maintain separate `wallet.dat` files in the default location (simpler if you do not need to run the wallets simultaneously), or specify entirely separate data directories such as e.g. `C:\GoByte1` (simpler if you do want to run the wallets simultaneously).

Separate wallet.dat files

For this scenario, we will create two shortcuts on the desktop, each using a different wallet file. Navigate to the binary file used to start GoByte Core (typically `locatd` at `C:\Program Files\GoByteCore\gobyte-qt.exe` or similar) and create two shortcuts on the desktop. Then open the **Properties** window for each of these shortcuts.

Modify the **Target** property of each shortcut to point to a different wallet file by specifying the `wallet` argument when starting the wallet. If you do not specify a `wallet` argument, `wallet.dat` will be used by default. The specified wallet file will be created if it does not exist. The following example demonstrates two wallets named `workwallet.dat` and `homewallet.dat`:

- Wallet Target 1: `"C:\Program Files\GoByteCore\gobyte-qt.exe" -wallet=workwallet.dat`
- Wallet Target 2: `"C:\Program Files\GoByteCore\gobyte-qt.exe" -wallet=homewallet.dat`

You can now use the two icons to quickly and easily open different wallets from your desktop. Note that you cannot open both wallets simultaneously. To do this, you will need two separate data directories, as described below.

The image displays two side-by-side screenshots of the 'GoByte1 Properties' and 'GoByte 2 Properties' dialog boxes. Both dialogs have a 'General' tab selected, showing fields for 'Target type', 'Target location', 'Target', 'Start in', 'Shortcut key', 'Run', and 'Comment'. The 'Target' field in both is highlighted with a red rectangle. In the 'GoByte1' dialog, the target is 'Windows64\gobyte-qt.exe -wallet=workwallet.dat'. In the 'GoByte 2' dialog, the target is 'Windows64\gobyte-qt.exe -wallet=homewallet.dat'. The 'GoByte 2' dialog also features 'OK', 'Cancel', and 'Apply' buttons at the bottom.

1.6. Wallets

Separate data directories

Start GoByte Core and allow it to synchronize with the network, then close GoByte Core again. You can now create two directories at e.g. `C:\GoByte1` and `C:\GoByte2` and copy the `blocks` and `chainstate` directories from the synchronized data directory into the new directories. Each of these will serve as a separate data directory, allowing you to run two instances of GoByte Core simultaneously. Create two (or more) shortcuts on your desktop as described above, then specify arguments for `datadir` as shown below:

- **Datadir** **Target** **1:** `"C:\Program Files\GoByteCore\gobyte-qt.exe"`
 `-datadir=C:\GoByte1 -listen=0`
- **Datadir** **Target** **2:** `"C:\Program Files\GoByteCore\gobyte-qt.exe"`
 `-datadir=C:\GoByte2 -listen=0`

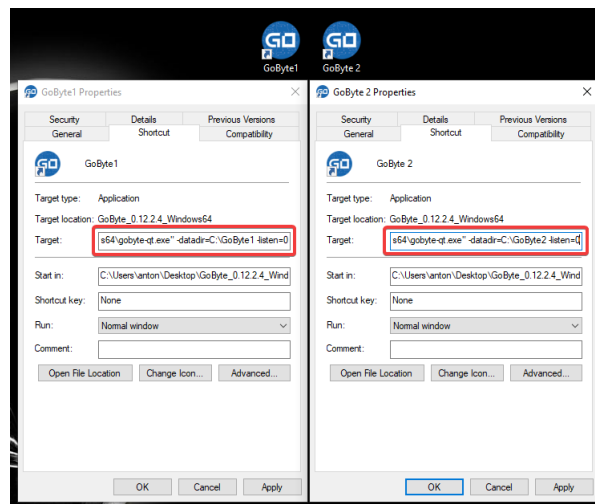


Fig. 86: Specifying separate datadirs

You can now use the two icons to quickly and easily open different wallets simultaneously from your desktop. Both wallets maintain separate and full copies of the blockchain, which may use a lot of drive space. For more efficient use of drive space, consider using an SPV or “light” wallet such as *GoByte Electrum* to maintain multiple separate wallets without keeping a full copy of the blockchain.

KeePass

Since version 0.11.0, GoByte Core has supported integration with KeePass, the popular open source password manager. This guide describes how to configure the association between GoByte Core and KeePass, and how to save a GoByte Core wallet passphrase in KeePass using the integration. When this is done, KeePass can be used to unlock the wallet.

Installation

You will need the following:

- KeePass 2: <http://keepass.info>
- KeePassHttp plugin: <https://github.com/pfn/keepasshttp>
- GoByte Core: <https://www.gobyte.network>

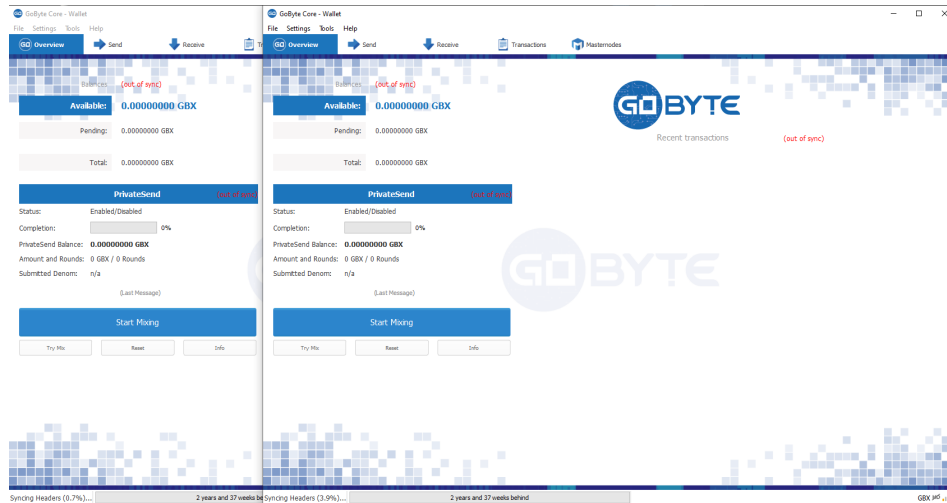


Fig. 87: Two instances of GoByte Core running simultaneously

If not already installed, install these packages according to the instructions linked below:

- KeePass: <https://keepass.info/help/v2/setup.html>
- KeePassHttp: <https://github.com/pfn/keepasshttp/blob/master/README.md>
- GoByte Core: <https://docs.gobyte.network/en/stable/wallets/gobytecore/installation.html>

Commands

The following KeePass RPC commands are available in the GoByte Core client console or server:

keepass genkey Generates a base64 encoded 256 bit AES key that can be used for communication with KeePassHttp. This is only necessary for manual configuration. Use init for automatic configuration.

keepass init Sets up the association between GoByte and KeePass by generating an AES key and sending an association message to KeePassHttp. This will trigger KeePass to ask for an ID for the association. Returns the association and the base64 encoded string for the AES key.

keepass setpassphrase Updates the passphrase in KeePassHttp to a new value. This should match the passphrase you intend to use for the wallet. Please note that the standard RPC commands `walletpassphrasechange` and the wallet encryption from the QT GUI already send the updates to KeePassHttp, so this is only necessary for manual manipulation of the password.

The following new arguments are available for `gobytd` and `gobyte-qt`:

keepass Use KeePass 2 integration using KeePassHttp plugin (default: 0)

keepassport=<port> Connect to KeePassHttp on port <port> (default: 19455)

keepasskey=<key> KeePassHttp key for AES encrypted communication with KeePass

keepassid=<name> KeePassHttp id for the established association

keepassname=<name> Name to construct url for KeePass entry that stores the wallet passphrase



Fig. 88: GoByte Core Wallet

1.6.2 GoByte Electrum Wallet

GoByte Electrum is a light wallet which uses powerful external servers to index the blockchain, while still securing the keys on your personal computer. Transactions are verified on the GoByte blockchain using a technique called Secure Payment Verification (SPV), which only requires the block headers and not the full block. This means that wallet startup is almost instant, while still keeping your funds secure and mobile. It does not currently support advanced InstantSend and PrivateSend features.

GoByte Electrum is a fork of the Electrum wallet for Bitcoin. While this documentation focuses on using GoByte Electrum, full documentation of all Bitcoin Electrum features (mostly identical in GoByte Electrum) is available at the [official documentation site](#).

Installation

Download

You can download GoByte Electrum from the official GoByte website or the GoByte Electrum minisite.

- <https://www.gobyte.network/wallets>
- <https://electrum.gobyte.network>

GoByte Electrum is developed by GoByte Core **gobytecoin** and is released through our GitHub account. SHA256 checksums are available in the releases section of the GoByte Electrum repository to verify authenticity.

- <https://github.com/gobytecoin/electrum-gobyte/releases>

Linux

GoByte Electrum for Linux is available from a PPA for Ubuntu and Linux Mint, and as a source tarball for other systems. As of version 3.0.6, it requires Python 3 to run. Enter the following commands to install from PPA:

```
sudo add-apt-repository ppa:gobytecoin/gobyte-electrum
sudo apt update
sudo apt install electrum-gobyte
```

Enter the following commands (changing the version number to match the current version as necessary) in the terminal to install GoByte Electrum from the source tarball:

```
sudo apt install python3-pyqt5 python3-pip python3-setuptools
wget https://github.com/gobytecoin/electrum-gobyte/releases/download/3.0.6.3/Electrum-
↳GBX-3.0.6.3.tar.gz
tar -zxvf Electrum-GBX-3.0.6.3.tar.gz
cd Electrum-GBX-3.0.6.3
sudo python3 setup.py install
```

macOS

Simply download and run the DMG file. You may need to grant permission to install, depending on your security settings. Click through the installation wizard and run GoByte Electrum from your Applications folder when complete.

Windows

Simply download and run the installer file to set up GoByte Electrum. You may need to grant permission to install, depending on your security settings. Click through the installation wizard and run GoByte Electrum from the Start menu when complete.

Android

Download and run the APK file from <https://electrum.gobyte.network> to set up GoByte Electrum. You may need to grant permission to install from unknown sources, depending on your security settings. Click through the installation wizard and run GoByte Electrum when complete.

Creating a New Wallet

GoByte Electrum gathers configuration data when run for the first time. For more on the concepts behind this process, skip to the later sections of this guide discussing backups, security, and addresses. When setting up GoByte Electrum for the first time, a wizard will guide you through the process of creating your first wallet. The first screen asks how you would like to connect to the remote server. Select **Auto connect** and click **Next** to continue. You will see a notice that no wallet currently exists. Enter a name for your wallet (or accept the default name) and click **Next** to create your wallet.

wallets/electrum/img/connect.png

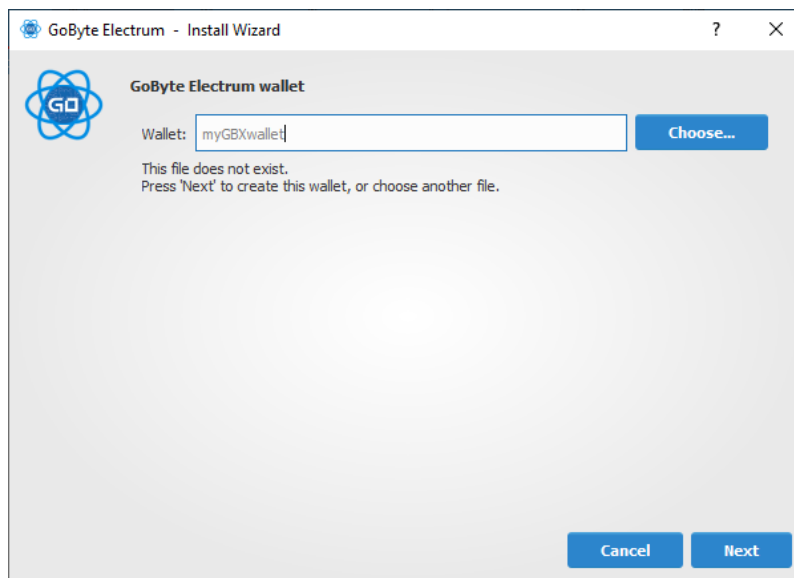


Fig. 89: Selecting the server and naming your first wallet

You will be asked what kind of wallet you want to create. Choose between **Standard wallet**, **Multi-signature wallet** and **Watch GoByte addresses**. If you are unsure, select **Standard wallet** and click **Next** to continue. You will then be asked how you want to store/recover the seed. If stored safely, a seed can be used to restore a lost wallet on another computer. Choose between **Create a new seed**, **I already have a seed**, **Use public or private keys** or **Use a hardware device**. If you are using Electrum GoByte for the first time and not restoring an existing wallet, choose **Create a new seed** and click **Next** to continue.

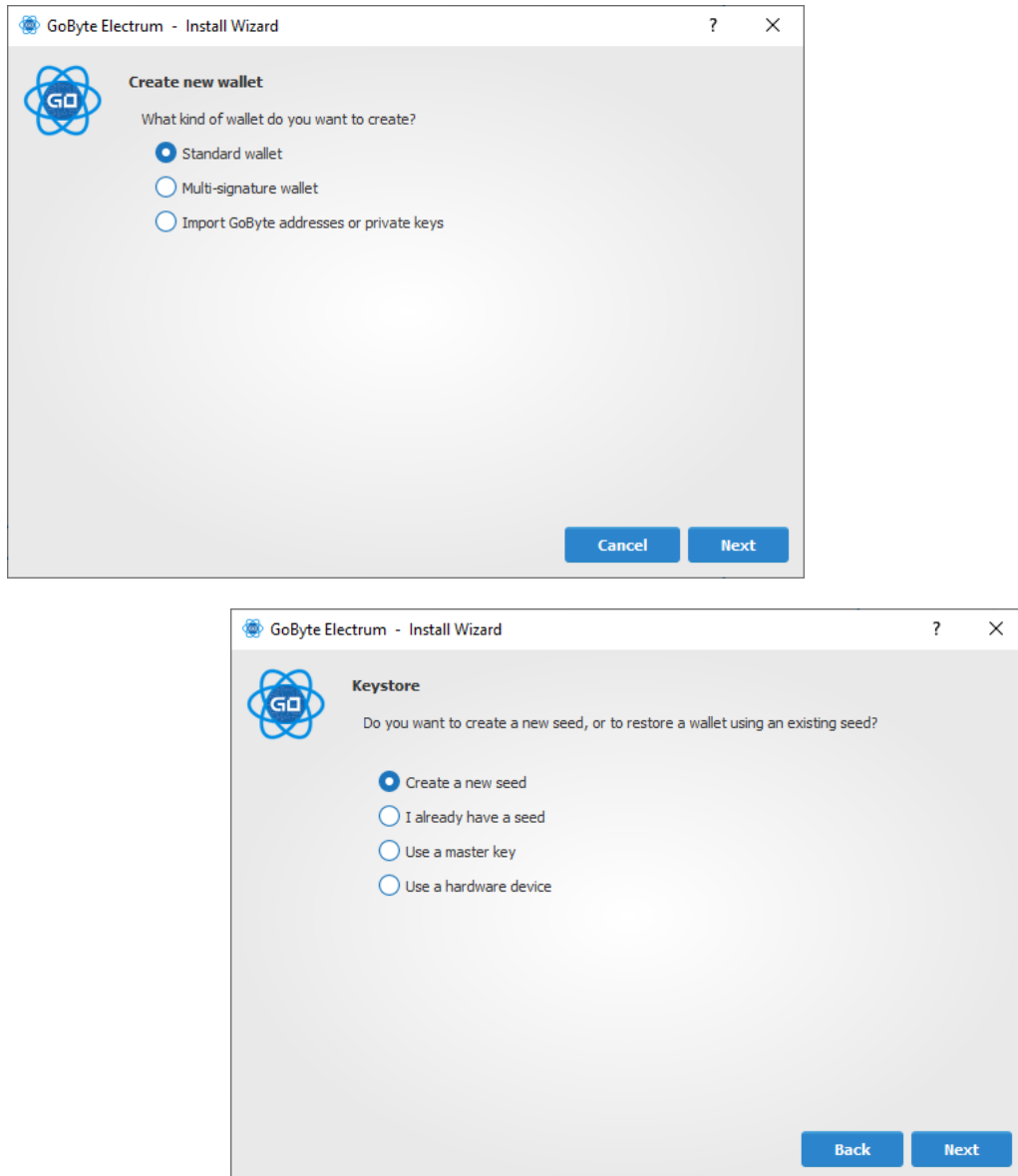


Fig. 90: Selecting the wallet type and keystore

Electrum GoByte will generate your wallet and display the recovery seed. Write this seed down, ideally on paper and not in an electronic format, and store it somewhere safe. This seed is the only way you can recover your wallet if you lose access for any reason. To make sure you have properly saved your seed, Electrum GoByte will ask you to type it in as a confirmation. Type the words in the correct order and click **Next** to continue.

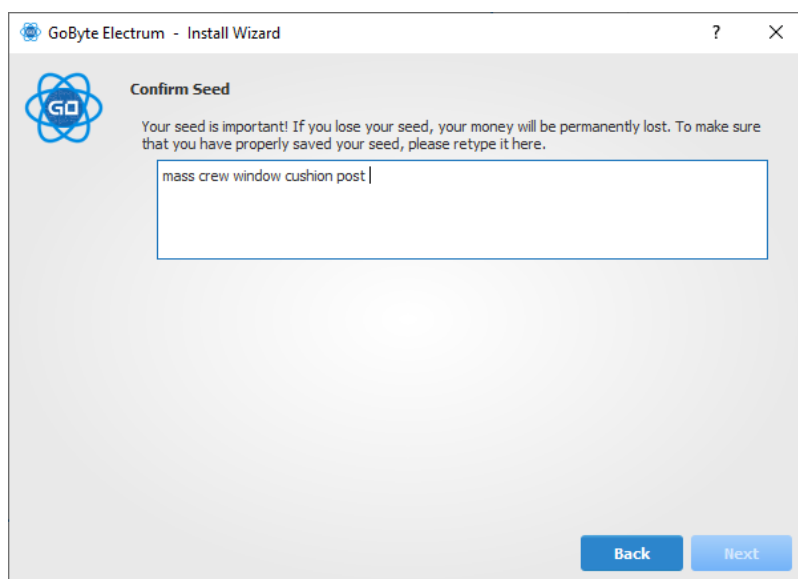
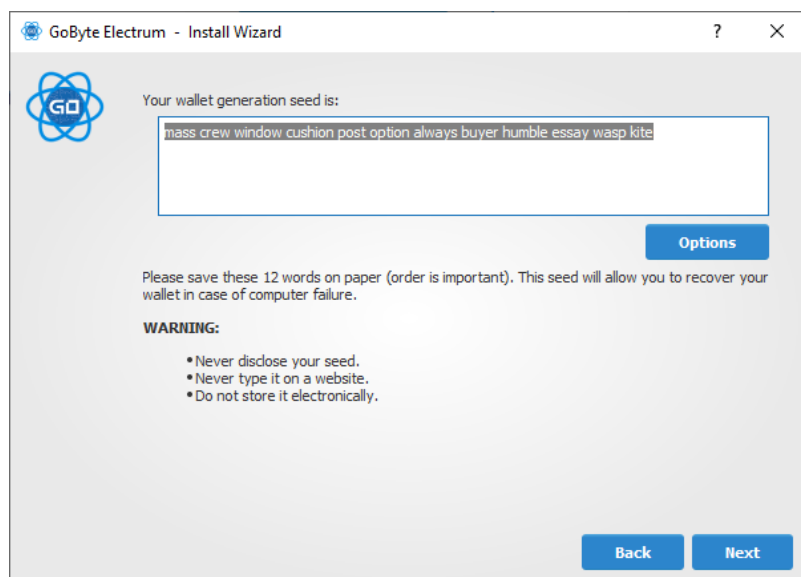


Fig. 91: Generating and confirming the recovery seed

A password optionally secures your wallet against unauthorized access. Adding a memorable, strong password now improves the security of your wallet by encrypting your seed from the beginning. Skipping encryption at this point by not selecting a password risks potential theft of funds later, however unlikely the threat may be. Enter and confirm a password, ensure the **Encrypt wallet file** checkbox is ticked and click **Next** to continue.

Your GoByte Electrum wallet is now set up and ready for use.

Sending and receiving

You may own GoByte stored in another software wallet, or on an exchange such as Bittrex or Kraken, or simply want to send or receive funds as a wage or business transaction. Funds can be transferred between these source and the Electrum wallet using GoByte addresses. Your wallet contains multiple addresses, and will generate new addresses as necessary. Since the GoByte blockchain is transparent to the public, it is considered best practice to use a new address for each transaction in order to maintain your privacy.

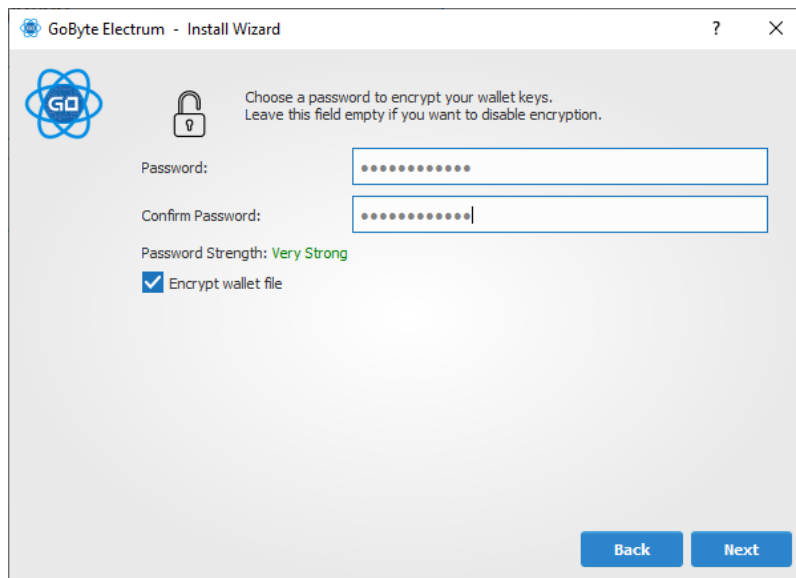


Fig. 92: Entering and confirming a wallet encryption password

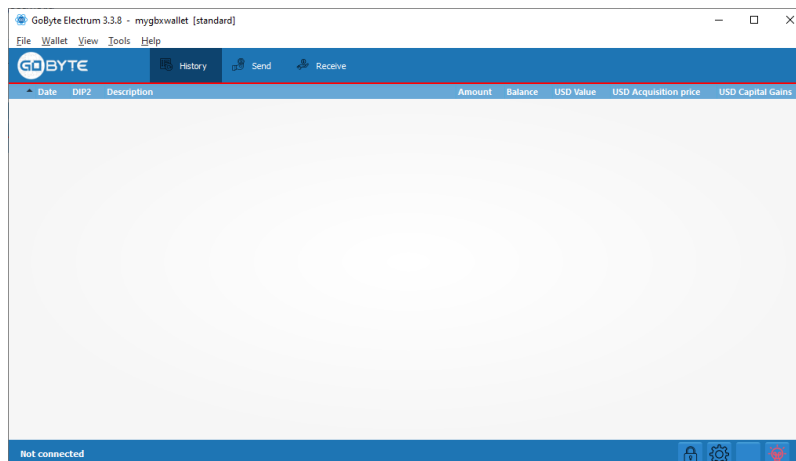


Fig. 93: GoByte Electrum after setup is complete

Sending

Click the **Send** tab to make a payment. Enter the destination address in the **Pay to** field, either manually or by pasting from the clipboard. Optionally enter a **Description** for to appear in your transaction history, followed by the **Amount** to be sent. The total amount of the transaction is the sum of the sent amount and transaction fee, which is calculated automatically. GoByte Electrum issues a warning if the total transaction amount exceeds the wallet balance.

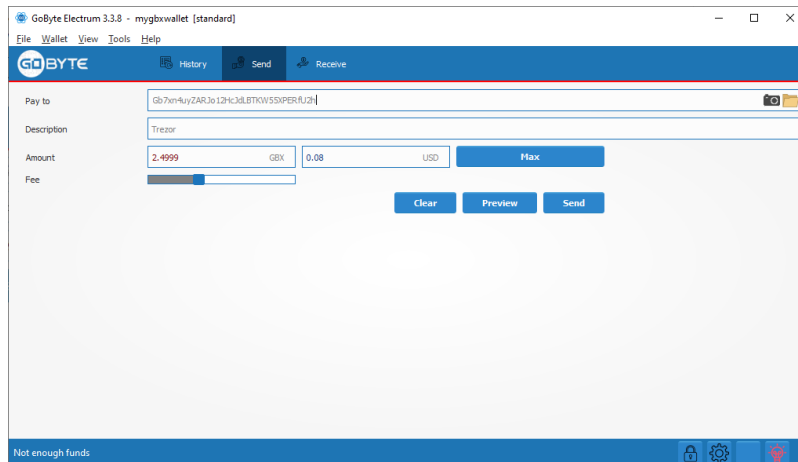


Fig. 94: Transaction ready to send in GoByte Electrum wallet

The wallet will request your password, then broadcast the transaction to the network and display a confirmation dialog with your transaction ID.



wallets/electrum/img/send-password.png

Receiving

You can view your receiving addresses by clicking the **Receive** tab. Double-click the **Receiving address**, then copy it to the clipboard by clicking the Copy to clipboard icon. If you intend to use the address repeatedly, you can also enter a description click Save to store the address in the Requests list. Clicking an address in the list will display the stored information in the top area, together with a QR code containing the same information.

Enter this address in the software sending the funds, send it to the person transferring funds to you or scan it directly from your mobile wallet. Once the transaction is complete, the balance will appear in the lower left corner of your wallet, and the indicator in the **Requests** table will change from **Pending** to **Paid**.

Once you have used an address, you can either continue using it or click **New** to generate a new address.

Monitoring transactions

The **History** tab lists all current and pending transactions. A transaction to an address in your wallet will appear in the list soon after it is made. Initially, this transaction will be marked as **Unconfirmed**, followed by a clock indicator on the left. As the GoByte network processes the transaction, the status will update in the transaction history list. The



Fig. 95: Password prompt and transaction confirmation in GoByte Electrum wallet

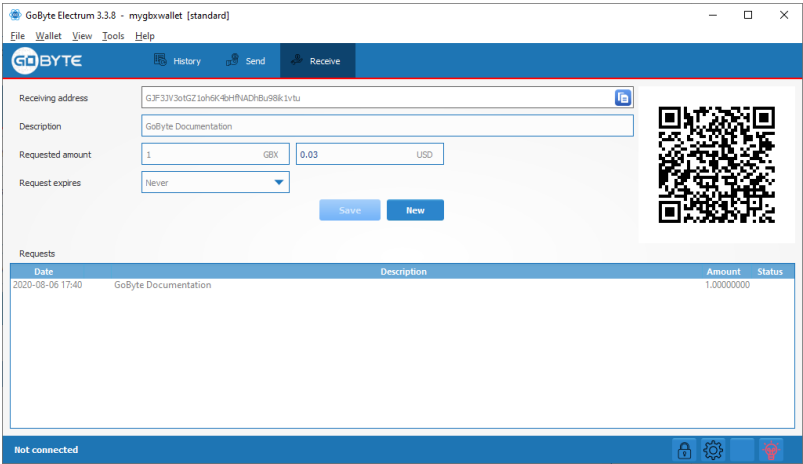


Fig. 96: Transaction ready to send in GoByte Electrum wallet



Fig. 97: Successfully received payment in GoByte Electrum wallet

network confirms transactions with a new block roughly every 2.5 minutes, and a transaction is considered confirmed (and therefore spendable) after six confirmations. These processed transactions are denoted with a green checkmark and the timestamp at which the transaction was made.



To view additional transaction details, right click a transaction on the **History** tab and select **Details** from the context menu. You can also use this menu to copy the transaction ID to the clipboard (this can be used as proof that a given transaction occurred), edit the transaction description for your records or view the transaction on an external block explorer.

Wallet security

Change password

To change the wallet's password, select the **Wallet > Password** option from the main menu, or click on the lock icon in the lower right of the main window. Enter and confirm a new secure password. Should you forget your wallets' password, all is not lost. Your wallet can be restored in its entirety using the backup procedure described here.

Backup

In GoByte Electrum, a seed is a complete backup of all addresses and transactions. Access your wallet's seed through the seed icon in the lower right of the main screen, or the **Wallet > Seed** main menu option. When prompted, enter the secure password you chose when setting up the GoByte Electrum wallet.

Hand-copy the twelve words found in the box to a piece of paper and store it in a safe location. Remember, anyone who finds your seed can spend all of the funds in your wallet.

Alternatively, a backup file can be saved using the **File > Save Copy** main menu option. This file stores the wallet's encrypted seed along with any imported addresses. Restoring this backup will require the wallet password.



Fig. 98: GoByte Electrum wallet History tab immediately after receiving a transaction and after confirmation is complete



Fig. 99: Transaction details in GoByte Electrum wallet



Fig. 100: Displaying the wallet recovery seed in GoByte Electrum



Fig. 101: Viewing the recovery seed

Restore

The only thing needed to recover a GoByte Electrum wallet on another computer is its seed. You can test wallet recovery with your current installation of GoByte Electrum by selecting the **File > New/Restore** menu item. A dialog will appear asking you to name your new wallet. Enter a name, select **Standard wallet** as the wallet type and then choose **I already have a seed**.



Fig. 102: Restoring a wallet from an existing seed

Next, copy the twelve word seed into the text field.

If your seed was entered correctly, GoByte Electrum gives you the option to add a password for your wallet. After restoring your wallet, GoByte Electrum will list any existing transactions from this wallet. This process may take a few minutes, and the transactions may appear as **Not Verified**. This problem disappears after restarting the program.

To restore a wallet file without using the recovery seed, copy the file to the application data folder according to your operating system:

- **Linux:** Open Files, select **Go > Go to folder**, copy the path `~/ .electrum-gobyte` and paste it into the dialog box.
- **macOS:** Open Finder, select **Go > Go to Folder**, copy the path `~/ .electrum-gobyte` and paste it into the dialog box.
- **Windows:** Open Explorer, copy the path `%APPDATA%\Electrum-GBX` and paste it in to the address bar.

Frequently Asked Questions



wallets/electrum/img/restore-phrase.png

Fig. 103: Entering the recovery seed



Fig. 104: Unverified transactions after recovery

How does GoByte Electrum work?

GoByte Electrum focuses on speed, low resource usage and providing a simple user experience for GoByte. Startup times are instant because it operates in conjunction with high-performance servers that handle the most complicated parts of the GoByte system.

Does GoByte Electrum trust servers?

Not really; the GoByte Electrum client never sends private keys to the servers. In addition, it verifies the information reported by servers using a technique called [Simple Payment Verification](#).

What is the Seed?

The seed is a random phrase that is used to generate your private keys. Example:

```
constant forest adore false green weave stop guy fur freeze giggle clock
```

Your wallet can be entirely recovered from its seed. To do this, select the **I already have a seed** option during startup.

How secure is the seed?

The seed created by GoByte Electrum has 128 bits of entropy. This means that it provides the same level of security as a GoByte private key (of length 256 bits). Indeed, an elliptic curve key of length n provides $n/2$ bits of security.

What are change addresses?

The GoByte Electrum wallet design and workflow are based on a concept called a “wallet generation seed”. This seed is a unique, randomly- selected list of twelve words. A GoByte Electrum wallet uses its seed as a template for generating addresses.

To understand the problem that seeds solve, browse to the Electrum **Receive** tab. Next, open the collapsible entry marked **Change**.

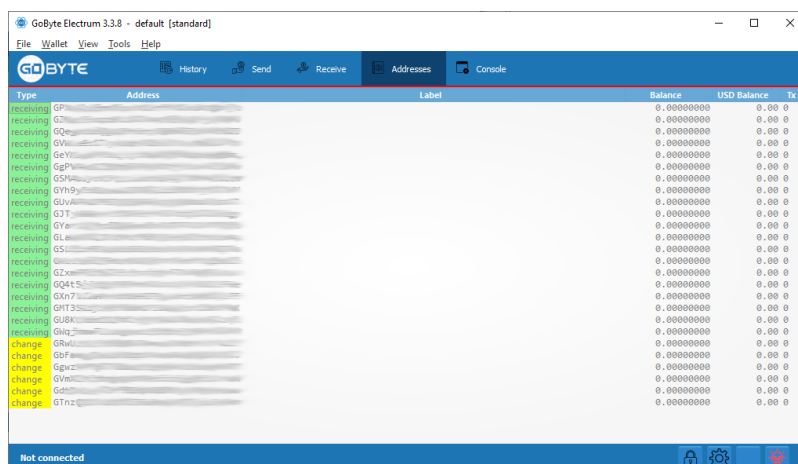


Fig. 105: Receiving and change addresses in GoByte Electrum

Notice that the total balance does not only show the sum of all receiving addresses, but also the separately listed **Change** addresses. Where did these new change addresses come from and why does the first one now hold funds?

GoByte is an electronic cash system, meaning that it shares much in common with the process of using paper banknotes. Although some cash payments involve exact change, many do not. You tend to “overpay” when using cash, and expect to receive the difference as change. Perhaps surprisingly, this is how GoByte transactions work as well. If the entire balance of an address is not required for any given transaction, the remainder is sent to a new and unused address under control of the same wallet. This address is generated deterministically (rather than randomly) from the wallet seed, which means that any other wallet will also regenerate the change addresses in the same order from the same recovery seed, and have access to the balances.

Spending the entire balance and sending any remainder to a change address is considered good practice because it prevents the transaction recipient from linking transactions by browsing the blockchain, thus compromising your privacy. If privacy is not a concern, change addresses can be disabled via the **Tools > Electrum preferences** menu option.

How can I send the maximum available in my wallet?

Type an exclamation mark (!) in the **Amount** field or simply click the **Max** button. The fee will be automatically adjusted for that amount.

How can I send GoByte without paying a transaction fee?

You can create a zero fee transaction in the GUI by following these steps:

- Enable the **Edit fees manually** option
- Enter 0 in the **Fee** field
- Enter the amount in the **Amount** field

Note that transactions without fees might not be relayed by the GoByte Electrum server, or by the GoByte network.

Is there a way to enter amounts in USD in GoByte Electrum?

Yes, go to **Tools > Preference > Fiat** and select a **Fiat currency** to display the current exchange rate from the chosen **Source**.

What does it mean to “Freeze” an address in GoByte Electrum?

When you freeze an address, the funds in that address will not be used for sending GoByte. You cannot send GoByte if you don’t have enough funds in your non-frozen addresses.

How is the wallet encrypted?

GoByte Electrum uses two separate levels of encryption:

- Your seed and private keys are encrypted using AES-256-CBC. The private keys are decrypted only briefly, when you need to sign a transaction; for this you need to enter your password. This is done in order to minimize the amount of time during which sensitive information is unencrypted in your computer’s memory.

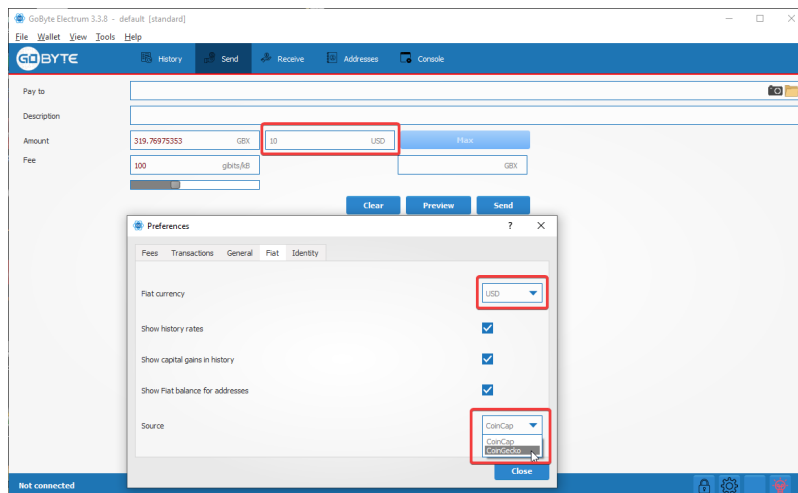


Fig. 106: Entering values in fiat currency in GoByte Electrum

- In addition, your wallet file may be encrypted on disk. Note that the wallet information will remain unencrypted in the memory of your computer for the duration of your session. If a wallet is encrypted, then its password will be required in order to open it. Note that the password will not be kept in memory; GoByte Electrum does not need it in order to save the wallet on disk, because it uses asymmetric encryption (ECIES).

Wallet file encryption is activated by default since version 2.8. It is intended to protect your privacy, but also to prevent you from requesting GoByte on a wallet that you do not control.

I have forgotten my password but still have my seed. Is there any way I can recover my password?

It is not possible to recover your password. However, you can restore your wallet from its seed phrase and choose a new password. If you lose both your password and your seed, there is no way to recover your money. This is why we ask you to save your seed phrase on paper.

To restore your wallet from its seed phrase, create a new wallet, select the type, choose **I already have a seed** and proceed to input your seed phrase.

Does GoByte Electrum support cold wallets?

Yes. See the [cold storage](#) section.

Can I import private keys from other GoByte clients?

In GoByte Electrum 2.0, you cannot import private keys in a wallet that has a seed. You should sweep them instead.

If you want to import private keys and not sweep them you need to create a special wallet that does not have a seed. For this, create a new wallet, select **Use public or private keys**, and instead of typing your seed, type a list of private keys, or a list of addresses if you want to create a watching-only wallet. A master public (xpub) or private (xprv) will also work to import a hierarchical deterministic series of keys. You will need to back up this wallet, because it cannot be recovered from seed.

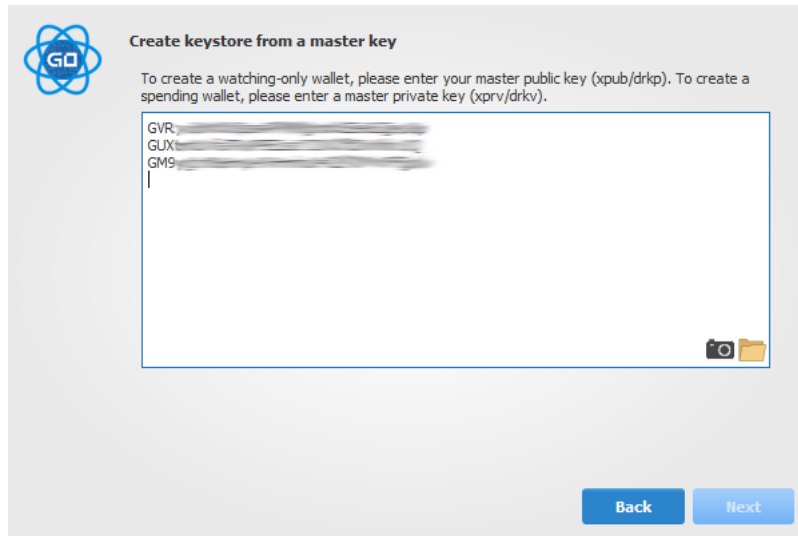


Fig. 107: Importing a list of private keys to create a wallet

Can I sweep private keys from other GoByte clients?

Sweeping private keys means to send all the GoByte they control to an existing address in your wallet. The private keys you sweep do not become a part of your wallet. Instead, all the GoByte they control are sent to an address that has been deterministically generated from your wallet seed.

To sweep private keys go to **Wallet > Private Keys > Sweep**. Enter the private keys in the appropriate field. Leave the **Address** field unchanged. This is the destination address from your existing GoByte Electrum wallet. Click on **Sweep**. GoByte Electrum then takes you to the **Send** tab where you can set an appropriate fee and then click on **Send** to send the coins to your wallet.

Where is my wallet file located?

The default wallet file is called `default_wallet` and is created when you first run the application. It is located under the `/wallets` folder.

- **Linux:** Open Files, select **Go > Go to folder**, copy the path `~/ .electrum-gbx` and paste it into the dialog box
- **macOS:** Open Finder, select **Go > Go to Folder**, copy the path `~/ .electrum-gbx` and paste it into the dialog box
- **Windows:** Open Explorer, copy the path `%APPDATA%\Electrum-GBX` and paste it in to the address bar

Can I do bulk payments with GoByte Electrum?

You can create a transaction with several outputs. In the GUI, type each address and amount on a line, separated by a comma.

Amounts are in the current unit set in the client. The total is shown in the GUI. You can also import a CSV file in the **Pay to** field by clicking on the folder icon.

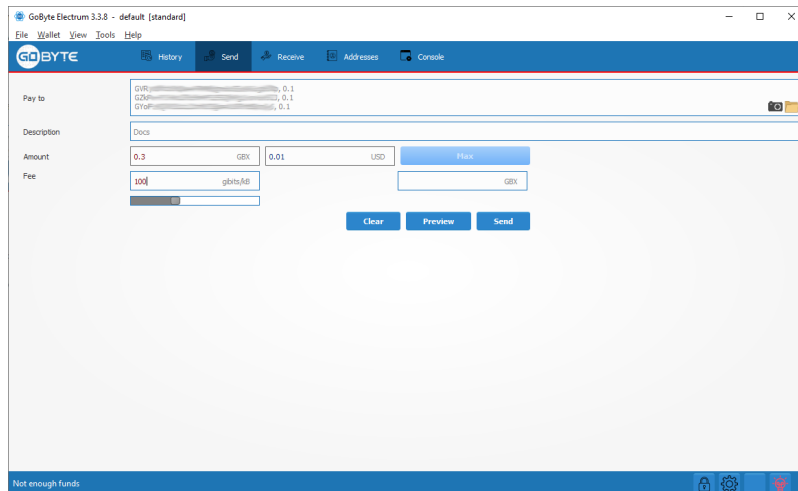


Fig. 108: Creating a transaction with multiple outputs in GoByte Electrum

Can GoByte Electrum create and sign raw transactions?

GoByte Electrum lets you create and sign raw transactions right from the user interface using a form.

GoByte Electrum freezes when I try to send GoByte

This might happen if you are trying to spend a large number of transactions outputs (for example, if you have collected hundreds of donations from a GoByte faucet). When you send GoByte, GoByte Electrum looks for unspent coins that are in your wallet in order to create the new transaction. Unspent coins can have different values, much like physical coins and bills.

If this happens, you should consolidate your transaction inputs by sending smaller amounts of GoByte to one of your wallet addresses; this would be the equivalent of exchanging a stack of nickels for a dollar bill.

What is the gap limit?

The gap limit is the maximum number of consecutive unused addresses in your deterministic sequence of addresses. GoByte Electrum uses it in order to stop looking for addresses. In GoByte Electrum 2.0, it is set to 20 by default, so the client will get all addresses until 20 unused addresses are found.

How can I pre-generate new addresses?

GoByte Electrum will generate new addresses as you use them, until it hits the *gap limit*.

If you need to pre-generate more addresses, you can do so by typing `wallet.create_new_address()` in the console. This command will generate one new address. Note that the address will be shown with a red background in the address tab, to indicate that it is beyond the gap limit. The red color will remain until the gap is filled.

WARNING: Addresses beyond the gap limit will not automatically be recovered from seed. To recover them will require either increasing the client's gap limit or generating new addresses until the used addresses are found.

If you wish to generate more than one address, you may use a 'for' loop. For example, if you wanted to generate 50 addresses, you could do this:

```
for x in range(0, 50):
    print wallet.create_new_address()
```

How to upgrade GoByte Electrum?

Warning: always save your wallet seed on paper before doing an upgrade.

To upgrade GoByte Electrum, just *install* the most recent version. The way to do this will depend on your OS. Note that your wallet files are stored separately from the software, so you can safely remove the old version of the software if your OS does not do it for you.

Some GoByte Electrum upgrades will modify the format of your wallet files. For this reason, it is not recommended to downgrade GoByte Electrum to an older version once you have opened your wallet file with the new version. The older version will not always be able to read the new wallet file.

The following issues should be considered when upgrading GoByte Electrum 1.x wallets to GoByte Electrum 2.x:

- GoByte Electrum 2.x will need to regenerate all of your addresses during the upgrade process. Please allow it time to complete, and expect it to take a little longer than usual for GoByte Electrum to be ready.
- The contents of your wallet file will be replaced with a GoByte Electrum 2 wallet. This means GoByte Electrum 1.x will no longer be able to use your wallet once the upgrade is complete.
- The **Addresses** tab will not show any addresses the first time you launch GoByte Electrum 2. This is expected behaviour. Restart GoByte Electrum 2 after the upgrade is complete and your addresses will be available.
- Offline copies of GoByte Electrum will not show the addresses at all because it cannot synchronize with the network. You can force an offline generation of a few addresses by typing the following into the Console: *wallet.synchronize()*. When it's complete, restart GoByte Electrum and your addresses will once again be available.

Advanced functions

GoByte Electrum is based on [Electrum](#), a Bitcoin wallet. Most functions are identical, which means it is not necessary to reproduce the entirety of the Electrum documentation here. The following sections describe some frequently used advanced functions. For further details on other advanced functions in Electrum for both Bitcoin and GoByte, please click the links below.

- [Electrum documentation](#)
- [Electrum seed version system](#)
- [Electrum protocol specification](#)
- [Serialization of unsigned or partially signed transactions](#)
- [Simple Payment Verification](#)
- [The Python Console](#)
- [Using Electrum Through Tor](#)

Masternodes in GoByte Electrum

GoByte Electrum supports masternode creation through an interface called the **Masternode Manager**. The functionality is available starting from the protocol version 70201.

Masternode Manager

The Masternode Manager can be accessed either from the **Wallet > Masternodes** menu or by pressing **Ctrl+M**. This manager displays the status of your masternode(s). A wallet with no masternodes will begin with a default masternode for which you can fill in the necessary information.

The manager displays the following data about each masternode you have set up:

- The alias (name) of the masternode.
- The status of the masternode (e.g. whether it has been activated).
- The collateral payment of the masternode.
- The private delegate key.
- The IP address and port that your masternode can be reached at.
- The protocol version that your masternode supports.

Masternode setup

A masternode requires a “delegate” key, which is known to both GoByte Electrum and your masternode. Your masternode will use this key to sign messages, and the GoByte network will know that you authorized it to. A delegate key can either be one of your GoByte Electrum keys, or an imported key. Either way, your masternode and GoByte Electrum will both need to know the private key.

To use one of your GoByte Electrum keys as a delegate key, put its private key in the **Masternode Private Key** field of the **View Masternode** tab.

IP address and protocol version

Certain information about your masternode is required. The IP address and port that your masternode uses must be supplied. Also, the protocol version that your masternode supports is required. This information is filled in automatically if you import a “masternode.conf” file.

Collateral

To start a masternode, you must have a 1000 GBX payment available in your wallet. You can scan your wallet for 1000 GBX payments in the **Choose Collateral** tab of the Masternode Manager.

After scanning, a list of available 1000 GBX collateral payments will be displayed. Selecting one of them will cause the selected masternode’s data to be filled in, though these changes won’t be saved until you click the **Save** button in the lower-right corner of the tab.

Activating your masternode

After selecting a collateral payment and specifying a delegate key, you can activate your masternode. Do this by clicking **Activate Masternode** in the **Activate Masternode** tab of the Masternode Manager. If the **Activate Masternode** button cannot be clicked, look at the message in the **Status** bar. It will show you why your masternode cannot be activated.

Activation will require your password if your wallet is encrypted, because a message must be signed. After waiting for GoByte Electrum to sign and broadcast your masternode announcement, you will be presented with a message detailing the result. The status of your masternode will be updated in the table and the **View Masternode** tab.



wallets/electrum/img/mn-view.png

Fig. 109: Entering IP and protocol information



Fig. 110: Entering IP and protocol information



Fig. 111: Entering IP and protocol information

Importing masternode.conf

You can import a *masternode.conf* file using the **Masternode.conf** tab of the Masternode Manager. This is the recommended way of setting up masternodes, as it allows you to configure masternodes for GoByte Core and GoByte Electrum in the same way. Importing a *masternode.conf* file will automatically set up one or more masternode configurations in the Masternode Manager.

Multisig wallets

This tutorial shows how to create a 2 of 2 multisig wallet. A 2 of 2 multisig consists of 2 separate wallets (usually on separate machines and potentially controlled by separate people) that have to be used in conjunction in order to access the funds. Both wallets have the same set of addresses.

- A common use-case for this is if you want to collaboratively control funds: maybe you and your friend run a company together and certain funds should only be spendable if you both agree.
- Another one is security: one of the wallets can be on your main machine, while the other one is on a offline machine. That way you make it very hard for an attacker or malware to steal your coins.

Create a pair of 2-of-2 wallets

Each cosigner needs to do this: In the menu select **File > New**, then select **Multi-signature wallet**. On the next screen, select 2 of 2.



Fig. 112: Selecting x of y signatures for a multi-signature wallet

After generating and confirming your recovery seed, you will be shown the xpub address for this wallet.



Fig. 113: xpub key of the first wallet

After generating a seed (keep it safely!) you will need to provide the master public key of the other wallet. Of course when you create the other wallet, you put the master public key of the first wallet.



Fig. 114: Entering xpub from the second wallet in the first wallet

You will need to do this in parallel for the two wallets. Note that you can press cancel during this step, and reopen the file later.

Receiving

Check that both wallets generate the same set of Addresses. You can now send to these **Addresses** (note they start with a “7”) with any wallet that can send to P2SH Addresses.

Sending

To spend coins from a 2-of-2 wallet, two cosigners need to sign a transaction collaboratively. To accomplish this, create a transaction using one of the wallets (by filling out the form on the **Send** tab). After signing, a window is shown with the transaction details.

The transaction now has to be sent to the second wallet. Several options are available for this:

- You can transfer the file on a USB stick

You can save the partially signed transaction to a file (using the **Save** button), transfer that to the machine where the second wallet is running (via USB stick, for example) and load it there (using **Tools > Load transaction > From file**)



Fig. 115: Partially signed 2-of-2 multisig transaction in GoByte Electrum

- You can use QR codes

A button showing a QR code icon is also available. Clicking this button will display a QR code containing the transaction, which can be scanned into the second wallet (**Tools > Load Transaction > From QR Code**)

With both of the above methods, you can now add the second signature to the transaction (using the **Sign** button). It will then be broadcast to the network.



Fig. 116: Fully signed 2-of-2 multisig transaction in GoByte Electrum

Sweep a paper wallet

You may have received a paper wallet as a gift from another GoByte user, or previously stored one in a safe deposit box. Funds are swept from a *paper wallet* into a live wallet by importing its *private key*, which is a long sequence of characters starting with the number “7” or the capital letter “G”. The example below displays a private key (WIF format).

Funds from paper wallets are swept into an GoByte Electrum Wallet by creating a transaction using the private key and sending it to a new address from your wallet. This is necessary because it is not possible to add new public or private keys to an existing deterministic series of addresses derived from a seed phrase.

Begin by selecting the **Wallet > Private Keys > Sweep** menu item. The **Sweep private keys** dialog will appear, where you can paste your private key(s). An unused address controlled by your GoByte Electrum wallet appears in the lower field, and can be changed by clicking the **Address** button. Once you have pasted your private key, click the **Sweep** button.

GoByte Electrum then prepares a transaction using the private key you just imported to derive the public address for the transaction input and the address from your wallet as the output, and signs the message. Click **Broadcast** to enter



Fig. 117: Public address and associated private key produced by GoByte Paper Wallet Generator



Fig. 118: Entering the private key

the transaction on the blockchain. The balance will then appear in your wallet under the specified address. The address you swept is left with zero balance.



Fig. 119: Broadcasting the sweep transaction

Cold storage

This section shows how to create an offline wallet that holds your GoByte and a watching-only online wallet that is used to view its history and to create transactions that have to be signed with the offline wallet before being broadcast on the online one.

Create an offline wallet

Create a wallet on an offline machine, as per the usual process (**File > New**). After creating the wallet, go to **Wallet -> Master Public Keys**.

The Master Public Key of your wallet is the string shown in this popup window. Transfer that key to your online machine somehow.

Create a watching-only version of your wallet

On your online machine, open GoByte Electrum and select **File > New/Restore**. Enter a name for the wallet and select **Use public or private keys**. Paste your master public key in the box. Click **Next** to complete the creation of your wallet. When you're done, you should see a popup informing you that you are opening a watching-only wallet.

The transaction history of your cold wallet should then appear.

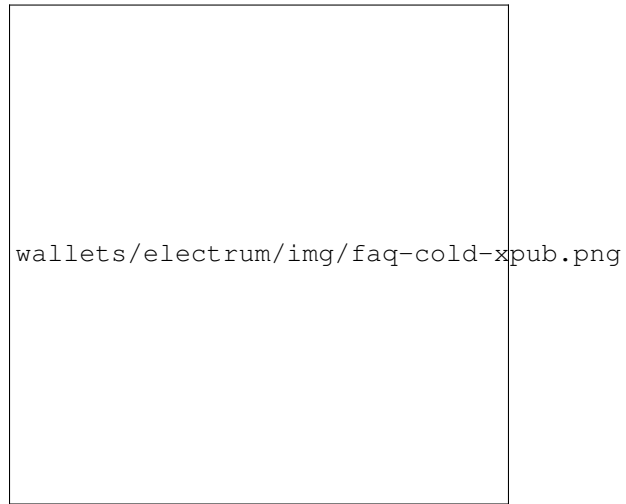


Fig. 120: Master Public Key of a new offline wallet



Fig. 121: Master Public Key of a new offline wallet

Create an unsigned transaction

Go to the **Send** tab on your online watching-only wallet, input the transaction data and click **Send**. A window will appear to inform you that a transaction fee will be added. Continue. In the window that appears up, click **Save** and save the transaction file somewhere on your computer. Close the window and transfer the transaction file to your offline machine (e.g. with a USB stick).

Sign your transaction

On your offline wallet, select **Tools > Load transaction -> From file** in the menu and select the transaction file created in the previous step. Click **Sign**. Once the transaction is signed, the Transaction ID appears in its designated field. Click **Save**, store the file somewhere on your computer, and transfer it back to your online machine.

Broadcast your transaction

On your online machine, select **Tools -> Load transaction -> From file** from the menu. Select the signed transaction file. In the window that opens up, click **Broadcast**. The transaction will be broadcast over the GoByte network.

Command line

GoByte Electrum has a powerful command line available when running under Linux or macOS. This section will show you a few basic principles.

Using the inline help

To see the list of GoByte Electrum commands, type:

```
electrum help
```

To see the documentation for a command, type:

```
electrum help <command>
```

Magic words

The arguments passed to commands may be one of the following magic words: ! ? : -.

The exclamation mark **!** is a shortcut that means ‘the maximum amount available’. Note that the transaction fee will be computed and deducted from the amount. Example:

```
electrum payto GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1 !
```

A question mark **?** means that you want the parameter to be prompted. Example:

```
electrum signmessage GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1 ?
```

Use a colon **:** if you want the prompted parameter to be hidden (not echoed in your terminal). Note that you will be prompted twice in this example, first for the private key, then for your wallet password:

```
electrum importprivkey :
```

A parameter replaced by a gobyte - will be read from standard input (in a pipe):

```
cat LICENCE | electrum signmessage GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1 -
```

Aliases

You can use DNS aliases in place of bitcoin addresses, in most commands:

```
electrum payto ecdsa.net !
```

Formatting outputs using jq

Command outputs are either simple strings or json structured data. A very useful utility is the 'jq' program. Install it with:

```
sudo apt-get install jq
```

The following examples use it.

Sign and verify message

We may use a variable to store the signature, and verify it:

```
sig=$(cat LICENCE| electrum signmessage GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1 -)
```

And:

```
cat LICENCE | electrum verifymessage GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1 $sig -
```

Show the values of your unspents

The *listunspent* command returns a list of dict objects, with various fields. Suppose we want to extract the *value* field of each record. This can be achieved with the jq command:

```
electrum listunspent | jq 'map(.value)'
```

Select only incoming transactions from history

Incoming transactions have a positive 'value' field:

```
electrum history | jq '.[ ] | select(.value>0)'
```

Filter transactions by date

The following command selects transactions that were timestamped after a given date:

```
after=$(date -d '07/01/2015' +"%s")
electrum history | jq --arg after $after '.[[] | select(.timestamp>($after|tonumber))'
```

Similarly, we may export transactions for a given time period:

```
before=$(date -d '08/01/2015' +"%s")
after=$(date -d '07/01/2015' +"%s")
electrum history | jq --arg before $before --arg after $after '.[[] | select(
  .timestamp>($after|tonumber) and .timestamp<($before|tonumber))'
```

Encrypt and decrypt messages

First we need the public key of a wallet address:

```
pk=$(electrum getpubkeys GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1 | jq -r '.[0]')
```

Encrypt:

```
cat | electrum encrypt $pk -
```

Decrypt:

```
electrum decrypt $pk ?
```

Note: this command will prompt for the encrypted message, then for the wallet password.

Export private keys and sweep coins

The following command will export the private keys of all wallet addresses that hold some GoByte:

```
electrum listaddresses --funded | electrum getprivatekeys -
```

This will return a list of lists of private keys. In most cases, you want to get a simple list. This can be done by adding a jq filer, as follows:

```
electrum listaddresses --funded | electrum getprivatekeys - | jq 'map(.[0])'
```

Finally, let us use this list of private keys as input to the sweep command:

```
electrum listaddresses --funded | electrum getprivatekeys - | jq 'map(.[0])' |
  electrum sweep - [destination address]
```

Using cold storage with the command line

This section will show you how to sign a transaction with an offline GoByte Electrum wallet using the command line.

Create an unsigned transaction

With your online (watching-only) wallet, create an unsigned transaction:

```
electrum payto GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1 0.1 --unsigned > unsigned.txn
```

The unsigned transaction is stored in a file named ‘unsigned.txn’. Note that the `--unsigned` option is not needed if you use a watching-only wallet.

You may view it using:

```
cat unsigned.txn | electrum deserialize -
```

Sign the transaction

The serialization format of GoByte Electrum contains the master public key needed and key derivation used by the offline wallet to sign the transaction. Thus we only need to pass the serialized transaction to the offline wallet:

```
cat unsigned.txn | electrum signtransaction - > signed.txn
```

The command will ask for your password, and save the signed transaction in ‘signed.txn’.

Broadcast the transaction

Send your transaction to the GoByte network, using broadcast:

```
cat signed.txn | electrum broadcast -
```

If successful, the command will return the ID of the transaction.

How to accept GoByte on a website using GoByte Electrum

This tutorial will show you how to accept gobyte on a website with SSL signed payment requests. It is updated for GoByte Electrum 2.6.

Requirements

- A webserver serving static HTML
- A SSL certificate (signed by a CA)
- Electrum version `>= 2.6`

Create a wallet

Create a wallet on your web server:

```
electrum create
```

You can also use a watching only wallet (restored from xpub), if you want to keep private keys off the server. Once your wallet is created, start GoByte Electrum as a daemon:


```
electrum daemon start
```

Add your SSL certificate to your configuration

You should have a private key and a public certificate for your domain. Create a file that contains only the private key:

```
-----BEGIN PRIVATE KEY-----
your private key
-----BEGIN END KEY-----
```

Set the path to your the private key file with setconfig:

```
electrum setconfig ssl_privkey /path/to/ssl.key
```

Create another file that contains your certificate and the list of certificates it depends on, up to the root CA. Your certificate must be at the top of the list, and the root CA at the end:

```
-----BEGIN CERTIFICATE-----
your cert
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
intermediate cert
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
root cert
-----END CERTIFICATE-----
```

Set the `ssl_chain` path with setconfig:

```
electrum setconfig ssl_chain /path/to/ssl.chain
```

Configure a requests directory

This directory must be served by your webserver (eg Apache):

```
electrum setconfig requests_dir /var/www/r/
```

By default, GoByte Electrum will display local URLs, starting with `'file:/'`. In order to display public URLs, we need to set another configuration variable, `url_rewrite`. For example:

```
electrum setconfig url_rewrite "['file:///var/www/', 'https://electrum.org/']"
```

Create a signed payment request

```
electrum addrequest 3.14 -m "this is a test"
{
  "URI": "gobyte:GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1?amount=3.14&r=https://electrum.
↪org/r/7c2888541a",
  "address": "GZMkeSpJgyog12qrSyxvPfThRn5Vu5DWE1",
  "amount": 314000000,
  "amount (GBX)": "3.14",
```

(continues on next page)

(continued from previous page)

```
"exp": 3600,  
"id": "7c2888541a",  
"index_url": "https://electrum.org/r/index.html?id=7c2888541a",  
"memo": "this is a test",  
"request_url": "https://electrum.org/r/7c2888541a",  
"status": "Pending",  
"time": 1450175741  
}
```

This command returns a json object with two URLs:

- *request_url* is the URL of the signed BIP70 request.
- *index_url* is the URL of a webpage displaying the request.

Note that *request_url* and *index_url* use the domain name we defined in *url_rewrite*. You can view the current list of requests using the *listrequests* command.

Open the payment request page in your browser

Let us open *index_url* in a web browser.

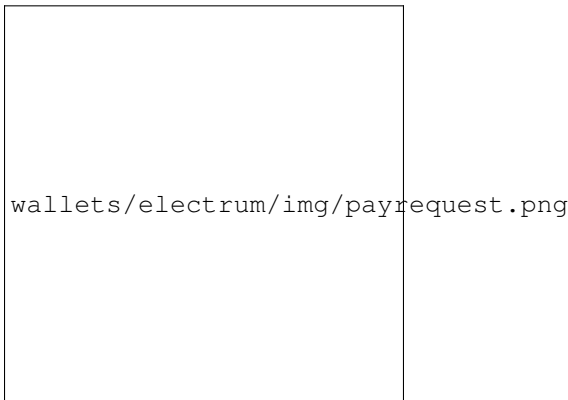


Fig. 122: Payment request page in a web browser

The page shows the payment request. You can open the gobyte: URI with a wallet, or scan the QR code. The bottom line displays the time remaining until the request expires.

This page can already be used to receive payments. However, it will not detect that a request has been paid; for that we need to configure websockets.

Add web sockets support

Get SimpleWebSocketServer from here:

```
git clone https://github.com/ecdsa/simple-websocket-server.git
```

Set *websocket_server* and *websocket_port* in your config:

```
electrum setconfig websocket_server <FQDN of your server>  
electrum setconfig websocket_port 12455
```



Fig. 123: Wallet awaiting payment

And restart the daemon:

```
electrum daemon stop
electrum daemon start
```

Now, the page is fully interactive: it will update itself when the payment is received. Please notice that higher ports might be blocked on some client's firewalls, so it is more safe for example to reverse proxy websockets transmission using standard 443 port on an additional subdomain.

JSONRPC interface

Commands to the GoByte Electrum daemon can be sent using JSONRPC. This is useful if you want to use GoByte Electrum in a PHP script.

Note that the daemon uses a random port number by default. In order to use a stable port number, you need to set the *rpcport* configuration variable (and to restart the daemon):

```
electrum setconfig rpcport 7777
```

With this setting, we can perform queries using curl or PHP. Example:

```
curl --data-binary '{"id":"curltext","method":"getbalance","params":[]}' http://127.0.0.1:7777
```

Query with named parameters:

```
curl --data-binary '{"id":"curltext","method":"listaddresses","params":{"funded":true}}' http://127.0.0.1:7777
```

Create a payment request:

```
curl --data-binary '{"id":"curltext","method":"addrequest","params":{"amount":"3.14","memo":"test"}}' http://127.0.0.1:7777
```

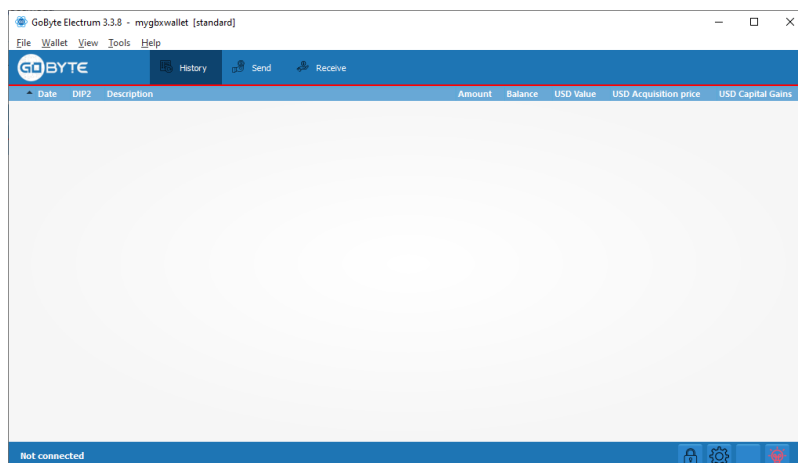
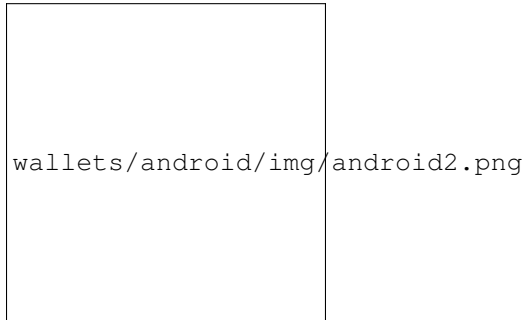
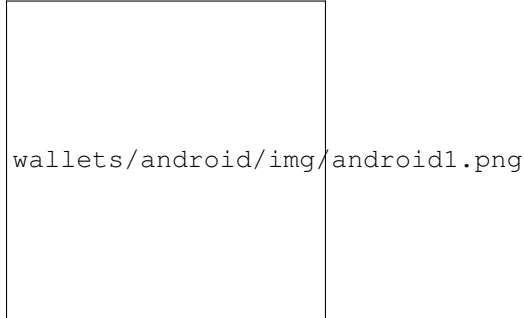


Fig. 124: GoByte Electrum Wallet

1.6.3 GoByte Android Wallet

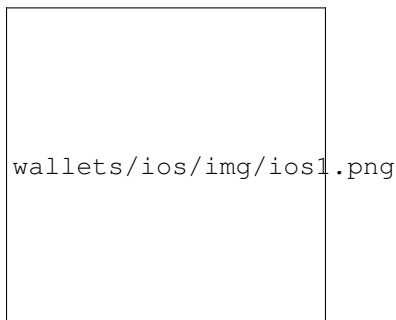
GoByte offers a standalone wallet for Android, with development supported by the GoByte budget. The GoByte Android Wallet supports advanced GoByte features, including contact management and InstantSend. You can scan and display QR codes for quick transfers, backup and restore your wallet, keep an address book of frequently used addresses, pay with NFC, sweep paper wallets and more.

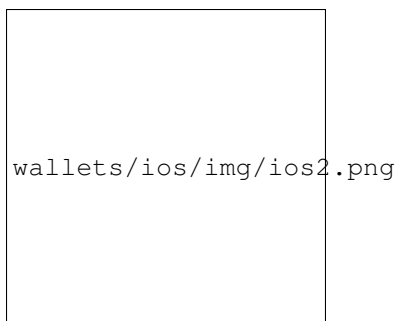


GoByte Android Wallet

1.6.4 GoByte iOS Wallet

GoByte offers a standalone wallet for iOS, with development supported by the GoByte budget. The official GoByte Wallet supports advanced GoByte features such as InstantSend sending and receiving. You can also scan and display QR codes for quick transfers and backup your wallet using a secure recovery phrase.



*GoByte iOS Wallet*

1.6.5 GoByte Paper Wallet

The [GoByte Paper Wallet generator](#) allows you to generate, encrypt and secure the keys to a single GoByte address on a clean computer without ever connecting to the internet. Perfect for long term secure storage.

Introduction

A paper wallet is a method of storing a private key to access funds stored on a single address. It can be generated on a computer that has never been connected to the internet, and printed out for air-gapped offline storage. As such, they are suitable for storing large amounts of GoByte, but care must be taken not to lose the private key, since there is no way of recovering funds if it is ever lost. To use the key, it must be imported or “swept” into an online wallet and should not be used again. Paper wallets are extremely secure but somewhat inconvenient for everyday use compared to hardware wallets, which also offer a high degree of security.

Paper wallets use random user and machine input to create a set of keys/addresses which you then print. You can never regenerate a paper wallet once you turn off the machine. What you print is all you get. For this reason, paper wallets are somewhat vulnerable and require special care because they can get damaged, lost, destroyed or stolen. Even if you encrypt them with BIP38 (which you should), a sufficiently motivated adversary (e.g. robbery/home invasion) could bypass this encryption using the proverbial “\$5 wrench attack”.

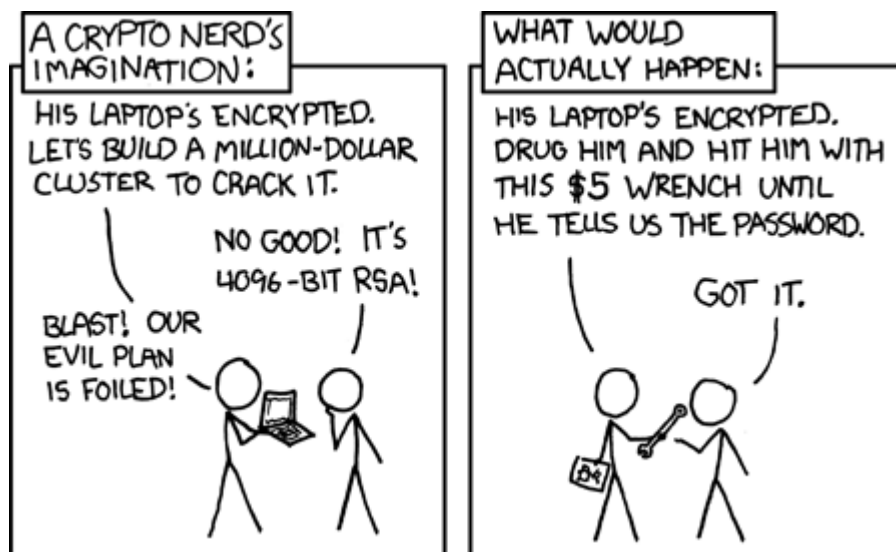


Fig. 125: The \$5 wrench attack. Credit: xkcd.com

Nevertheless, together with appropriate planning, paper wallets are a highly convenient and user-friendly way to store GoByte long term.

Security

While you can create a paper wallet using a machine that is connected to the internet, wallets that will be used to store significant funds should be generated using an offline computer running a single-use operating system to ensure that all generated data will be permanently wiped from memory once the process is complete.

A simple method of doing this is to burn a live Linux CD. [Ubuntu Desktop](#) is recommended because it will have the most drivers and is simple to use, while [Tails](#) and [Kali Linux](#) are popular choices for extremely strong security. Booting from an actual CD is most secure since it is mounted read-only, but a USB stick is generally fine as well. Both laptops and desktops can be used if you can ensure that all networking hardware is disabled when you get to the stage of actually generating your keys.

Boot from the CD and download/install your tools (or download them ahead of time to a USB drive). Disconnect from the internet, generate your keys/addresses/printouts, and power off the machine. You are now the only person with access to these addresses.

Death plan

Whichever type of cold storage you choose, make a plan to pass on the necessary data to regenerate the keys to your loved ones in the event of an accident - it will happen to us all eventually. Write down your paper wallet BIP38 decryption password or brain wallet passphrase. Then write down instructions on how to use it, and keep them separate with a clear procedure on how they can be accessed when necessary.

Tools

A GoByte paper wallet can be generated in several ways.

- Using the generator at <https://paper.gobyte.network>
- Offline using the GoByte Paper Wallet source code from GitHub at <https://github.com/gobytecoin/paper.gobyte.network/releases/latest>
- Offline using the same code which powers both sites, by viewing the [GitHub project](#) or [downloading directly](#)

Since the source code for all three options is largely similar, this guide will use <https://paper.gobyte.network> as an example. The websites listed here run entirely in your web browser without sending any of the data generated to an external server, but the most secure option is to download the wallet generator and run it on a computer with a freshly installed operating system that is not connected to the internet.

Address generation

Go to <https://paper.gobyte.network> in your web browser (or open index.html if you downloaded the wallet generator). Select your language and choose GoByte as the currency if necessary. The following screen will appear:

Some random data is required to ensure the generated address and key are unique. Move our mouse around and/or type random characters into the text box until the process reaches 100% and the following screen appears:

Once your public address and private key (shown in Wallet Import Format or WIF) are visible on the **Single Wallet** tab, you should immediately click **Print** to print the data and store it securely. If you leave the page without somehow recording the gobyte address and private key, all data will be irretrievably lost, together with any funds you have sent to the address.



Fig. 126: The GoByte Paper Wallet Generator at paper.gobyte.network

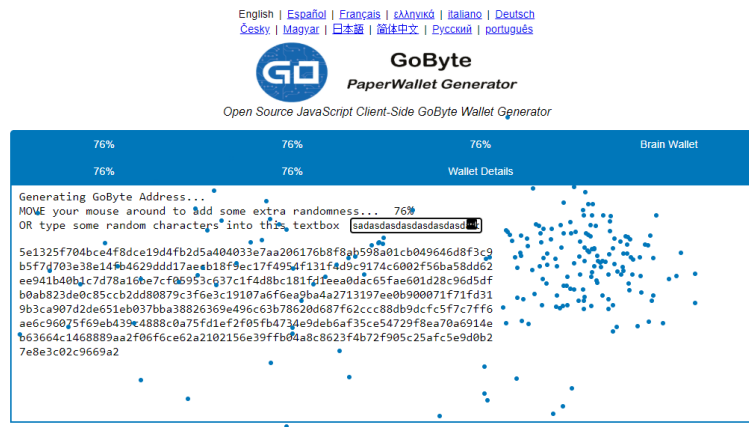


Fig. 127: Generating randomness for the GoByte Paper Wallet Generator

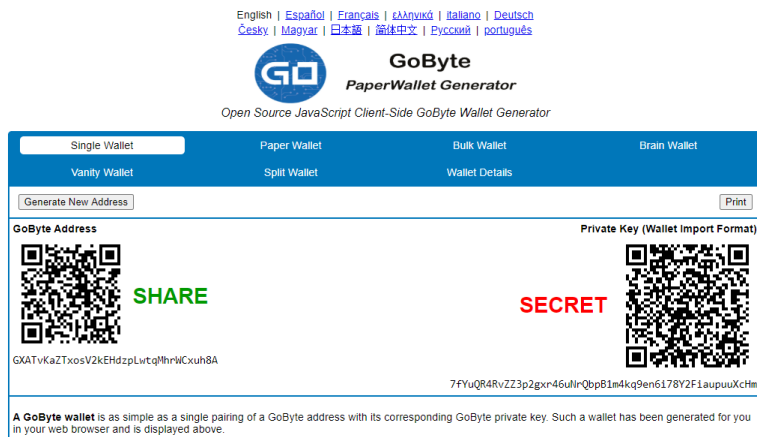


Fig. 128: A GoByte address and private key generated using GoByte Paper Wallet Generator

Encryption

The information shown on the **Single Wallet** tab does not have a passphrase and is not encrypted. You can print this paper wallet as it is and use it, but it is not protected from being stolen if someone finds it. You should keep it safe the same way you would jewels or cash.

If you decide that you need the additional security of a password for this address, click **Paper Wallet**. A different address/key pair will appear. To create an encrypted wallet, select **BIP38 Encrypt?** and enter a passphrase. Tick the box **Hide Art?** and change to **1** the field **Addresses to Generate** and then click on **Generate**. A new wallet will be generated where the private key (WIF) is encrypted using the password you specified, resulting in a BIP38 key. You now need both this BIP38 key and the password to control funds on the address, so be sure to click **Print** and store both safely. If you are unsure about how to use BIP38 encryption, it is highly recommended to test the workflow with a low amount of GoByte before storing significant funds on an encrypted paper wallet. If you forget the password or lose the encrypted key, you will permanently lose access to your funds.

A GoByte WIF address can be easily identified because it always begins with “7”. A BIP38 format encrypted key can be identified because it always begins with “6P”. See [here](#) to learn more about BIP38.



Fig. 129: Encrypted paper wallet generated using GoByte Paper Wallet Generator

Sending to a paper wallet and viewing the balance

You can send GoByte to a paper wallet address in the same way as to any other GoByte address. See the documentation for your wallet if you do not know how to do this. For this example, 0.05 GBX (minus transaction fee) has been sent to the paper wallet address. Anyone with knowledge of the public address is able to see the balance of the wallet using a block explorer, but only someone with knowledge of the private key can access the funds. You can make as many deposits and send as many coins to the same address as you'd like. Just make sure you test your wallet with small amounts first to learn how it works.

Spending from a paper wallet

In order to access the funds stored on the paper wallet address, you will need the following:

- The public address
- The private key in WIF

If you encrypted the wallet, you will additionally need the following to convert the BIP38 key into the WIF key:

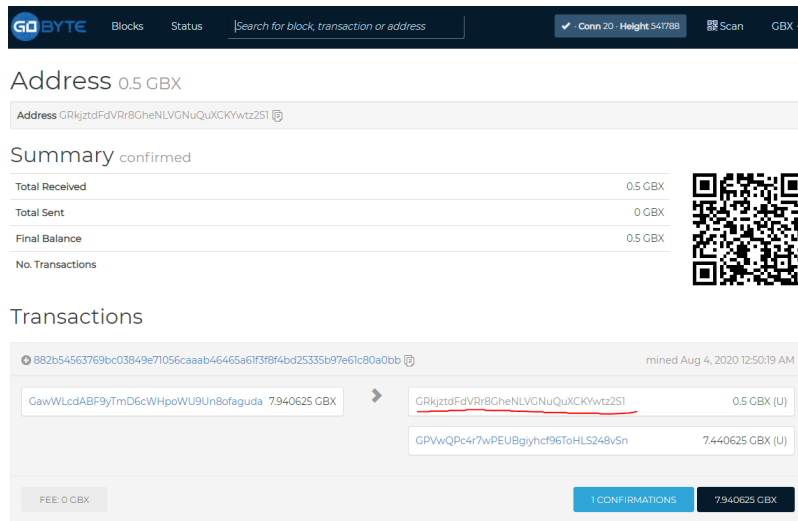


Fig. 130: Viewing the balance of the paper wallet using the GoByte Block Explorer at explorer.gobyte.network

- The encrypted private key in BIP38 format
- The passphrase you used to encrypt the key

Optional: Decrypt BIP38 key to WIF

If you encrypted your paper wallet, you will first need to decrypt the BIP38 key. You can skip this step if your private key was not encrypted.

Go to the **Wallet Details** tab, enter the encrypted key in the **Enter Private Key** field and click **View Details**. You will be asked to **Enter BIP38 Passphrase** in the field. Enter the passphrase and click **Decrypt BIP38**. A range of information derived from the key will appear, the information required to access the funds on the public address appears under **Private Key WIF**. Copy the Private Key WIF and use it in the next step.

Importing the private key to your live wallet

When you are ready to spend the balance on the paper wallet, you will need to import the private key used to control the address printed on the wallet into another GoByte wallet that is connected to the internet. We will use the GoByte Core Wallet in this example, although GoByte Electrum and mobile wallets are also supported. Open GoByte Core Wallet, click **Settings** and **Unlock Wallet**. Enter your wallet passphrase. Then click **Tools** and select **Debug Console**. The console appears. Enter the following command:

```
importprivkey <your private key in WIF>
```

This process requires rescanning the entire downloaded blockchain for transactions involving this address, and will therefore take some time. Be patient. Once the process is complete, any transactions involving the imported address will appear in your list of transactions. If you use Coin Control, you can also enable or disable the address for spending there.

English | Español | Français | 日本語 | Italiano | Deutsch
 Česky | Magyar | 普通话 | 繁體中文 | Pycckий | português

GoByte
 PaperWallet Generator
 Open Source JavaScript Client-Side GoByte Wallet Generator

Single Wallet | Paper Wallet | Bulk Wallet | Brain Wallet
 Vanity Wallet | Split Wallet | **Wallet Details**

Enter Private Key [View Details](#)
 BIP38 Encrypt? ☐
 Key Formats: WIF, WIFC, HEX, B64, B6, MINI, BIP38 [Print](#)

Your GoByte Private Key is a unique secret number that only you know. It can be encoded in a number of different formats. Below we show the GoByte Address and Public Key that corresponds to your Private Key as well as your Private Key in the most popular encoding formats (WIF, WIFC, HEX, B64).

GoByte stores public keys in compressed format. The client now also supports import and export of private keys with `importprivkey/dumpprivkey`. The format of the exported private key is determined by whether the address was generated in an old or new wallet.

GoByte Address

 GRKjztDfVtr8GhehLVGNuQuXCKYvtz251

GoByte Address Compressed

 GSkYHhSDCKY2hEnAR6dVv2RbhhDmEoa79m

Public Key (130 characters [0-9A-F]):
 041000B11377EFD0D642578FA2BE289D4F686715829FC68C2D5410705C087E15A5AC2BFC
 96BF4C2BF3BC842BA7F541B1770203AF7926462C8628E9478A988AFCC

Public Key (compressed, 66 characters [0-9A-F]):
 021000B11377EFD0D642578FA2BE289D4F686715829FC68C2D5410705C087E15A

Private Key WIF
 51 characters base58, starts with a '7'

 7f4xSBzrcF659obU12pxTSdsyKA68bNQg4H8zwRsn1F7FVVVx2

Private Key WIF Compressed
 52 characters base58, starts with a 'W'

 WP2TgAwfGZBPKr-fBuH94fm19myZ991xhVVPUE3jEoAm3GU6EY2Gd

Private Key Hexadecimal Format (64 characters [0-9A-F]):
 98ACC211B62889C7D01A7069B65403C259CF395A7243A417BC2C3088561C9388

Private Key Base64 (44 characters):
 m6zCEBYocrfGdn1pt17Tw1nPOVpyQ8QkvCww11Yck4s=

How do I make a wallet using dice? What is B6? [+](#)

Donations for original project:
 1NINja1bUmhSoTXozBRBEtR8LeF0TGoZBN
[Github Repository \(zip\)](#)
 Copyright bladdress.org. The GoByte Developers. JavaScript copyrights are included in the source. No warranty.

Fig. 131: GoByte Paper Wallet Generator displaying information derived from an encrypted private key

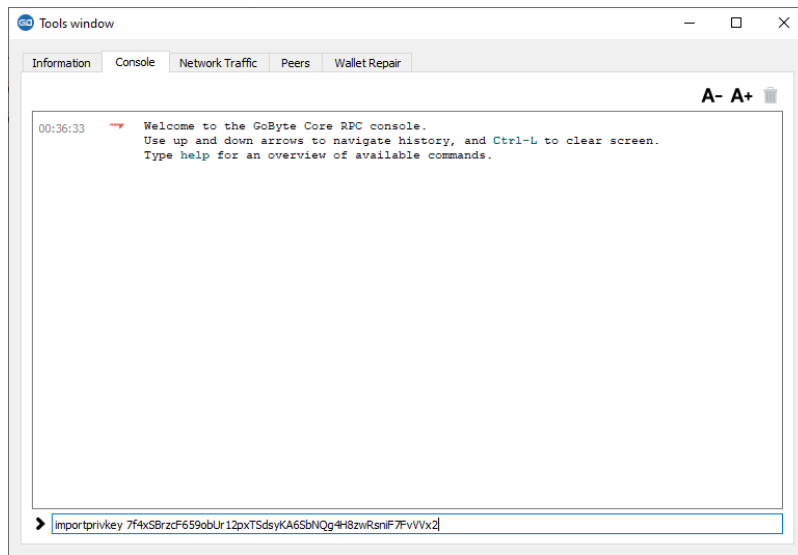
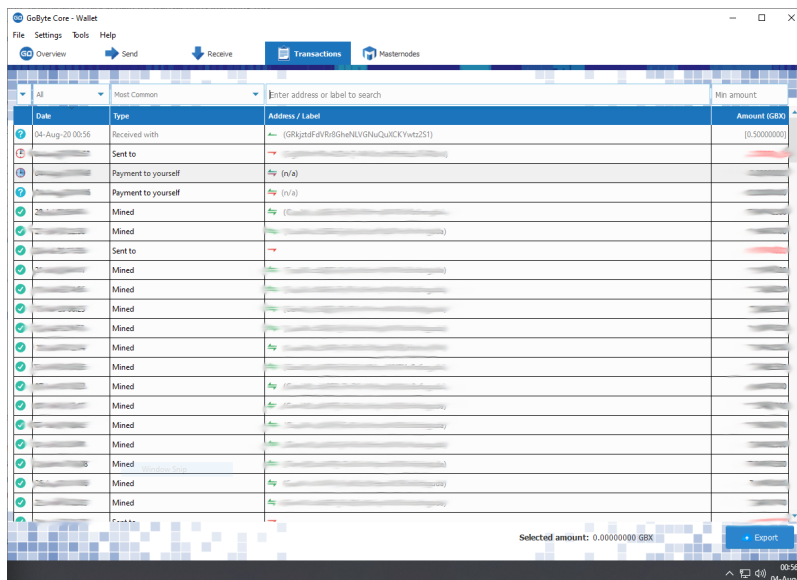


Fig. 132: GoByte Core Wallet importing a private key



Since the paper wallet public address still holds the funds, it can also be imported again into a second wallet if it is not destroyed. It is recommended to transfer the balance from the paper wallet to an internal wallet address or another address where you have exclusive control over the private key. This will prevent a third party from obtaining unauthorised access to the same address from the paper wallet before you do. You can then spend your balance as usual.

Once the paper wallet is empty and you are sure it will not be receiving any further deposits, you can destroy the paper.

1.6.6 Hardware Wallets

A hardware wallet is a type of device which stores private keys for a blockchain in a secure hardware device, instead of in a database file such as `wallet.dat` used with common software wallets. This offers major security advantages over software wallets, as well as practical benefits over paper wallets. To date, there is no verifiable evidence of hardware wallets being compromised by viruses, and they are also immune to keylogger attacks that could be used to steal

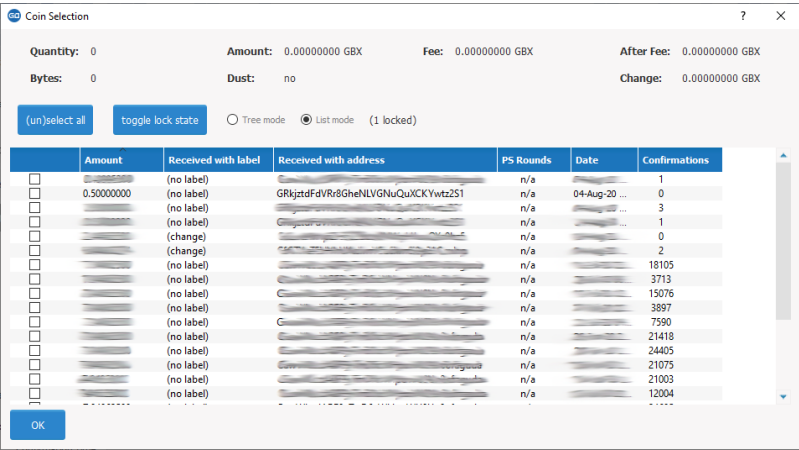


Fig. 133: Paper wallet address successfully imported into GoByte Core Wallet from WIF private key

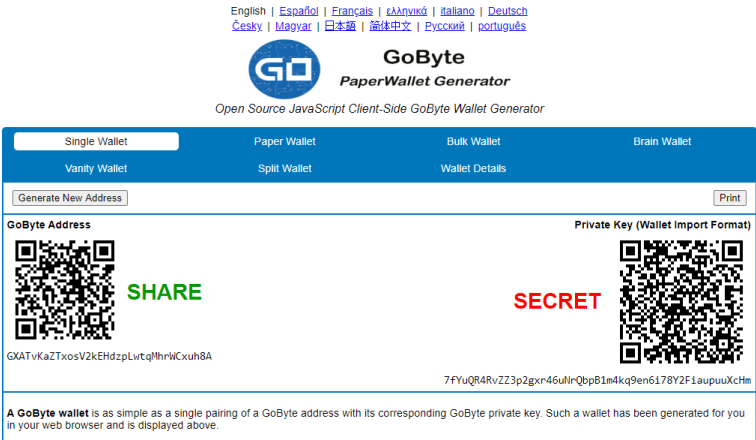


Fig. 134: GoByte Paper Wallet

passwords to unlock the private keys used with software wallets.

Hardware wallets function by storing your private keys in a protected area of a microcontroller. It is impossible for the private key to leave the device in plain text - only the signed output of the cryptographic hash is ever transmitted to the device interacting with the blockchain, such as your computer or smartphone. Most hardware wallets feature a screen which allows you to independently confirm the address you are sending to is correct.

This section lists the most common commercial hardware wallets supporting GoByte, although some other enthusiast projects may also be available.

Introduction

Hardware wallets offer you the security of storing your keys in secure device while still allowing you to make simple transactions through a web interface. Three major manufacturers of hardware wallets currently exist, with GoByte supported on all of them.

Trezor



Developed by Czech startup [SatoshiLabs](https://satoshi-labs.com/), the \$99 device is essentially a USB dongle designed to add an extra authentication layer to all outbound bitcoin transactions. Trezor has supported GoByte since TBA with the release of firmware version TBA.

By virtue of its design, Trezor can be used to sign transactions on ‘unsafe’ computers and is impervious to keyloggers and many other vectors of attack, so even if your host PC is compromised, the attacker has no way of getting your private key. That’s also where the device gets its name, as ‘trezor’ translates into ‘vault’ in most Slavic languages, including Czech. A kind of ‘vault’ for your private bitcoin key, Trezor claims to use a number of clever tricks to maintain security even on compromised and unsafe machines.

- Site: <https://trezor.io>
- Review: <https://www.buybitcoinworldwide.com/wallets/trezor/>
- Shop: <https://shop.trezor.io>
- Wallet: <https://wallet.trezor.io>

It is also possible to operate a GoByte masternode using your Trezor. See [here](#) for details.

Getting Started

Once you have bought your Trezor from <https://shop.trezor.io> or an [authorized reseller](#), you will need a wallet to use it with. Trezor supports the following GoByte wallets:

- [Trezor Wallet \(documentation\)](#)
- [GoByte Electrum Wallet \(documentation\)](#)
- [GoByte Masternode Tool \(documentation\)](#)

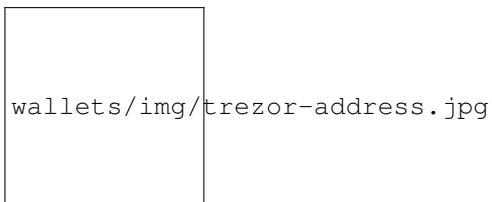
This documentation describes how to get started using the official Trezor web wallet at <https://wallet.trezor.io>. Always confirm the URL is correct and SSL encryption is enabled when working with the Trezor Wallet. Follow these steps when setting up your Trezor for the first time:

1. Inspect the packaging for tampering. There should be two seals and the flaps should be glued shut. It should be impossible to remove the device without totally destroying the packaging.
2. Go to <https://trezor.io/start/> and watch the video to introduce the concepts of a shifting PIN layout and recovery seed.
3. Go to <https://wallet.trezor.io/> to begin the setup process.
4. If not already installed, install the Trezor Bridge application from <https://wallet.trezor.io/#/bridge>
5. Connect the Trezor to your computer when prompted.
6. If this is the first time you connect your Trezor, you will be prompted to install firmware. Click the **Install** button, wait for the download and confirm on the device.
7. When complete, the device will display a fingerprint. Verify that this matches the fingerprint shown on the screen. Note that this is hexadecimal and therefore not case-sensitive.
8. After verification is complete, disconnect and reconnect your device. Enter a device label on the screen that appears.
9. Enter and confirm a PIN by clicking on the squares according to the mapping shown on the device.
10. Your Trezor device will now display a sequence of 24 words on the screen. This is your recovery seed. Write the words down in the order they appear on the recovery card. Never store your recovery seed in any digital format, including photos or text.
11. Verify the seed against what you have written down and store it in a safe place.
12. You will be asked to enter your PIN again.
13. The Trezor Wallet will appear with a message that your device is ready for use. Your device name will appear on the device.
14. Switch to the GoByte wallet using the menu at the top left. You can now use your Trezor to send and receive GoByte.

Receiving GoByte

We will now create a GoByte receiving address and attempt to receive 1.0 GBX.

1. In the Trezor GoByte wallet, click **Account #1**, then click Receive.
2. A GoByte address will appear. Click **Show full address** to verify the address on the Trezor device.



3. Send 1 GBX to this address using an exchange or another wallet.
4. Once the transaction is confirmed, it will appear on the **Transactions** tab of your Trezor Wallet.

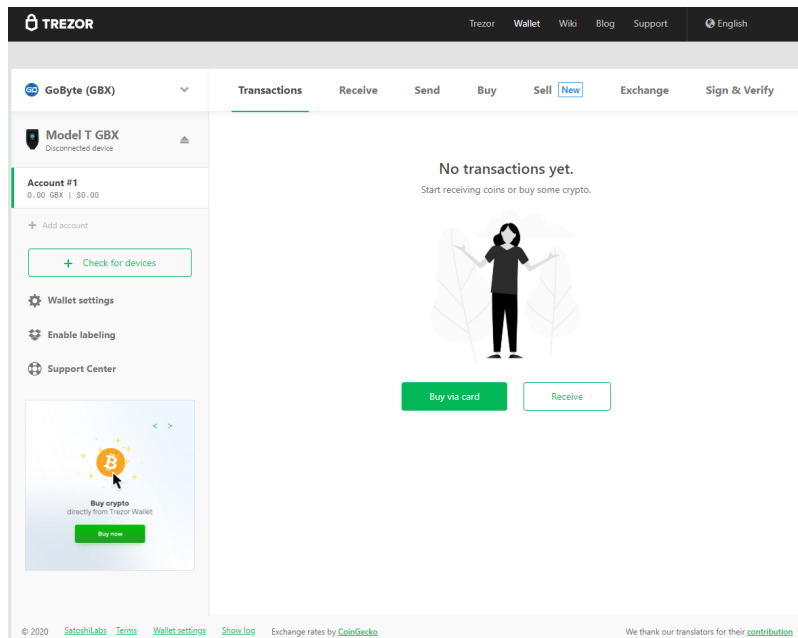


Fig. 135: Trezor Web Wallet for GoByte ready for first use

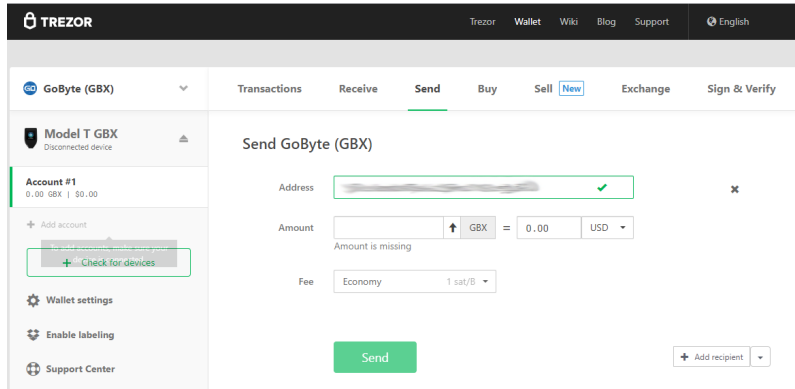


Fig. 136: Trezor Web Wallet after receiving GoByte

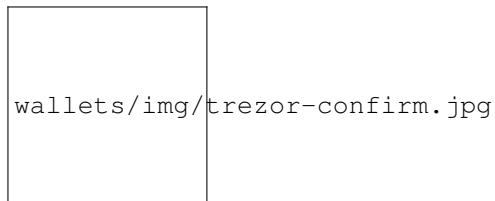
Sending GoByte

We will now send the GoByte (minus transaction costs) to an external address.

1. In the Trezor GoByte wallet, click **Account #1**, then click **Send**.
2. Enter the GoByte address and amount in the fields.



3. Enter your PIN.
4. Confirm the address on the device, then confirm the action.



5. The transaction will be transmitted to the network and the recipient receives the funds.

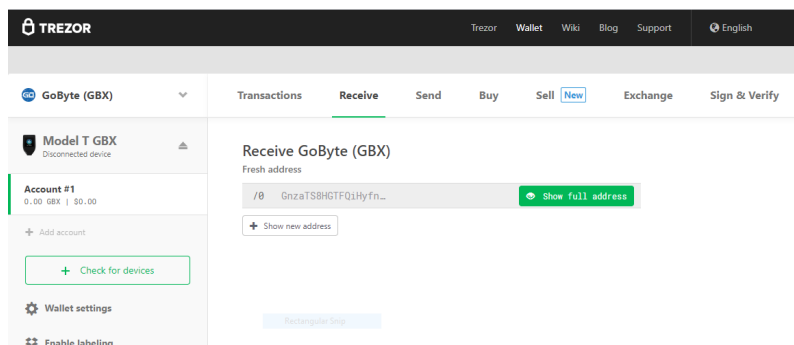


Fig. 137: Trezor Web Wallet after sending GoByte

Advanced Functions

Changing the PIN/Passphrase

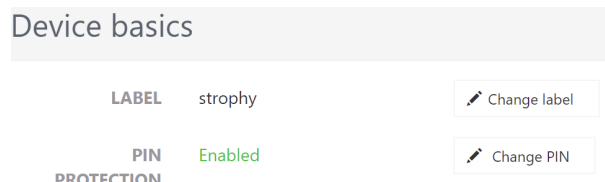
Your PIN is your primary form of security when using your hardware wallet. It is considered secure because the layout of the PIN keypad changes each time you use it. If you suspect your PIN has been compromised, change it using the following instructions. For extra security, or if you believe you may be subjected to duress at some point in the future,

you can add a passphrase to reveal further wallets which only appear when you enter your passphrase. Since the passphrase acts as a cryptographic salt, there is no “correct” passphrase - a different wallet will be displayed for each passphrase you enter. Be absolutely sure you understand passphrases before using them. For more information, see [here](#).

Changing your PIN

You can change your Trezor PIN from both the [Trezor wallet](#) and [GMT](#).

From Trezor: Go to <https://wallet.trezor.io> and click the cog icon next to your username. Then click **Change PIN**. You will need to confirm you want to change your PIN on the hardware device, then enter your existing PIN and the new PIN twice.



From GMT: Open GMT and click **Tools > Hardware Wallet PIN/Passphrase configuration**. The following window will appear. Click **Change**. You will need to confirm you want to change your PIN on the hardware device, then enter your existing PIN and the new PIN twice.



Adding a passphrase

You can add a passphrase to your Trezor from both the Trezor wallet and GMT. Before you add a passphrase, you should be fully aware of how it functions as a “25th word” to your seed, and the risks of forgetting your passphrase. Note that you do not need to enter a passphrase - blank is perfectly acceptable and will reveal your normal wallet.

From Trezor: Click **Advanced**, confirm you understand the risks and click **Enable passphrase encryption**. This enables a prompt to enter a passphrase each time you use your Trezor.

Advanced setup

FIRMWARE	1.5.2
PASSPHRASE	<p>Passphrase encryption allows you to access new wallets, each hidden behind a particular passphrase. Your old accounts will be accessible behind an empty passphrase.</p> <p>If you forget your passphrase, your wallet is lost for good. There is no way to recover your funds.</p> <p>Learn more in user manual ></p> <p><input type="checkbox"/> OK, I understand</p> <p><input checked="" type="checkbox"/> Enable passphrase encryption</p>

From GMT: Open GMT and click **Tools > Hardware Wallet PIN/Passphrase configuration**. The following window will appear. Click **Enable**. This enables a prompt to enter a passphrase each time you use your Trezor.

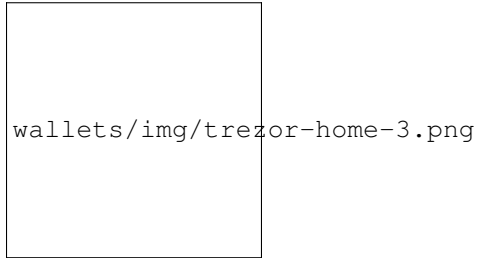
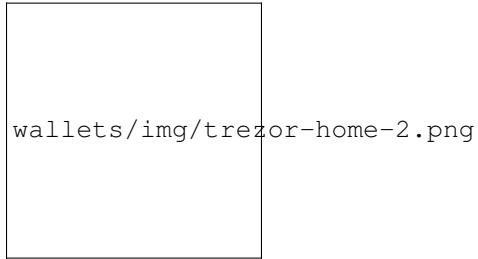
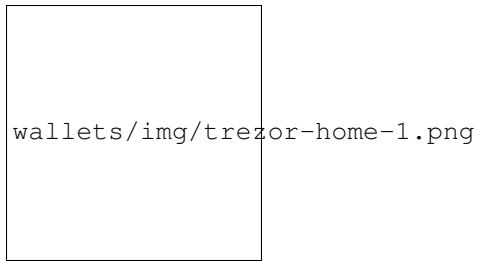


Changing the Homescreen

Your Trezor allows you to change the homescreen image from the default Trezor logo. A range of existing images can be selected, you can generate one yourself using the [Trezor Homescreen Editor](#), or you can create and upload your own 128x64px black and white image. To change your homescreen image:

1. Go to <https://wallet.trezor.io> and open your wallet
2. Click the small cog icon next to your device name
3. Click the **Homescreen** tab
4. Select the new homescreen, then click the **Set as homescreen** button at the top
5. Confirm the change on the Trezor device

A few sample images are available for GoByte:



Storage Suggestions

While losing a Trezor is not a security threat (unless someone knows your PIN and/or passphrase), it is a moderately expensive device that can be damaged by pressure or water, for example. For this reason, GoByte community member tungfa has shared photos of a custom-made Trezor case. The following materials are required:

- [Pelican Case 1010 Micro Case](#)
- Foam
- Trezor + Cable
- USB Stick (for wallet.dat files + blockchains of all portfolios)
- Notepad





KeepKey



The \$129 KeepKey hardware wallet features a large screen and 100% open source firmware to guarantee the security of your private keys. KeepKey has supported GoByte since firmware version TBA, released on TBA. Follow these instructions to begin using GoByte on your KeepKey device.

- Site: <https://www.keepkey.com>
- Review: <https://coincentral.com/keepkey-wallet-review>
- Shop: <https://keepkey.myshopify.com/>
- Product video: <https://vimeo.com/133811189>

It is also possible to operate a GoByte masternode using your KeepKey. See [here](#) for details.

Ledger



Founded in 2014, French startup **Ledger** markets enterprise and consumer blockchain security solutions, including the €58 **Ledger Nano S** and upcoming **Ledger Blue**. Ledger Nano S has supported GoByte since TBA and firmware version TBA. Follow *these instructions* to add GoByte support to the device.

- Site: <https://www.ledger.com>
- Review: <https://www.gobyteforcenews.com/ledger-nano-s-review>
- Shop: <https://www.ledger.com/collections/all-products>

Product video:

It is also possible to operate a GoByte masternode using your Ledger. See [here](#) for details.

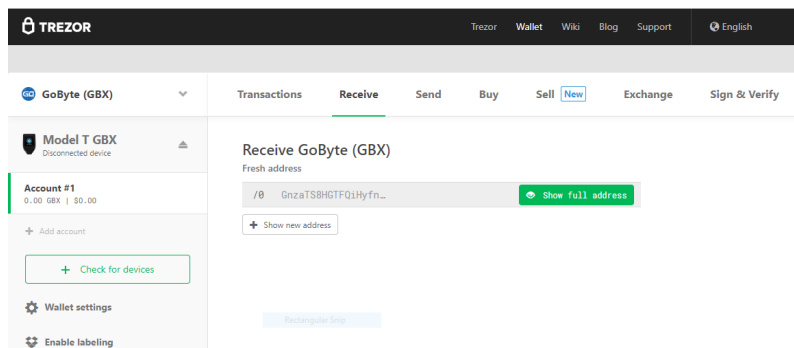


Fig. 138: Trezor Web Wallet

1.6.7 Third Party Wallets

These wallets have been developed by external developers to support GoByte. Many third party wallets support multiple different cryptocurrencies at the same time, or integrate instant cryptocurrency exchanges.

Introduction

The GoByte protocol and many GoByte products such as GoByte Core and the mobile wallets are entirely open source, which makes it easy for third parties to integrate GoByte with their existing cryptocurrency wallet solutions. This section describes some of the third party wallets available and the functions they offer. Please note that GoByte does not provide support for any of these wallets, and any listing here should not be considered an endorsement or recommendation. Contact the software vendor for support.

Coinomi

<https://coinomi.com>



Coinomi is an open-source multi-currency mobile wallet available for iOS and Android. Your private keys never leave your device, and strong wallet encryption guarantees that your funds are always under your control only. Instant exchange is available directly in the wallet through ShapeShift and Changelly integrations.

Download



Coinomi desktop wallets are available from <https://www.coinomi.com/downloads> for Linux, macOS and Windows. Coinomi is also available from the [Google Play Store](#) for Android and the [Apple App Store](#) for iOS.

Documentation

Coinomi offers detailed documentation of all functions at <https://coinomi.freshdesk.com>

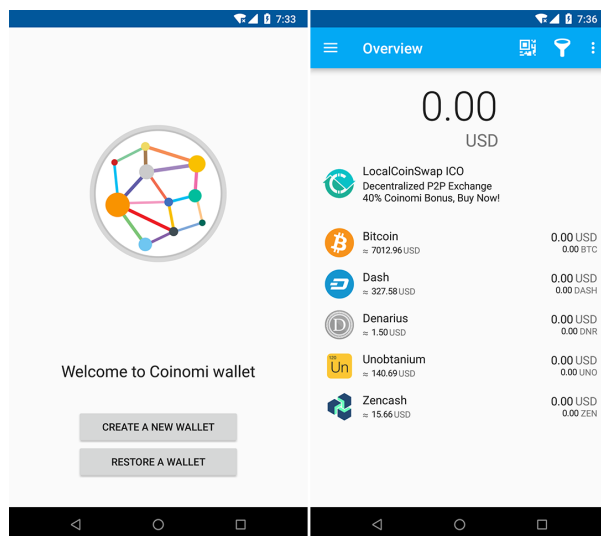


Fig. 139: Coinomi wallet running on Android

Edge

<https://edge.app>



Edge is a secure multi-currency wallet for iOS and Android. It offers a unique login system to store your encrypted HD seed on the cloud while still performing all sensitive operations requiring a private key on your device. Edge is fast and simple to use, allowing you to scan QR codes and sign transactions using your fingerprint ID or a simple PIN code. ShapeShift is also integrated to facilitate exchange between different cryptocurrencies.

Installation



Edge is available from the [Google Play Store](#) for Android and the [Apple App Store](#) for iOS.

Documentation

Edge offers detailed documentation of all functions at <https://support.edge.app> and a few quick links are also collected here:

- [Getting started](#)
- [How do I create a new wallet?](#)
- [How do I send money?](#)
- [How do I receive money into my account?](#)

Exodus

<http://www.exodus.io>



The Exodus wallet features an engaging visual design and can simultaneously store multiple currencies. It is available for Windows, Mac, Linux and iOS. It is also fully integrated with Shapeshift to offer exchange between the different currencies.

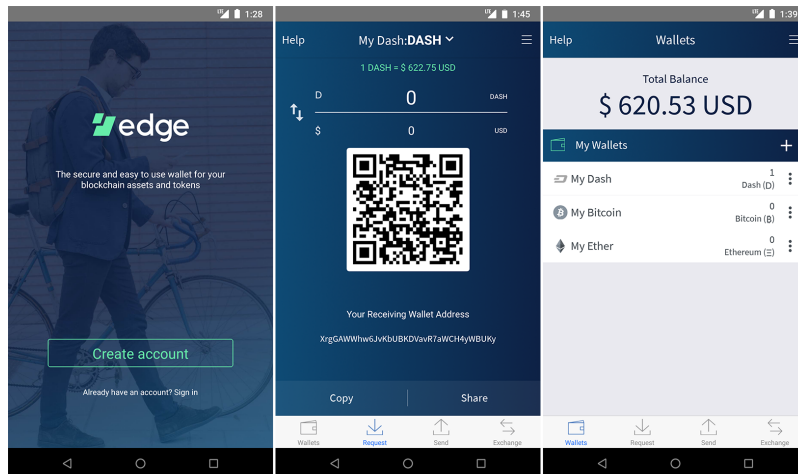


Fig. 140: Edge Welcome, Receive GoByte and Balance screens

Installation

All Exodus releases are available from <https://www.exodus.io/download> - simply download and install the appropriate package for your system. Exodus is also available from the [Apple App Store](#) for iOS.

Documentation

Exodus offers detailed documentation of all functions at <http://support.exodus.io> and a few quick links are also collected here:

- [How do I install Exodus?](#)
- [How do I get started with Exodus?](#)

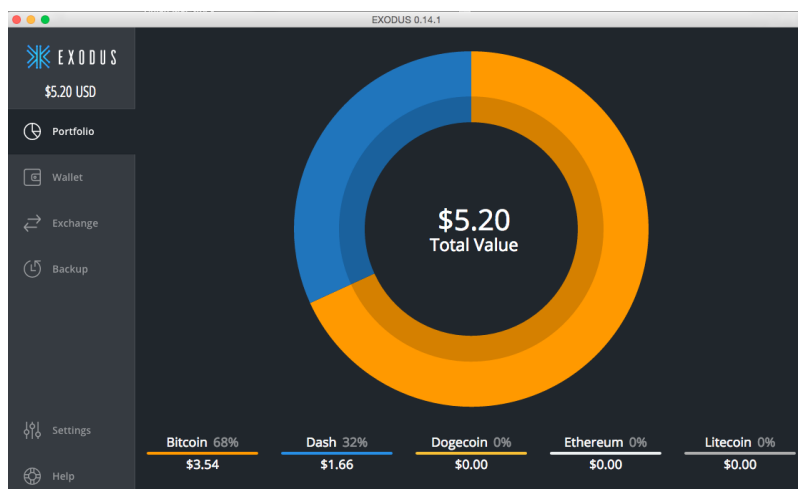


Fig. 141: Exodus wallet Portfolio screen

Guarda

<https://guarda.co>



Guarda offers an entire blockchain ecosystem consisting of desktop, web and mobile wallets, OTC crypto sales and instant crypto exchange. GoByte is supported throughout the ecosystem, making it an easy and convenient way for new users to get started. All keys are held by the user, ensuring the safety of your funds.

Installation



Guarda desktop wallets are available from <https://guarda.co/desktop> for Linux, macOS and Windows, or you can use web wallet at <https://guarda.co/app> to create new or restore existing wallets. Guarda is also available from the [Google Play Store](#) for Android and the [Apple App Store](#) for iOS.

Documentation

Guarda offers detailed documentation of all functions at <https://guarda.freshdesk.com> and a few quick links are also collected here:

- [How to create a wallet?](#)
- [What is Guarda Exchange?](#)

Jaxx

<https://jaxx.io>



Jaxx supports multiple currencies in one wallet, including GoByte. It is available for almost all platforms including Android, iOS, macOS, Windows, Linux and also as a Chrome extension. Jaxx is open source software.

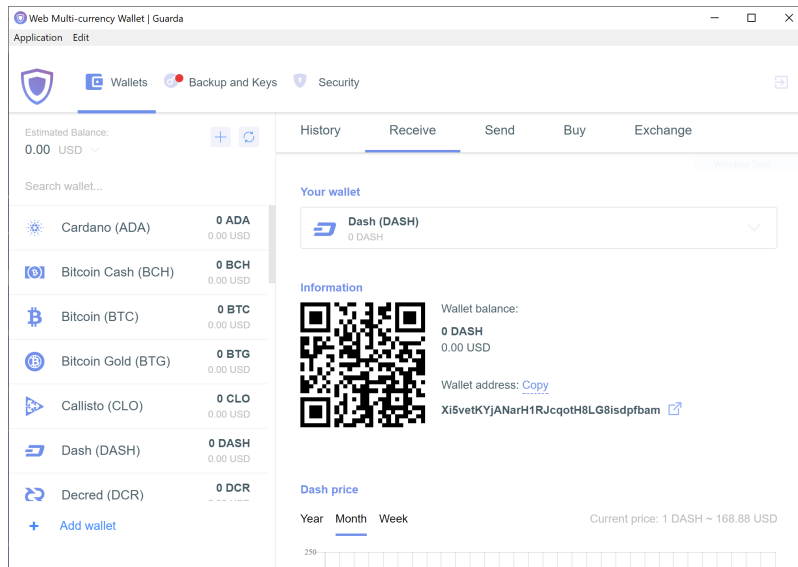


Fig. 142: Guarda wallet

Installation



All Jaxx releases are available from <https://jaxx.io/downloads.html> - simply download and install the appropriate package for your system. Jaxx is also available from the Google Play Store for Android and the Apple App Store for iOS.

Documentation

Jaxx offers detailed documentation of all functions at <https://decentral.zendesk.com> and a few quick links are also collected here:

- [Getting started](#)
- [How do I send currency?](#)
- [How do I receive currency?](#)

Magnum

<https://magnumwallet.co>



Fig. 143: Jaxx wallet running on various devices



Magnum is a multi-currency web and mobile wallet with support for 100+ cryptocurrencies, including GoByte. The wallet integrates Changelly for in-app exchange and supports staking, delegation and airdrop functions. Magnum focuses on providing a simple and secure interface to store and interact with your digital assets.

Installation



Magnum is available from the [Google Play Store](#) for Android.

Documentation

Join the [Magnum Telegram](#) group for Magnum support.

Mobi

<https://www.mobi.me>

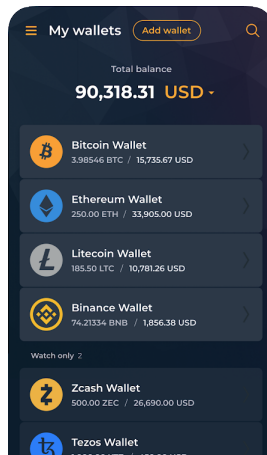


Fig. 144: Magnum wallet running on Android

wallets/img/mobi.png

Mobi is a multi-currency mobile wallet linked to your phone number. As a hosted wallet, Mobi holds the private keys to your funds on your behalf, meaning you can restore your funds simply by receiving a text message and entering your PIN. However, you must trust Mobi to act responsibly with these private keys, and you will lose access to your funds if you lose access to your phone number. A web interface is also available, and you can use fiat currency to buy cryptocurrency in the app.

Installation



Mobi is available from the [Google Play Store](#) for Android and the [Apple App Store](#) for iOS.

Documentation

Read the [FAQ](#), join the [Mobi Telegram](#) group or send an email to support@mobi.me for support with Mobi.

Ownbit

<https://ownbit.io>

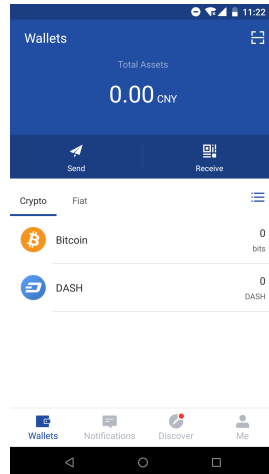


Fig. 145: Mobi wallet running on Android



Ownbit is a multi-currency and multi-signature capable mobile wallet with support for GoByte. It allows you to manage multiple wallets and contacts to facilitate easy transactions.

Installation



Ownbit is available from the [Google Play Store](#) for Android and the [Apple App Store](#) for iOS.

Documentation

Join the [Ownbit Telegram group](#) or send an email to hi@bitbill.com for support with Ownbit.

Spend

<https://www.spend.com>



Spend offers a multicurrency wallet for Android and iOS, which is also used to manage balance for the Spend Visa Card and loyalty program.

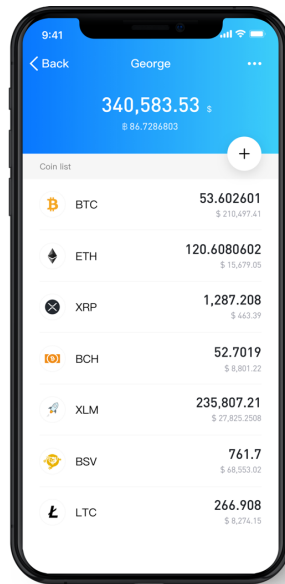


Fig. 146: Ownbit wallet

Installation



Spend is available from the [Google Play Store](#) for Android and the [Apple App Store](#) for iOS.

Documentation

Support for Spend is available at <https://help.spend.com>

Trust

<https://trustwallet.com>



Backed by [Binance](#), Trust wallet is a secure and intuitive multi-currency mobile wallet with support for Dash, Bitcoin, Ethereum and a wide range of tokens and DApps.

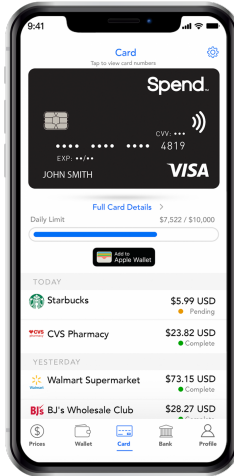


Fig. 147: Spend wallet running on iOS

Installation



Trust is available from the [Google Play Store](#) for Android and the [Apple App Store](#) for iOS.

Documentation

See the [Help Center](#) or join the [Trust Telegram group](#) for support with Trust.

1.6.8 Web Wallets

Web wallets are services which keep a GoByte balance for you, while maintaining control of the private keys on your behalf. Any GoByte stored on *exchanges* falls under this category, but there are also some services able to store GoByte for you through simple Google/Facebook login systems. Be extremely careful with web storage, as your GoByte is only as secure as the reputation of the company storing it for you. A particular exception is [MyGoByteWallet.org](#), which provides a secure web interface to the GoByte blockchain while leaving you with full control of your private keys.

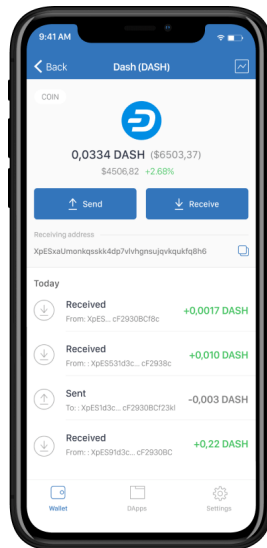


Fig. 148: Trust wallet running on iOS

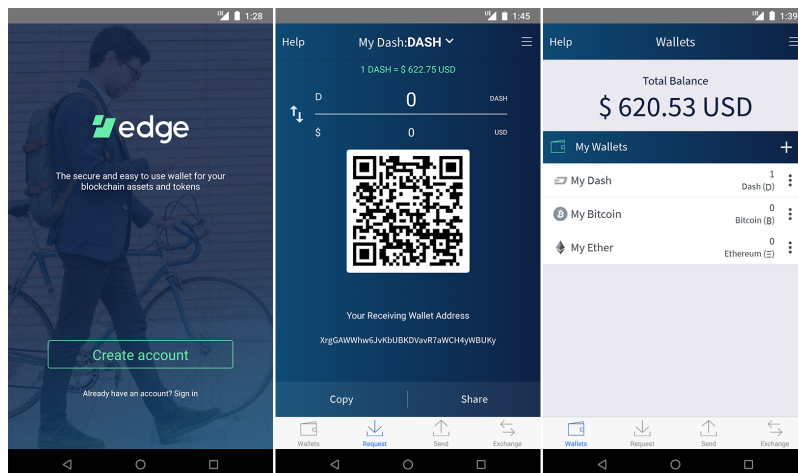
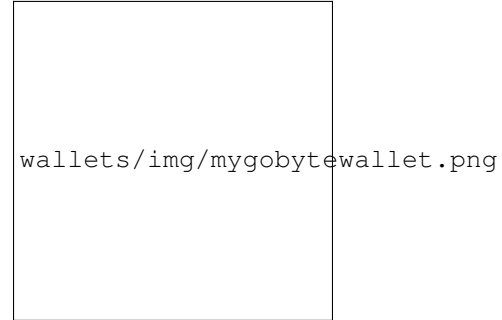


Fig. 149: Edge Wallet

MyGoByteWallet



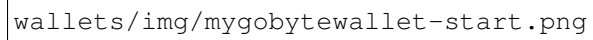
[MyGoByteWallet](#) is a web interface to the GoByte blockchain, inspired by [MyEtherWallet](#) and created by [DeltaEngine.net](#). It is explicitly not an online wallet, meaning you maintain control over your private keys at all times. Unlike many other light wallets, MyGoByteWallet also supports advanced GoByte features such as InstantSend and PrivateSend. The project is non-profit, open source and free to use. You can load a wallet and transact in a variety of wallet formats:

- Keystore wallet (file-based)
- Ledger hardware wallet
- Trezor hardware wallet
- Private key
- BIP39/44 HD recovery phrase (coming soon)
- BIP32 HD recovery phrase (coming soon)

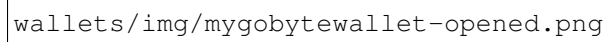
Please note that web wallets may not be as secure as alternatives such as hardware wallets. Be aware of the risk of storing large amounts of GoByte in keyfile wallets, since they are an easier target to attack than mobile or hardware wallets.

MyGoByteWallet offers complete and detailed documentation for all functions.

- [Getting started](#)
- [How to Create a Wallet via Keystore file](#)
- [Using the Ledger Hardware Wallet on MyGoByteWallet](#)
- [Using the Trezor Hardware Wallet on MyGoByteWallet](#)
- [How to does GBX InstantSend work on MyGoByteWallet?](#)
- [How to does GBX PrivateSend work on MyGoByteWallet?](#)

A screenshot of the GoByte wallet application in its initial state, showing the start screen.

wallets/img/mygobytewallet-start.png

A screenshot of the GoByte wallet application after it has been opened, showing the main interface.

wallets/img/mygobytewallet-opened.png

Magnum Wallet



Magnum Wallet is a web wallet with support for 20+ cryptocurrencies, including GoByte. The wallet stores encrypted private keys in a simple downloadable text file, which can only be decrypted with the users password. Hardware wallets are also supported, and in-wallet staking and exchange functionality is available. Magnum Wallet presents a clean and simple interface, allowing users to transact quickly and easily in GoByte.

Please note that web wallets may not be as secure as alternatives such as hardware wallets. Be aware of the risk of storing large amounts of GoByte in keyfile wallets, since they are an easier target to attack than mobile or hardware wallets.

Support for Magnum Wallet is available at support@magnumwallet.co.

- [Magnum GoByte Wallet](#)

wallets/img/magnum-wallet.png

1.6.9 Text Wallets

Text wallets (or SMS wallets) allow users without smartphones or internet access to transact in GoByte using text messages on simple feature phones. Innovative shortcodes, usually in collaboration with national mobile service providers, make it relatively simple to create transactions to both send and receive GoByte.



Fig. 150: My GoByte Wallet

Introduction

Text message (SMS) wallets allow you to easily transact in GoByte using a simple feature phone. An internet connection is not required. Because text wallets require access to text messages, they generally only support specific regions. See below for details.

GoByteText

<https://gobytext.io>



wallets/img/gobytext.png

GoByteText is a service available in (Not yet developed) to allow users to transact in GoByte using text messages. The only fees are the cost of a standard SMS, incurred by the network operator. Users can send GoByte to innovative shortcodes to securely confirm transactions.

Instructions

Simply text CREATE to the GoByteText phone number to get started. Further instructions and links will appear here once the project is ready for mass market.

1.6.10 Wallet Guides

Documentation in this section describes common tasks and questions relating to all wallets.

Wallet Recovery

Long-time users of cryptocurrency sometimes find old wallet files on USB drives or cloud storage that they have forgotten about. Others may have a backup, but can't remember the software they used to create it, or have forgotten the password. Other users may have an old version of GoByte Core that no longer works because the network has upgraded. This documentation is intended to help these users restore access to their funds.

Determining the backup format

The first step is to determine the format of your backup. In most cases, this will either be a file, probably named *wallet.dat*, or a phrase of words. In some cases, you may have stored the private key for a GoByte address directly. The following list shows the possibilities and methods to restore your wallet in order of probability.

- Backup is stored in an older version of GoByte Core that no longer works
 - Follow instructions for restoring wallet files using *GoByte Core*
- Backup is a file



Fig. 151: GoByte CoinText

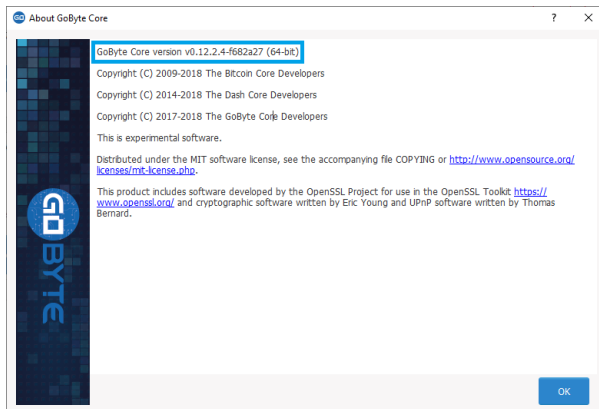
- If file name is similar to wallet.dat, try to restore using *GoByte Core*
- If file name is similar to gobyte-wallet-backup or includes the word ‘mobile’, try to restore using *GoByte Wallet for Android*
- Backup is a phrase of words
 - If 12 words long, try to restore using *GoByte Electrum wallet* or GoByte wallet for *Android* or *iOS*, depending what you used to create the backup
 - If 13 words long, try to restore using *GoByte Electrum wallet*
 - If 12, 18 or 24 or 25 words long, try to restore with the *hardware wallet* you used to create the recovery phrase
- Backup is a long string of random characters or a QR code
 - If 34 characters long and starting with X, this is a public address and cannot be used to restore access to lost funds. You need the private key.
- If 51 characters long and starting with 7, this is a *private key in WIF*, import using GoByte Core
- If 58 characters long and starting with 6P, this is a *BIP38 encrypted private key*, decrypt using paper wallet then import using GoByte Core

Once you have determined your backup format, follow the links to view the restore guide for that format.

File Backups

GoByte Core

One of the most common wallet backup formats is a *wallet.dat* file from GoByte Core wallet. Before you begin, make absolutely sure that you have a copy of this file stored somewhere safe in case the restore process accidentally corrupts your wallet file! In most cases, *wallet.dat* backups will also be protected by a password, which you will need to know to regain access to your GoByte funds. If you already have GoByte Core installed, first ensure it has been updated to the latest version by clicking **Help > About GoByte Core**. Compare this with the latest available version of *GoByte Core on the website* as follows:



Update GoByte Core to the latest version according to the *installation instructions*. If you have only a wallet file and no existing installation of GoByte Core, simply install GoByte Core according to the *installation instructions* and start it once to create the GoByteCore folder. Then close GoByte Core and copy the *wallet.dat* file you want to restore to the GoByteCore folder in the location specified below, replacing or renaming the existing file.



Fig. 152: Comparing the installed version of GoByte Core with the latest version available on the website

Platform	Path to data folder	How to navigate
Linux	~/	Go to your home folder and press Ctrl+H to show hidden files, then open <code>.gobytecore</code>
macOS	~/Library/Application Support/	Press Shift + Command + G , type <code>~/Library/Application Support</code> , then open “GoByteCore”
Windows	%APPDATA%	Press Windows Key + R and type <code>%APPDATA%</code> , then open GoByteCore

If your existing version of GoByte Core is older than v0.12.1.x, you may need to rename your data folder from GoByte to GoByteCore.

To repair a broken installation, navigate to the GoByteCore folder and delete all `.log` and `.dat` files except `wallet.dat`. The following files can be safely deleted:

- `banlist.dat`
- `budget.dat`
- `db.log`
- `debug.log`
- `fee_estimates.dat`
- `governance.dat`
- `mncache.dat`
- `mnpayments.dat`
- `netfulfilled.dat`
- `peers.dat`

Leave `.conf` files and the folders (such as `backups`, `blocks`, `chainstate`, etc.) intact, since they will help you get started faster by providing a copy of the blockchain and your settings.

Now open GoByte Core and wait for blockchain synchronization to complete. Your wallet will be restored/updated and all balances should be displayed. You should ensure you have the correct password by trying to unlock your wallet from **Settings > Unlock Wallet** to make sure you can actually create transactions using your balances. If you have any problems with your balance not appearing, try to force a rescan of the blockchain by going to **Tools > Wallet Repair** and selecting **Rescan blockchain files**. **Rebuild index** may also help. GoByte Core will restart and perform a full scan of the blockchain.

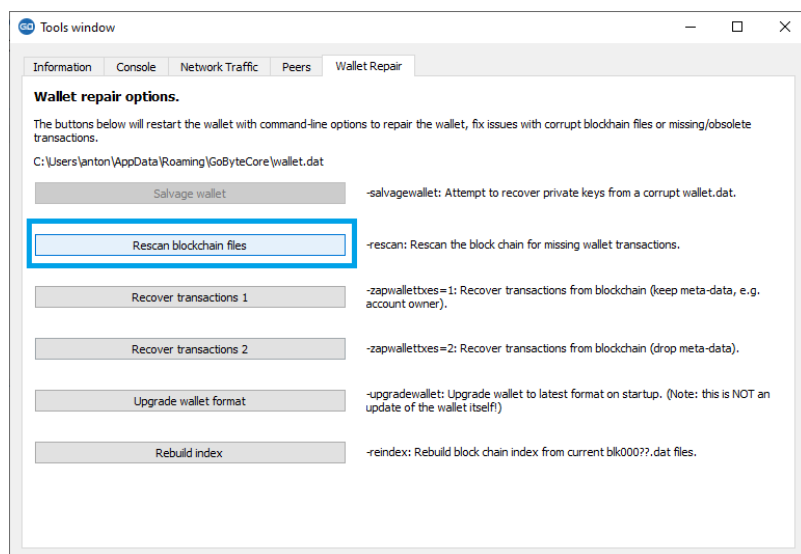


Fig. 153: Forcing GoByte Core to rescan the blockchain

At this stage, recovery is complete and you should make another backup using **File > Backup Wallet** or following the instructions [here](#). If you have any further problems, try asking on the [forum](#), [Reddit](#) or the [#gobyte-support-desk](#) channel at [GoByte Official Discord](#).

GoByte Android

Similar to GoByte Core wallet, GoByte Wallet for Android can back up your wallet to a file. To restore this wallet on another device, simply copy the backup file to the /Downloads folder of your device using either a computer connected by USB or a file manager app on the device. Ensure your GoByte wallet is fully updated in the Play Store, then open GoByte. If you have an existing balance, either make another backup or transfer it to an external address, because restoring a wallet will replace your existing wallet!

Click the menu button in the top left corner, select **Safety > Restore** wallet and select the appropriate file from the list. Enter your password and click **Restore**. This may take some time, and your balance will be displayed when complete.



Restoring a file backup using GoByte Wallet for Android

Recovery Phrases

If you have a 12-word phrase and feel certain your backup was made on an iOS or Android mobile device, follow these instructions.

12-word phrase on Android

Ensure your GoByte wallet is fully updated in the Play Store, then open GoByte. If you have an existing balance, either make another backup or transfer it to an external address, because restoring a wallet will replace your existing wallet! Click the menu button in the top left corner, select **Safety > Restore from recovery phrase** and enter your 12-word phrase.



Restoring a 12-word recovery phrase using GoByte wallet for Android

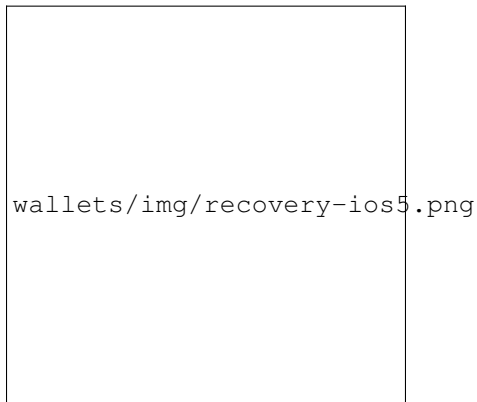
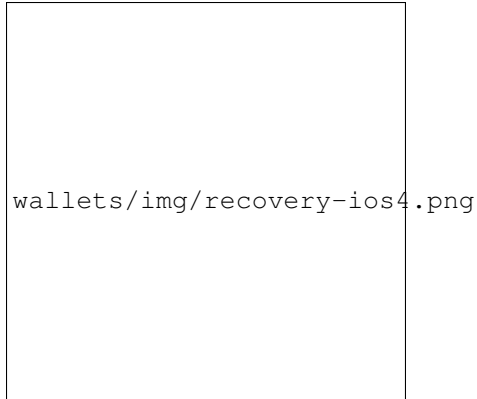
12-word phrase on iOS

Ensure your GoByte wallet is fully updated in the App Store, then open GoByte. If this is the first time you are opening the app, you can enter your recovery phrase directly by selecting **Recover wallet** on the start screen. If you

have an existing balance, either make another backup or transfer it to an external address, because restoring a wallet will replace your existing wallet!

Click the menu button in the top left corner, select **Settings > Start/recover another wallet**. Enter your current wallet recovery phrase, then the app will reset and you will see the option to **Recover wallet** again.

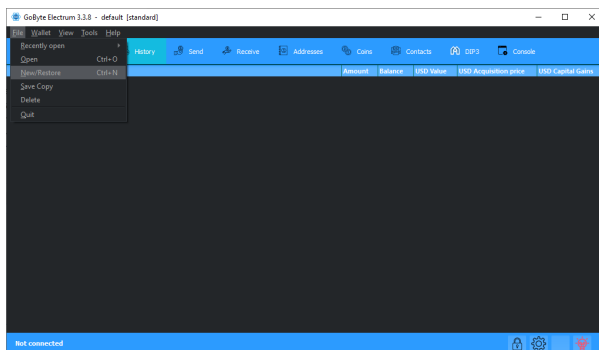


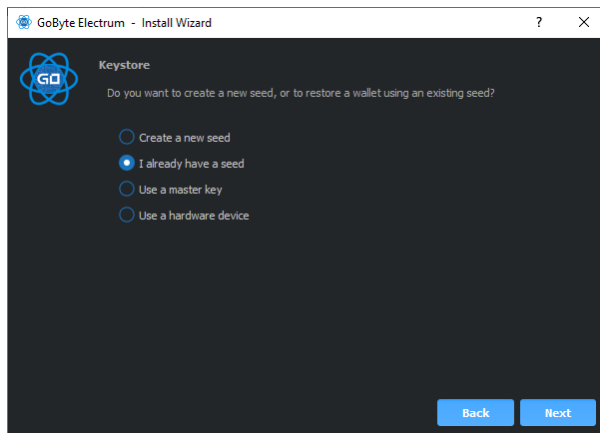
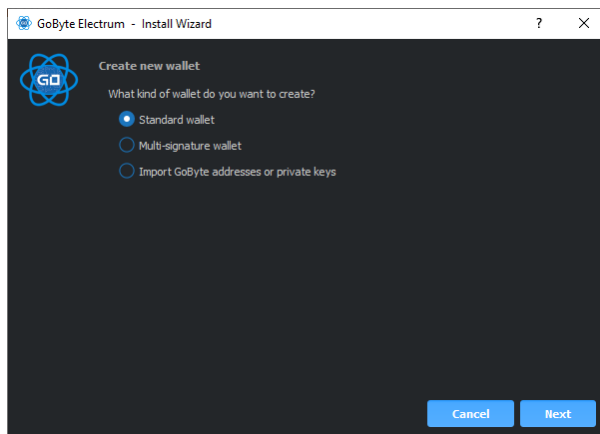
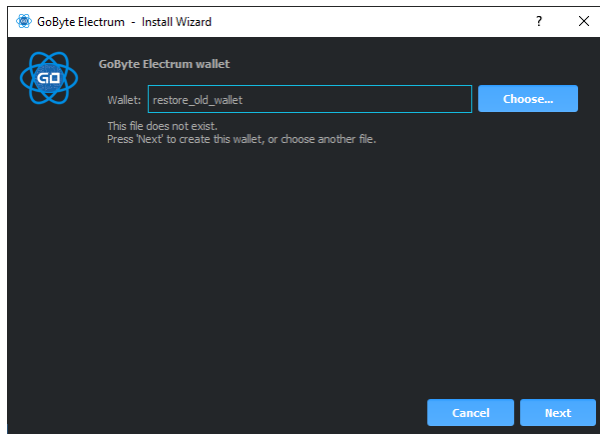


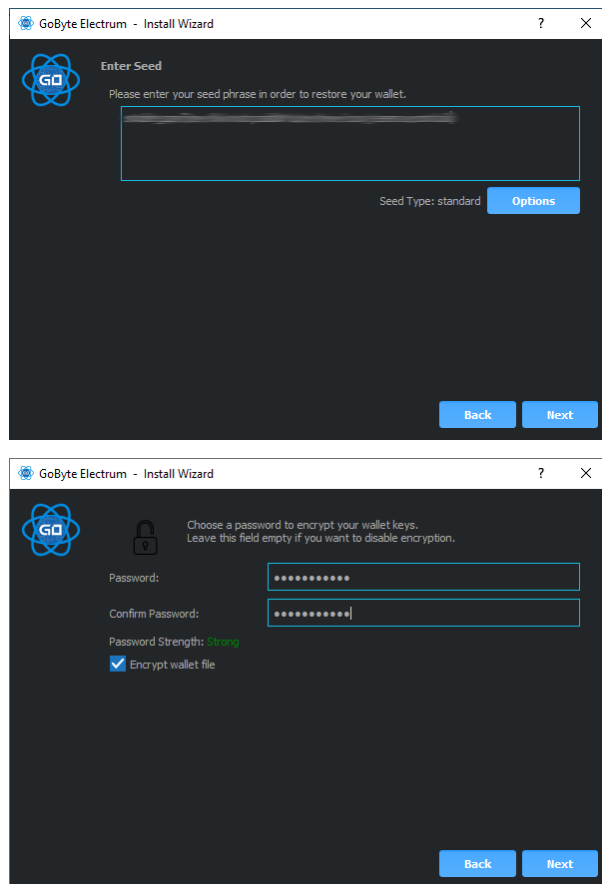
Restoring a 12-word recovery phrase using GoByte wallet for iOS

12/13-word phrase on GoByte Electrum

Ensure you are using the latest version of GoByte Electrum according to the installation instructions [here](#). GoByte Electrum supports multiple simultaneous wallets, so you can safely restore to a new wallet file without losing your old wallet. Click **File > New/Restore** and enter a file name to store your new wallet. Then select **I already have a seed** and enter your 12/13-word recovery phrase. Enter a new password for your wallet and click **Next** to recover your addresses from the recovery phrase.







Restoring a 12-word recovery phrase using GoByte Electrum

Hardware wallet recovery phrases

If your 12, 18 or 24-word recovery phrase was generated by a hardware wallet, follow these instructions:

- [KeepKey](#)
- [Ledger Nano S](#)
- [Trezor](#)

Restoring an iOS wallet in GoByte Electrum

You can use your GoByte iOS recovery phrase with GoByte Electrum to recover funds if you lose access to your iOS device for any reason. However, since the wallet derivation paths are not identical, the process only works in one direction, meaning it is not possible to restore a GoByte Electrum wallet using the GoByte iOS wallet. Also, because the import process uses an xprv key rather than the recovery phrase directly, it will not be possible to display the recovery phrase in GoByte Electrum. It is therefore recommended to move the funds (either to a standard GoByte Electrum wallet or some other wallet) once recovery is successful to ensure that standard backup procedures work as expected.

Recovery takes place in two steps. First, we will convert the GoByte iOS recovery phrase into an xprv key. In the second step, we will import the xprv key into GoByte Electrum.

Retrieving the correct GoByte iOS xprv key

Go to the [BIP39 Mnemonic Code Converter](#) page. This is a useful tool for manipulating/displaying BIP32/39 seed data. If you are not comfortable performing this procedure online, an offline version is available by downloading the file described in [these instructions](#). Once the tool is loaded in your browser, complete the following steps:

1. Enter your 12 word seed phrase in the **BIP39 Mnemonic** field.
2. Leave **BIP39 Passphrase** blank.
3. Set coin to **GoByte**.
4. Under **Derivation Path**, click the **BIP44** tab.
5. Copy the value shown in **Account Extended Private Key**.

Importing the xprv key into GoByte Electrum

1. Open GoByte Electrum and click **File -> New/Restore**.
2. Type a name for your wallet.
3. Select **Standard wallet**.
4. Select **Use public or private keys**.
5. Paste in your value from **Account Extended Private Key**.
6. Optionally enter a password.

GoByte Electrum should now detect your GoByte iOS balance and you should have complete access to your funds. The seed phrase won't be available in GoByte Electrum, so you will just need to follow the steps above again if you want to restore this wallet from the recovery phrase again. It is recommended to send your funds to a new GoByte Electrum wallet instead and follow standard backup procedures.

Older versions of the GoByte iOS wallet used **BIP32** addresses under the `m/0'` derivation path. The wallet should migrate these funds over to BIP44 addresses during normal use, but some residual balance may be under this derivation path, so restoring the **BIP32 Extended Private Key** may be helpful in some situations. Please see [this forum thread](#) for further discussion on this process.

Private Keys

Most wallets offer a function to import an address from a private key, see the documentation for your wallet for specific instructions. While private keys can be stored in many ways, in this example we will work through the process of restoring a private key from a paper wallet using GoByte Core. If you only have a QR code and not the key, use a barcode scanning app ([Android](#) or [iOS](#)) to read the code first.

First, start GoByte Core and unlock your wallet by selecting **Settings > Unlock Wallet**. Enter your password, then open the debug console by selecting **Tools > Debug Console**. In the console, type the following, replacing the example private key with your key:

```
importprivkey 7rPQWnMrh3oWLtZrzt1zLRSCVyuBbwnt7fRBXPP2EwcPhtzXSzp
```

GoByte Core will rescan the blockchain for transactions involving the public address of this key and enter the transactions and balance in your wallet.

The private key must be in wallet import format (WIF). If your key is encrypted using BIP38 (key begins with 6P instead of 7), you must first decrypt it to view the key in WIF. To do so, go to <https://paper.gobyte.network/> and click

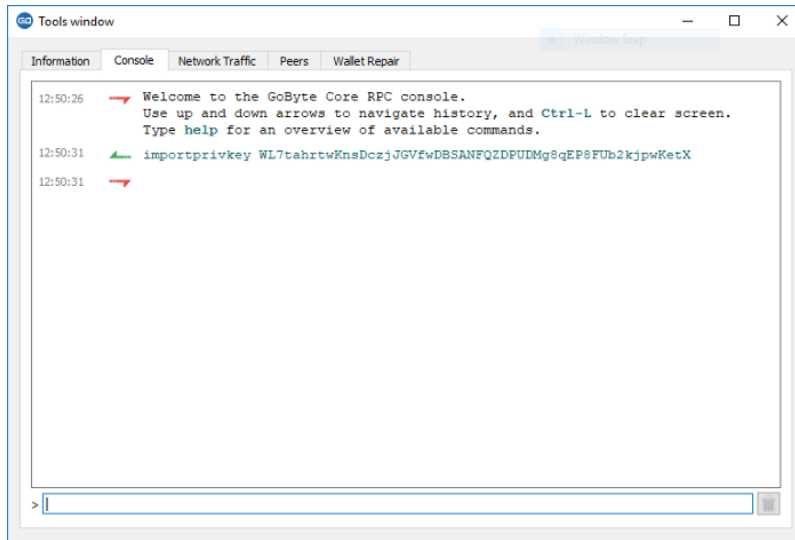


Fig. 154: Importing a private key using the debug console in GoByte Core wallet

Wallet Details. Enter the encrypted private key in the field and click **View Details**. You will be prompted for the password, and your keys will be decrypted. Find the key named **Private Key WIF** and import this into your wallet.



Private Key WIF
51 characters base58, starts with a '7'



7fRVn6jNwpUQksU87ARYpa8wkdG5eHGNX3qmnET4fUbf95rS8wB

Decrypting a BIP38 encrypted key to WIF for import in GoByte Core wallet

Forgotten Passwords

In most cases, if you selected a strong password and have forgotten or lost it, there is practically no hope of recovery. The encryption used by the GoByte wallets is extremely strong by design, and a well-chosen password should defeat

most brute force cracking attempts. If you can recall some details of the password, particularly its length or sequences of characters that may be included, then brute force password cracking techniques may be worth attempting. Several services exist to do this, or you can attempt it yourself. Because GoByte Core is based on Bitcoin Core, most approaches to apply brute force to crack a Bitcoin wallet will also work for GoByte wallets.

- [Wallet Recovery Services](#)
- [BTCRecover](#)

1.7 Earning and Spending

GoByte is designed from the ground up to function as digital cash. This documentation discusses how and where GoByte users can manage all of their personal finances using GoByte.

1.7.1 Earning

A range of services and businesses are available to convert your wage to and from GoByte as you receive it. It is of course easiest to receive payment from your employer in GoByte directly, however this may not always be an option. [PlusBit's GoByte POS App](#) allows you to invoice and receive payment from any employer, practically anywhere in the world, and have a percentage of your wage immediately converted to GoByte. You can then withdraw your wage to any GoByte wallet for spending or saving.

1.7.2 Spending

Merchant Directory

GoByte can be spent in hundreds of stores and services both online and in physical locations.



img/discover-gobyte.png

Discover GoByte lists businesses around the world accepting GoByte, sorted by category. It's easy to add your business to the list, and also features a short introduction for new GoByte users. The site is maintained by GoByte Nation, and you can [join the Telegram group here](#).

- [Discover GoByte](#)
- [GoByte Merchants](#)

Debit Cards

Debit cards work by prepaying in GoByte to load the account, then withdrawing cash from an ATM or spending online or anywhere debit/credit cards are supported. The GoByte is either exchanged at the time of purchase or in advance.

The rapidly evolving approach to regulation of cryptocurrencies such as GoByte and instant exchange solutions such as ShapeShift means that availability of debit cards cannot be guaranteed in any or all jurisdictions. Check with the following providers for updates on the availability of GoByte debit cards.

Disclaimer: This list is provided for informational purposes only. GoByte Core is not liable for any funds transmitted in error to these providers, or for the accuracy of information on this page.

The logo for Spend, featuring the word "Spend" in a bold, blue, sans-serif font, with a small "TM" trademark symbol to the right.

Spend

<https://www.spend.com>

Spend offers Simple, Preferred and Black Visa cards accepted at over 40 million locations worldwide. Linked

with the Spend Wallet, the system allows you to buy or deposit GoByte, which is then converted to the appropriate local fiat currency and loaded on the Spend Visa Card for use in purchase and ATM withdrawals.



Crypto.com

<https://www.crypto.com>

In the Crypto.com Wallet & Card App, users can purchase GBX at true cost with no fees - with credit card and bank transfer both supported. Holders of Crypto.com's MCO Visa card can also use GBX, making it easy to convert cryptocurrencies into fiat currencies and spend at over 40 million merchants globally.



PolisPay

<https://polispay.com>

The PolisPay Card is one of the fastest, easiest ways to turn your GoByte into fiat currency. You can use your PolisPay Card for online shopping and at any brick and mortar retailer that accepts MasterCard® debit cards. You can also withdraw cash at any MasterCard®-compatible ATM.

1.7.3 Tax

Taxation law is different depending on where you qualify as a resident for tax purposes. The following services are available to help you calculate your tax obligations.

- <https://www.node40.com>
- <https://gobyte-taxes.herokuapp.com>
- <https://cointracking.info>
- <https://bitcoin.tax>

1.8 Getting Started

GoByte welcomes new merchants and supports integration through a standardised onboarding process. It's easy to begin accepting payments in GoByte and enjoy the following benefits:

- Settlement within seconds and clearance within minutes
- Ability to accept payments from any market around the world
- Irreversible transactions to prevent fraud
- Advanced privacy for both customers and merchants
- Lowest fees in the industry

A three-part course on why GoByte is a popular choice for payments and how integration takes place is available in English and Spanish on GoByteAcademy.com. To get started with an integration in your sales system, simply select an online or point of sale payment solution from the lists below. If you are unsure, GoCoin is a popular choice due its support for *InstantSend*, while CoinPayments supports the largest range of online shop software. Anypay is an incredibly simple solution for retail stores, and also supports InstantSend. Larger integrations may require some customisation. This documentation also describes the *administrative* and technical steps required to integrate various GoByte services.

Many merchants accept GoByte - check out [PepSHIP](#) or [Vault.Investments](#) for examples of what merchant integration can look like. Once you are up and running accepting GoByte, consider adding your business to the directory maintained at [Discover GoByte](#) for increased visibility.

1.8.1 Payment Processors

This section lists known payment processors supporting GoByte and the business platforms they support. Please conduct thorough research before choosing a payment provider to ensure your needs will be met.

Online Stores

Due to the wide range of platforms for online stores, the following table is intended to help you select an appropriate payment processor for your existing store.

	CoinPay-ments	Go-Coin	Pay-Bear	Coin-gate	Cryp-toWoo	GoURL Cop-Pay	MyCryp-toCheckout	CD-Pay
aMember Pro	✓							
Arastta	✓							
bbPress						✓		
Blesta	✓			✓				
BoxBilling	✓							
Drupal Commerce	✓							
Easy Digital Downloads	✓					✓	✓	
Ecwid	✓							
Give (Donations)						✓		
Hikashop	✓							
Jigoshop						✓		
Magento	✓	✓		✓				✓
MarketPress						✓		
NATS		✓						
nopCommerce		✓						
OpenCart	✓	✓	✓	✓				✓
osCommerce	✓	✓		✓				
PaidMembershipsPro						✓		
PrestaShop	✓	✓	✓	✓				
Shopify		✓						
Tomato Cart	✓							
Uberscart	✓	✓						
VirtueMart		✓		✓				
WHMCS	✓	✓		✓				
WooCommerce	✓	✓	✓	✓	✓	✓	✓	✓
WP eCommerce	✓					✓		
X-Cart	✓							
ZenCart	✓	✓		✓				

Point of Sale

A range of Point of Sale systems are available. Many function as an app or simple website serving a checkout interface and QR code generator, while others support custom features such as NFC or a rewards scheme. QR.cr, PlusBIT Payments and Anypay are supported by the community and are particularly widespread.

Name	App?	Web interface?	Hardware?	NFC?	Notes
34 Bytes			✓		Hardware terminal capable of printing receipts.
Alt36					Full stack system. Supports integration of suppliers and employees.
Anypay	✓	✓			Popular solution for smartphones with web interface and backend.
CDPay	✓				
CoinPayments	✓	✓			
CopPay		✓			
EletroPay			✓		POS device with ePaper display for unique QR codes and built-in receipt printer.
Festy				✓	NFC wristband payments for festivals.
GB CortexPay			✓		Professional hardware terminal with multiple payment options.
Paytomat					Token rewards for crypto payments.
QR.cr	✓	✓			Cheap solution with many features to use a mobile phone as a POS terminal.
QuikWallet	✓	✓			India only. Also supports SMS payment.
PlusBIT	✓				Available for Android, and iOS. 94 exchange rates supported

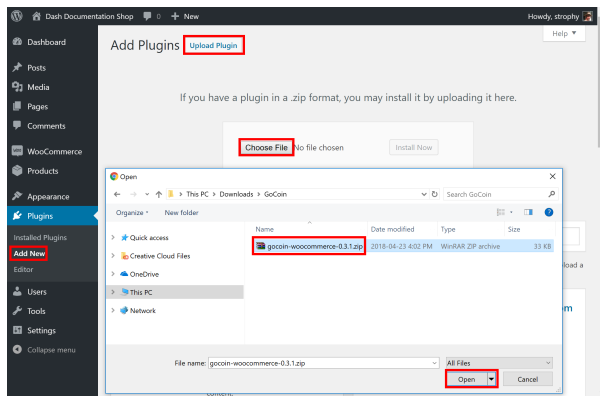
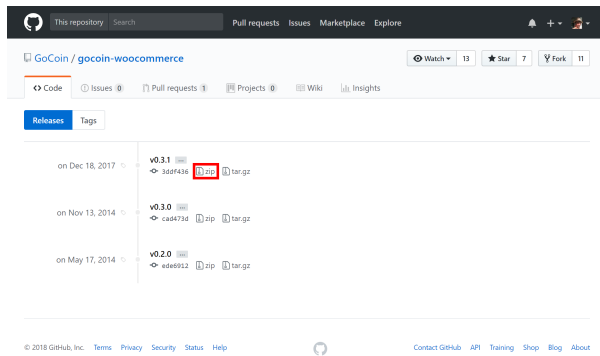
1.8.2 Installation Examples

This section contains worked examples of how to install, configure and process your first payment using the payment processors listed in this documentation.

WooCommerce and GoCoin

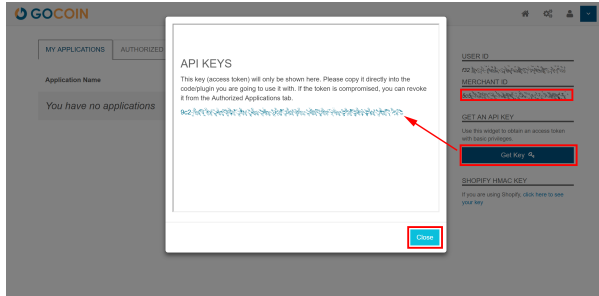
If your online store is built on WooCommerce, you can simply install GoCoin as an additional payment gateway and immediately begin accepting GoByte. This guide assumes you have already [installed Wordpress](#), [installed WooCommerce](#) and [created at least one product](#) in your store.

Go to the [gocoin-woocommerce GitHub Releases page](#) and download a zip file of the latest version of the plugin, as shown below. In your WordPress administration backend, select **Plugins -> Add New** and then click **Upload Plugin**. Click **Choose File** and select the file you just downloaded, then click **Install Now** and **Activate Plugin**.

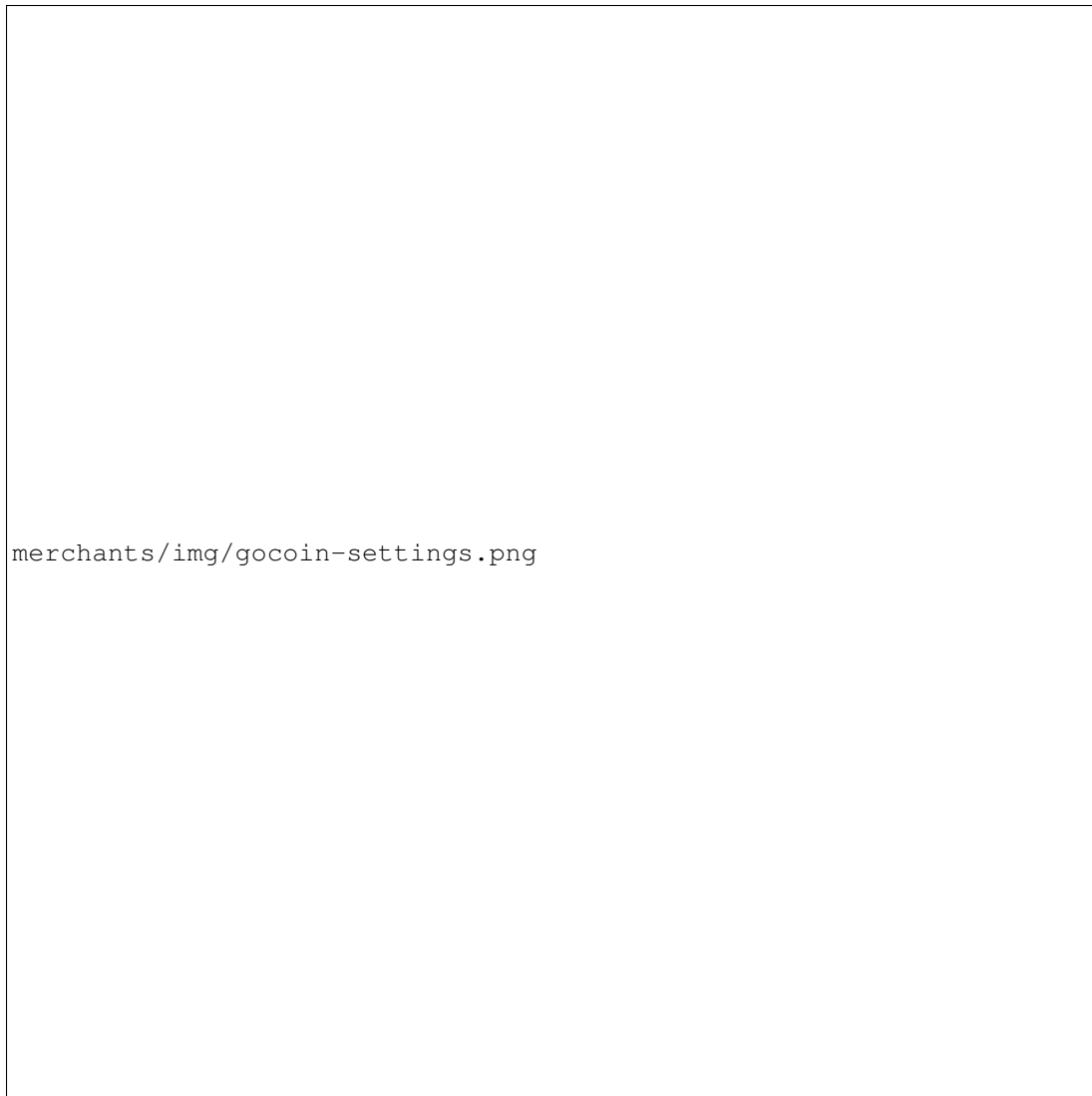


Next, go to the [GoCoin merchant sign up page](#) and create an account. Once you are logged in, go to **Preferences**, select **GoByte** and click **Add GBX Address** to add a payment withdrawal address. You will receive an email with a link to confirm the address. Next, go to **Developers** and copy the **Merchant ID** into a temporary text file. Next, click **Get Key** to display a valid API key. Copy this key into your temporary text file as well. Finally, you can optionally add a GoByte logo to your checkout by **Profile** section and clicking **Logo -> Upload**.

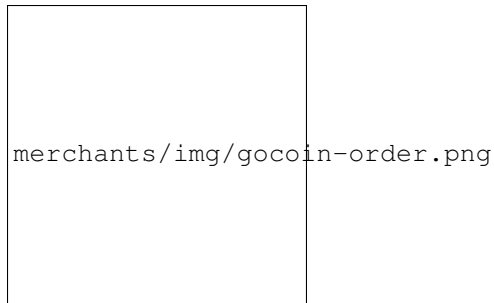




Back in the WordPress plugins section, click the **Settings** button for the WooCommerce plugin and navigate to **Check-out -> GoCoin** section. Ensure the GoCoin plugin is enabled here, then enter the **Merchant ID** and **API Key** in the appropriate fields as shown below, modifying the other fields as necessary. Click **Save changes** when you are ready.



Your customers will now see an option to pay with GoByte when completing the checkout process for an order. The payment will be processed by GoCoin, and you will receive emails detailing each purchase procedure. You can choose how often you want to withdraw your payments, to which GoByte address and various other options in the GoCoin administration section. See the [GoCoin Documentation](#) for more information.



WooCommerce and CoinPayments.net

If your online store is built on WooCommerce, you can simply install CoinPayments as an additional payment gateway and immediately begin accepting GoByte. This guide assumes you have already [installed Wordpress](#), [installed WooCommerce](#) and [created at least one product](#) in your store. A [video](#) of the process to install the CoinPayments payment processor is also available.

In your WordPress administration backend, select **Plugins -> Add New** and type “coinpayments.net” into the search box. A plugin named **CoinPayments.net Payment Gateway for WooCommerce** should appear. Click **Install Now** to install the plugin. Alternatively, you can [download the plugin](#) from the WordPress website as a zip file and upload it using the **Upload Plugin** button. Once the plugin is installed, click **Activate** to begin configuration.

Next, go to CoinPayments.net and [sign up](#) to create an account. Once you are logged in, go to **Account -> Coin Acceptance Settings** and enable GoByte, as well as optionally entering a withdrawal address. Next, go to **Account**



merchants/img/gocoin-paid.png

Fig. 155: Completing payment through the GoCoin payment processor

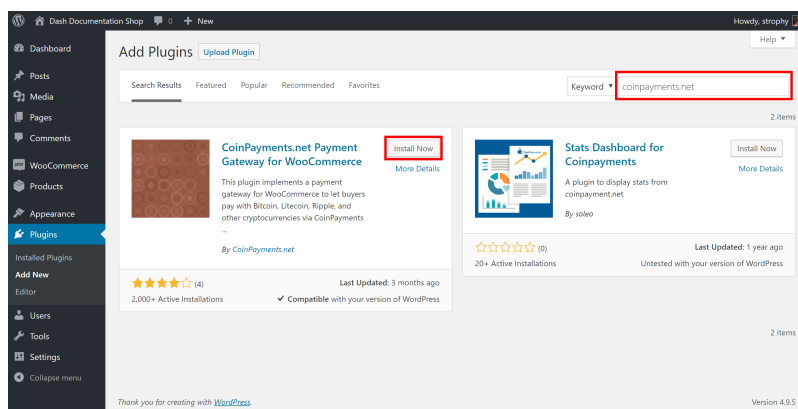



Fig. 156: Installing the CoinPayments.net WooCommerce plugin



merchants/img/coinpayments-confirm.png



merchants/img/coinpayments-scan.png

Point-of-Sale with Anypay

[Anypay.global](#) allows you to quickly start accepting point-of-sale payments in GoByte at a physical store. The service functions as a simple website that you load on any internet- connected and touch-enabled device, such as a smartphone or tablet.

Begin by registering an account with Anypay. You will be asked to specify an email address and password. Once you are signed in, you must add a GoByte payment withdrawal address.

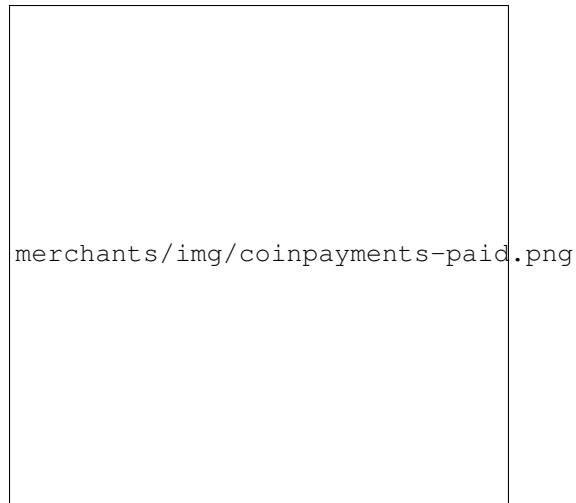
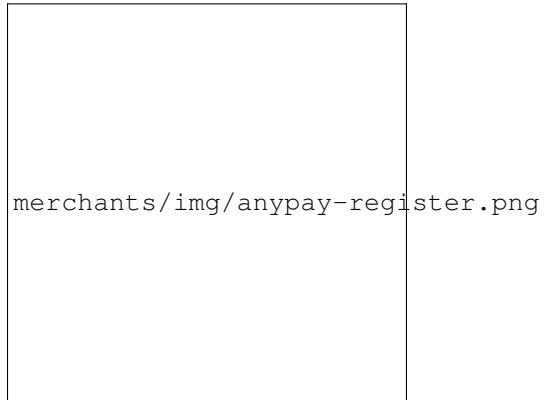


Fig. 158: Completing payment through the CoinPayments.net payment processor



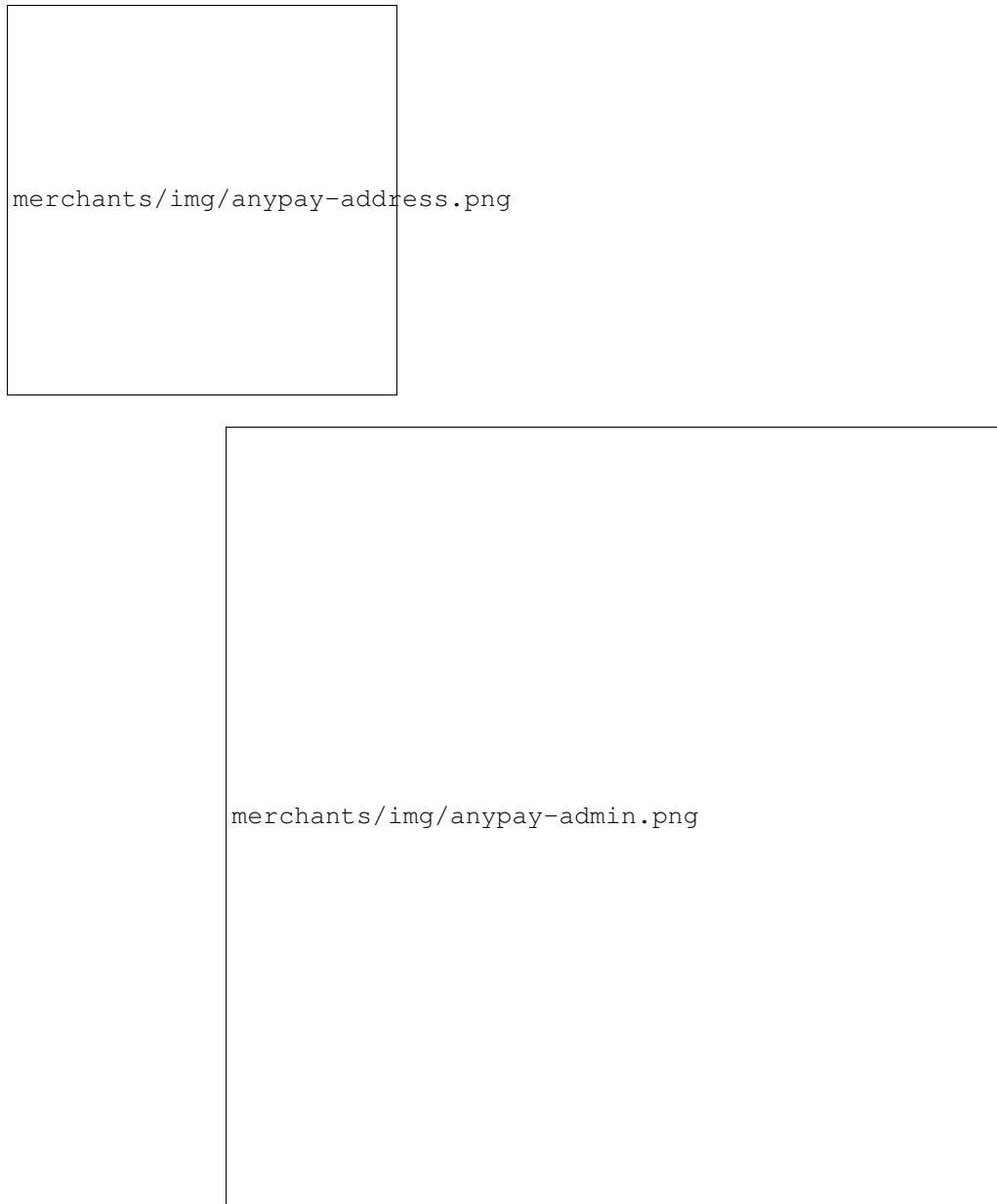
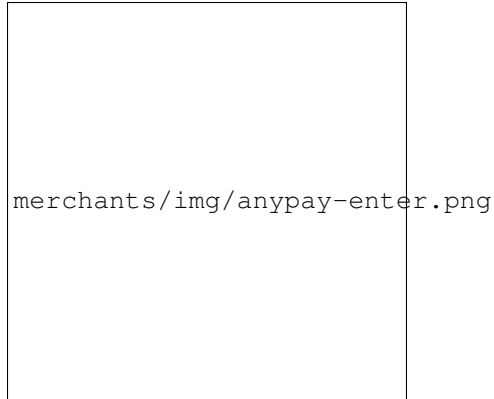


Fig. 159: Setting up Anypay

Once this has been set up, you can begin processing payments immediately. Simply log in to <https://pos.anypay.global> or tap **Merchant Point of Sale App** in the admin area using your device. A keypad will appear. Enter the invoice amount in USD or GBX and press the **COLLECT** button at the bottom of the screen. The app will generate a QR code for your customer to scan. Once payment is complete, you will be able to create a new invoice by tapping **Next Payment**, or view the status of your invoices by tapping the **menu button** in the top left corner of the keypad, or checking the **Invoices** section of the administration backend. Withdrawals are processed to the address you specified shortly after payment is complete.



Point-of-Sale with PlusBIT

PlusBIT GoByte POS allows you to quickly start accepting point-of-sale payments in GoByte at a physical store. The system works as an app, and is available for Android and iOS.

The project is an external terminal application for processing GoByte payments in brick and mortar stores. The merchant types the sale amount in their local currency (94 currencies supported), the application will generate a QR code sale for the proper amount of GoByte for the customer to scan. Then the terminal will provide feedback on the status of the payment (received, timed out, partial, instant send or regular).

To use PlusBIT, open the app on your device. If this is the first time you are using the app, you will need to specify a GoByte address to receive payments from the system, as well as your chosen fiat currency. You can change this information at any time from the menu. To generate a payment invoice, enter the amount in fiat currency. PlusBIT will generate a QR code containing your specified address and the requested amount, denominated in GoByte. The

customer scans the QR code, and the app will display a visual indication when payment is complete.

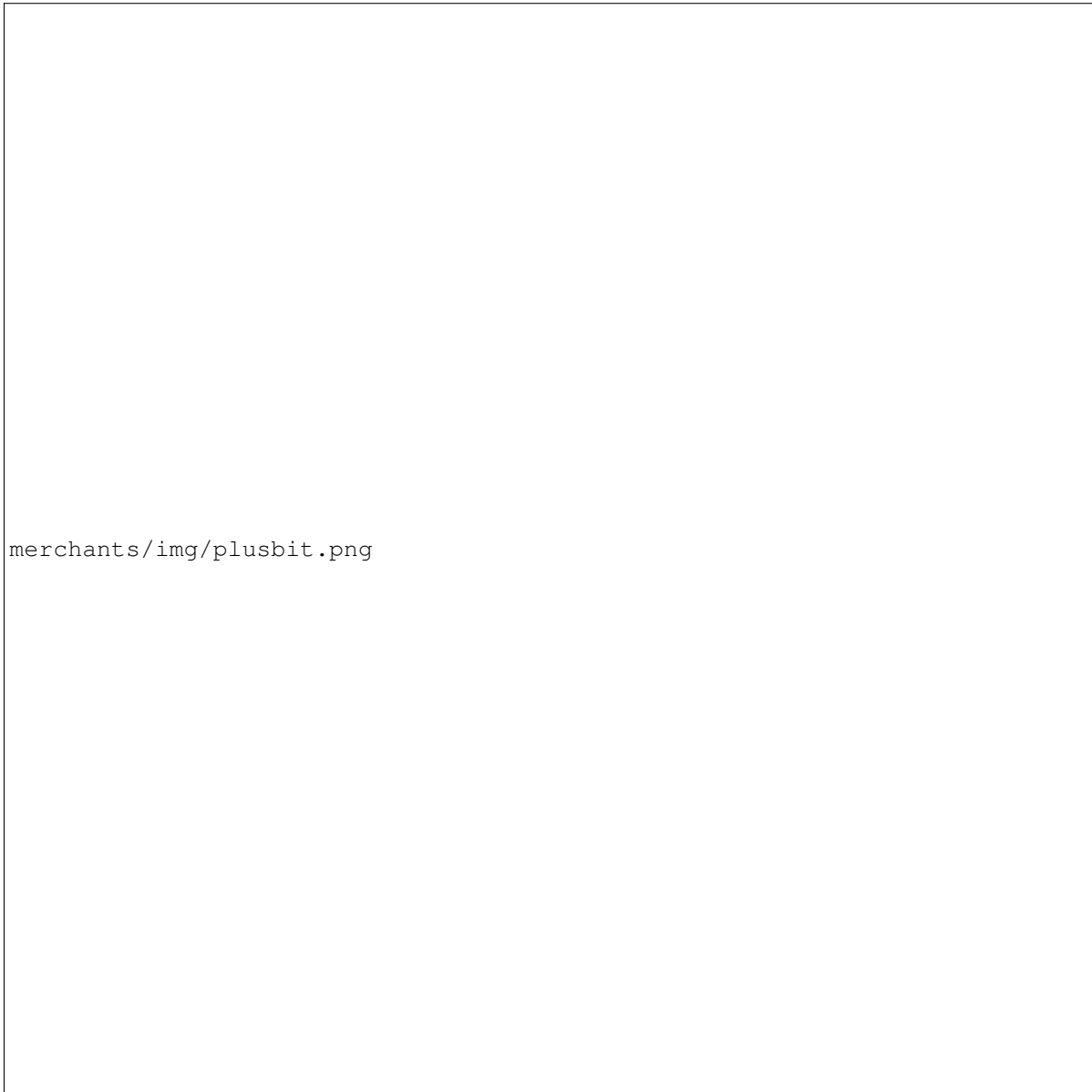


Fig. 160: Configuring and receiving payment using PlusBIT Payments

Payment systems like Anypay and PlusBIT can be integrated with your existing terminal and/or accounting software (such as Square Register, by recording sales invoiced in GoByte as an **Other Payment Type** in the system. This allows you to keep track of your GoByte income as easily as if you were accepting cash.

1.9 Administrative Processes

It's easy to get started integrating GoByte, but you will need to make some decisions about whether you plan to convert your income earned in GoByte into your local fiat currency, or if you prefer to hold some or all of it in GoByte. Most payment processors offer a range of fiat conversion options, although various fees and limits may be applicable.

1.9.1 Onboarding Process

New merchants typically go through the following steps when joining the GoByte ecosystem:

1. Set up a GoByte wallet
2. Identify an appropriate payment processor
3. Decide on how and when to convert funds
4. Implementation and testing
5. Release and marketing
6. Integration on [DiscoverGoByte](#)

1.9.2 Promoting GoByte

A wide range of ready-to-go visual products are available to help you promote GoByte as a payment method to your customers. This includes promotional graphics and stickers, fonts for consistent visual design and guidelines on how to use the GoByte visual identity. See the [Marketing](#) section for more information.

The reduced fees may also offer an additional incentive for your customers to pay with GoByte, particularly in businesses with high cash handling fees or where it is necessary to add a fee to process credit card transactions.

1.9.3 Currency Conversion

Cryptocurrency is a relatively recent development, and rapid development in the ecosystem coupled with various barriers to access and heavy trading mean that fiat-denominated value is subject to considerable fluctuation. As a merchant, you will need to make decisions about how much of your income taken in cryptocurrency should actually be held in cryptocurrency, and how much should be converted back to a fiat currency (such as USD) directly. Different payment processors offer different solutions to this problem.

Services such as [GoCoin](#) are able to convert a specified percentage of received payments into a range of fiat currencies for withdrawal. Others such as [CoinPayments](#) offer the ability to diversify your payments into a range of different cryptocurrencies, but require you to set up automatic withdrawals to an [exchange](#) for conversion to fiat currency. Finally, services such as [Uphold](#) allow you to convert your Crypto payments between various currencies and commodities very easily, and even offer automated investment services.

Note that these listing are not endorsements, and you must complete your own due diligence and/or seek advice from a tax and investment specialist before investing.

1.9.4 Legal considerations

Tax, legal and regulatory considerations may be applicable in some jurisdictions.

1.10 Governance

Decentralized Governance by Blockchain, or DGBB, is GoByte's attempt to solve two important problems in cryptocurrency: governance and funding. Governance in a decentralized project is difficult, because by definition there are no central authorities to make decisions for the project. In GoByte, such decisions are made by the network, that is, by the owners of masternodes. The DGBB system allows each masternode to vote once (yes/no/abstain) for each proposal. If a proposal passes, it can then be implemented (or not) by GoByte's developers. A key example is early in

2020, when GoByte’s Core Team submitted a proposal to the network asking to fund the ElectrumX wallet development. Within 24 hours, consensus had been reached to approve this change. Compare this to Bitcoin, where debate on the blocksize has been raging for nearly three years and has resulted in serious splits within the community and even forks to the Bitcoin blockchain.

The DGBB also provides a means for GoByte to fund its own development. While other projects have to depend on donations or premined endowments, GoByte uses 10% of the block reward to fund its own development. Every time a block is mined, 25% of the reward goes to the miner, 65% goes to a masternode, and the remaining 10% is not created until the end of the month. During the month, anybody can make a budget proposal to the network. If that proposal earns the net approval of at least 10% of the masternode network, then at the end of the month the requested amount will be paid out in a “superblock”. At that time, the block rewards that were not paid out (10% of each block) will be used to fund approved proposals. The network thus funds itself by reserving 10% of the block reward for budget projects.

In late 2016, IOHK prepared a detailed report on version 0.12.1 of the Dash governance system (adopted by GoByte), including formal analysis of weaknesses and areas for improvement. You can view the report [here](#).

You can learn more about GoByte Governance in the following sections:

1.10.1 Understanding GoByte Governance

One of the greatest challenges of building a cryptocurrency platform is ensuring you create a decentralized system of governance to manage, fund, maintain and expand the project. This key element has been absent in every major currency to date, so the natural response is to create a not-for-profit foundation that is tasked with maintaining the core protocol and promoting the coin, but is not really connected to the coin holders in any meaningful way. This approach has a few issues that have been made evident from the experience of older crypto currency platforms.

Current crypto foundations are not related to the currency itself by any mechanism that is included in the protocol and are not designed to outlive early adopters when they lose interest. The foundation then struggles to maintain funding until it implodes and core development of the protocol is left scrambling for funding or depending on charity that can’t be counted on and does not allow for proper budgeting and planning. Donations are also unfair to donors because there are always free riders that benefit from the effort done by others without contributing. Other projects have financed themselves by premining coins or running prelaunch sales, which is not a great solution either because control of the funds is centralized and at that stage it is impossible to quantify the future needs of the project.

Through the network of full nodes and the collateral requirement, GoByte already has a decentralized network of masternode operators that are heavily invested in the future of the currency, and that as a group can act as stewards of the core protocol development and promotion. We propose a decentralized management system based on the masternode voting mechanism. Masternode operators are not the only ones interested in the success of GoByte, but they are the most stable ones because, unlike miners, they can’t reuse their asset for any other purpose or coin.

In the budget system, a portion of the block reward is held in escrow by the network itself, in the name of the operators, to be executed in the development and expansion of the ecosystem according to the vote of the masternodes in different budget proposals. These funds are directed to supporting development and promotion of the coin. Masternode operators vote on specific budgets and projects to be funded, thus defining the direction the coin is taking. This is done in a completely transparent way through a public portal where new initiatives are proposed and masternodes can vote on them. Functioning like a decentralized Kickstarter or Lighthouse, the budget can be used for anything that creates value within the ecosystem.

This is a 100% decentralized system powered by the masternodes, where budgets are set and paid directly from the blockchain. The blockchain hires core developers in this way and introduces a new concept of paid blockchain contractors, where people work for and are directly compensated by the network, through the decentralized votes of all masternode operators. One advantage of this model is it can survive early adopters. If early masternode operators sell their coins, the new owner can set up a masternode and with it acquire the right to vote on the budgets and projects. This guarantees there is a working system of maintenance as people come and go, making the network capable of sustaining itself on its own without depending on specific actors.

Budgets and masternode voting

The system works as a decentralized voting mechanism set up in the rules governing the blockchain, where budgets for specific projects are proposed, then the masternodes as a whole vote on them. Each project, if it passes, is added to the total budget and paid directly from the blockchain to the person doing the work. This allows GoByte to hire core developers and pay them directly after approval of the work in a decentralized fashion.

A masternode votes on a proposal (technically a governance object on the blockchain) using the example command “masternode vote yes”, “masternode vote no” or “masternode vote abstain”. The votes then propagate across the network, and are tallied according to instructions followed by the network itself. Budgets under discussion and voting progress can be viewed using the example command “masternode budget show”.

A well defined decentralized system of governance allows a cryptocurrency network to endure and survive its original creators. In this way, later generations of masternode operators have a clear way to support the system as defined by the protocol itself, applying wisdom of the crowd techniques and the bond of trust established by the masternode collateral to create a decentralized management system. This creates incredible value within the currency, allowing us to be more agile and compete with other payment systems, such as Bitcoin and credit cards, on a global scale.

As the system has developed, a strong team of productive contractors paid from blockchain rewards has arisen and become established. This includes the core development team, escrow providers, news and reporting staff, experimental development labs, partnerships with universities, hiring of marketing and PR firms and integrations with third party exchanges and payment platforms. The market recognizes the value of the stability of the network as a whole, and that the possibility of reliable and sufficient funding results in faster and more coherent implementation of the GoByte roadmap and core GoByte services.

Reward schedule

To guarantee long term sustainability of the blockchain, the network keeps a portion of the block rewards back as new blocks are created, with the masternode operators tasked to act as stewards and invest in the maintenance and expansion of the network by voting. This results in faster development and promotion, creating a virtuous cycle that benefits all actors, including miners, masternode operators, investors and users. More importantly, this gives the blockchain itself a self- preservation mechanism that is beyond the control of any individual.

Mining reward for Proof-of-Work	25%
Masternode reward for Proof-of-Service	65%
Decentralized governance budget	10%

Masternodes take 65% of the mining reward, while miners take 25% of the mining reward at the time it is created. The remaining 10% is disbursed monthly by the masternode operators once the results of their votes are tallied, creating the first self-sustaining decentralized cryptocurrency platform organized as a Decentralized Autonomous Organization (DAO). The masternode operators establish a social contract with the network they benefit from and are bound to act as caretakers, dedicating their time, due diligence work and a portion of the network rewards to furthering the ecosystem. This has a ripple effect that benefits all parties involved - especially the end users.

The value generated by work done implementing proposals is expected to be greater than allocating 100% of rewards to mining because the network has needs beyond only cryptographically securing the blockchain. The expected result is greater net benefit not only for proposal winners, but also masternode operators, miners and normal users. In fact, the introduction of the decentralized governance budget itself was decided by a masternode vote, making the first distributed decision the actual creation of the system, similar to establishing a constitution.

This approach of distributing the normal block reward in a way that considers all critical elements a cryptocurrency needs for its long term viability, e.g. mining, full nodes, development and promotion, is revolutionary as it is done without changing the emission or creating any additional inflation for investors. The network simply distributes the available resources in a way that is of greater net benefit to all parties.

Contractors and proposals

Contractors of the blockchain can be developers, outreach professionals, team leaders, attorneys or even people appointed to do specific tasks. Proposals generally begin life as simple [pre-proposal forum posts](#) on the GoByte Forum, where feedback and suggestions are solicited from the general community. Once the proposal owner decides they have a reasonable chance of passing their proposal, it is created as a governance object on the blockchain. A fee of 5 GBX is associated with this action to prevent spam and ensure only serious proposals make it to this stage. Several tools exist to allow masternode operators to comfortably review and vote on proposals. The net total of yes votes must exceed 10% of the total masternode count at the time votes are tallied in order to pass. If there are more passing proposals than the available block reward can provide for, the proposals with the most yes votes will pass first, creating a cut-off point for less popular proposals. The same process is then repeated every month, and the total amount of GoByte available for proposals decreases by approximately 8.33% per year, together with the overall block reward.

Proposal websites

The community has gathered around [GoByteCentral](#) as a website to facilitate discussion and voting on proposals formally entered on the GoByte blockchain. Other websites, such as [GoByte Ninja](#) are available to monitor progress over time and gather more detailed statistics. [GoByte Masternode Tool](#) also allows for voting without the need to share masternode private keys with a third party service.



Fig. 161: A typical view of proposal discussion and voting on GoByte Central

Each proposal includes a description of the proposal goals, details of what work will be done and a breakdown of the requested budget. Many proposals also link to their own website or the pre-proposal discussion, or include a video to validate the identity and sincerity of the proposal owner. Discussion on GoByte Central occurs below this information, and masternode owners have the option to verify their ownership of a masternode and ability to cast a vote by signing a message from the masternode collateral address. Masternodes can vote at any time, and also change their vote at any

time until the cutoff block is mined and voting stops. This occurs 1728 blocks prior to the superblock. After voting stops, the blockchain executes a decentralized tally and validates all votes. Once consensus is reached, the results are broadcast and the budget is allocated soon after in a superblock.



Fig. 162: Proposal details and voting buttons on GoByte Central

Once passed, proposals are able to report back to the network on the [GoByte Forum](#) or via published public channels and social media. Since it is possible to create proposals that pay out over several months, it is also possible to revoke funding from a project by changing the vote if development or spending of already allocated funds is unsatisfactory. This encourages proposal owners to work honestly and diligently to win the trust and approval of the network. Ongoing discussion and gradual improvement over time results in a close bond between the network and those working for the network in supporting roles.

Voting on proposals is updated in real time via P2P messages and stored by GoByte in cache files, so current winning proposals and the total allocation of the available budget are always open and visible to everyone. [GoByte Nexus](#) is a popular site used for to view progress on proposal voting.

Budget allocation

The total budget of the network can be calculated by taking 10% of the reward over the period of time between two superblocks, which occur every 17280 blocks or approximately 30 days. A voting cutoff occurs 1728 blocks before the superblock, and the final votes are tallied at this point. A proposal must satisfy the condition $(\text{YES votes} - \text{NO votes}) > (\text{Total Number of Masternodes} / 10)$ in order to be considered passing. Then, in the superblock, the winning proposals are awarded in the order of the margin by which they are passing until either the entire budget is allocated or no more passing proposals exist. This allows for completely trustless and decentralized allocation of the budget.

If a proposal has passed the voting threshold but insufficient funds remain to pay the full amount requested, it will



Fig. 163: Monitoring budget allocation on GoByte Nexus

not receive partial funding. Instead, any smaller proposals which have also passed the threshold that will fit in the budget will be funded, even if they have lower net approval than the larger proposal. Proposals requesting payment over multiple budget periods will remain in the treasury system for the duration of their validity, even if they do not pass the voting threshold, and even if insufficient budget is available for funding as described above. Any unallocated budget is simply never created in the superblock, reducing unnecessary inflation.

Due to the decentralized nature of the masternode system, it is sometimes necessary to form funded organisations, such as committees or companies, to be responsible for some project or task. These are submitted in the same way, but the committee itself receives the funds. Another alternative is to place trusted escrow services between the budget allocation event and the actual submitter of the proposal to ensure that work is paid for in stages, as it is delivered. Some oversight over blockchain contractors is sometimes needed. Each budgeted item requires either a team manager or a committee responsible for implementation of the work. Periodically, this manager is expected to report on budget expenditure and completed work to show the value created from the allocated funds. This allows repeat proposal submitters to build up a reputation and gain trust from the community. Proposals which do not provide regular reports and cannot answer questions about their budget allocation will soon be defunded if it is part of a regular monthly proposal cycle. The result is a kind of self-policing system.

Scaling and future uses

As the number of blockchain contractors increases, a point is reached where masternode operators cannot be realistically expected to evaluate the volume of proposals. At this point funding organizations can be created to act as contractors for the distribution of funds to many smaller decentralized projects, according to current needs. GoByte Core Group, Inc. is one example of such an organization.

The existence of the decentralized budget system puts the power of determining where GoByte goes in the future in the hands of the masternode network itself. All core development and several peripheral developers are already funded from the budget, and other projects not even conceivable at this time will likely arise in the future. This decouples the survival and value of the blockchain from the current userbase and developers, making GoByte the first blockchain designed to outlive its original users, a self sustainable decentralized cryptocurrency network that can still operate cohesively and bring added value services to end users in a consistent way.

Conclusion

Every masternode operator establishes a bond of trust and a social contract with the network in which she is bound to contribute to the development and maintenance of the ecosystem she benefits from. Under this model, a portion of the funds that the operator is bound to receive are in a sense allocated in custody, not in ownership, and are held in escrow by the network to be executed by the operators for the benefit of the ecosystem. Everyone contributes equally and proportionately to the benefits they are receiving and the risks they are taking, there are no privileges and no loopholes. This is complemented by the full node voting mechanism that allows for a distributed group to vote on a continuous basis on practical matters without the need to forfeit their right to decide to others, every few years, like with traditional governments or cooperative corporations.

We envision a future in which this model of transparent, unbreakable and verifiable contribution to the common good, in combination with continuous participation of the crowd through active voting, is utilized to manage organizations that are owned or operated jointly by its members, who share the benefits and responsibilities of those collectives, like governments, cooperative corporations, unions, DAOs, cryptocurrencies, etc. We call this model decentralized governance by blockchain.

1.10.2 Using GoByte Governance

GoByte's Decentralized Governance by Blockchain (DGBB) is a novel voting and funding platform. This documentation introduces and details the theory and practice to use the system.

Understanding the process

Introduction

- DGBB consists of three components: Proposals, Votes, and Budgets
- Anyone can submit a proposal for a small fee
- Each valid masternode can vote for, against or abstain on proposals
- Approved proposals become budgets
- Budgets are paid directly from the blockchain to the proposal owner

Proposals

- Proposals are a request to receive funds
- Proposals can be submitted by anyone for a fee of 5 GBX. The proposal fee is irreversibly destroyed on submission.
- Proposals cannot be altered once submitted

Votes

- Votes are cast using the registered voting address
- The voting address can be delegated to a third party
- Votes can be changed at any time
- Votes are counted every 17280 blocks (approx. 30 days)

Budgets

- Budgets are proposals which receive a net total of yes votes equal to or greater than 10% of the total possible votes (for example over 448 out of 4480)
- Budgets can be nullified at any time if vote totals (cast or re-cast) fall below the approval threshold
- Budgets are processed (paid) in order of yes minus no votes. More popular budgets get payment priority.
- Approximately 21,780 GBX (in 2020) are available for each budget cycle, decreasing by 8.333% every 210240 blocks (approx. 383.25 days).

Object structure

The following information is required to create a proposal:

- proposal-name: a unique label, 20 characters or less
- url: a proposer-created webpage or forum post containing detailed proposal information
- payment-count: how many cycles the proposal is requesting payment
- block-start: the requested start of proposal payments
- gobyte-address: the address to receive proposal payments

- monthly-payment-gobyte: the requested payment amount

Persistence

- Proposals become active one day after submission
- Proposals will remain visible on the network until they are either disapproved or the proposal's last payment-cycle is reached
- Approval occurs when yes votes minus no votes equals 10% or more of the total available votes.
- Disapproval occurs when no votes minus yes votes equals 10% or more of the total available votes.
- The total available votes is the count of online and responding masternodes and can be seen by running the command `masternode count` in the GoByte Core wallet debug window. A graph of the total masternode count can be found [here](#)

Templates

The following two Microsoft Word templates are available from GoByte Core Group to help facilitate standardized proposal submission and updates. Usage is recommended, but not required.

- *Project Proposal Template*
- *Project Status Update Template*

Budget cycles

When preparing a proposal, be aware of when the next cycle will occur and plan accordingly. It is recommended to choose your proposal payment start block at least one cycle in the future to allow time for discussion and gathering support and votes. Note that votes will no longer be tallied 1728 blocks (approximately 3 days) prior to the superblock.

Block height	Approximate date
552960	Fri Aug 21 02:38:52 UTC 2020
570240	Sun Sep 20 09:43:54 UTC 2020
587520	Wed Oct 21 16:48:56 UTC 2020
604800	Fri Nov 20 23:53:58 UTC 2020
622080	Mon Dec 21 06:59:00 UTC 2020
639360	Wed Jan 20 14:04:02 UTC 2020
656640	Mon Feb 21 21:09:04 UTC 2020
673920	Sun Mar 21 04:14:06 UTC 2020
691200	Tue Apr 20 11:19:08 UTC 2020
708480	Fri May 21 18:24:10 UTC 2020
725760	Sun Jun 20 01:29:12 UTC 2020
743040	Wed Jul 21 08:34:14 UTC 2020

Creating proposals

Once you have prepared the text of your proposal and set up a website or forum post, it is time to submit your proposal to the blockchain for voting. While all tasks involved with creating a budget proposal can be executed from the GoByte Core wallet console, several tools providing a user interface have been developed to simplify this procedure.

GoByte Budget Proposal Generator

- <https://proposal.gobyte.network>

The [GoByte Budget Proposal Generator](#) supports creating budget proposals on both mainnet and testnet. In the first step, you must enter a short, clear and unique name for the proposal as it will appear on the blockchain. Proposal names are limited to 40 characters. You can then provide a link to the forum or GoByteCentral where your proposal is described in more detail (use a [URL shortening service](#) if necessary), as well as select the amount of payment you are requesting, how often the payment should occur, and the superblock date on which you are requesting payment. This allows you to control in which budget period your proposal will appear, and gives you enough time to build support for your proposal by familiarising voters with your project. Note that the payment amount is fixed and cannot be modified after it has been submitted to the blockchain.

The figure consists of two screenshots of the GoByte Budget Proposal Generator interface. The first screenshot, titled 'Create a Proposal', shows a form with the following fields: 'Proposal Name' (with 'documentation-test' entered), 'Proposal Description URL' (with 'https://docs.gobyte.network/' entered), 'Payment Date' (set to 8/24/2020), 'Payments' (set to 1 Payment), and 'Payment Address' (CZMkE5p3yng13ar5pvrHfHedvUSdWE1). The 'Payment Amount' is set to 100. A 'Create Proposal' button is at the bottom. The second screenshot, titled 'Wallet Commands', shows a 'Prepare Proposal Command' section with a long command to be pasted into the wallet console. Below this is a 'Transaction ID' field with a placeholder '-tbc-1000-'.

Fig. 164: Steps 1 & 2: Creating your proposal and preparing the command

Next, the proposal generator will provide you with a command to run from the console of your GoByte Core wallet to prepare your budget proposal governance object. Running this command will cost you 5 GBX, which will be “burnt” or permanently removed from circulation. This one-time fee protects the governance system from becoming overwhelmed by spam, poorly thought out proposals or users not acting in good faith. A small transaction fee is charged as well, so make sure slightly more than 5 GBX is available in your wallet. Many budget proposals request reimbursement of the 5 GBX fee.

First unlock your wallet by clicking **Settings > Unlock wallet**, then open the console by clicking **Tools > Debug console** and paste the generated command. The transaction ID will appear. Copy and paste this into the proposal generator response window. As soon as you do this, the system will show a progress bar as it waits for 6 confirmations as follows:

GoByteCentral also includes a tool to create budget proposals, or claim existing proposals so you can add a description on GoByteCentral and begin discussion with the community. The steps to be taken are almost identical to the procedure described above, and documentation is available [here](#).

Voting on proposals

You must vote at least three days before the superblock is created or your vote will not be counted. The exact deadline is 1728 blocks before the superblock.

Voting on DGBB proposals is an important part of operating a masternode. Since masternodes are heavily invested in GoByte, they are expected to critically appraise proposals each month and vote in a manner they perceive to be consistent with the best interests of the network. Each masternode may vote once on each proposal, and the vote can be changed at any time before the voting deadline. The following sites and tools are available to view and manage proposals and voting:

- [GoByteCentral](#)
- [GoByte Nexus](#)
- [GoByte Ninja - Governance](#)
- [GoByte Masternode Tool - Proposals](#)

For information on how to create a proposal, see [here](#).

GoByteCentral

Many masternode operators store their password-protected masternode private key on [GoByteCentral](#) to enable simple voting with a user-friendly interface. The popularity of this site has made it a common place for discussion of the proposals after they are submitted to the governance system. To vote from the GoByteCentral web interface, first add your masternode private key to your account according to the instructions [here](#). Note that the masternode private key is not the same as the private key controlling the 1000 GBX collateral, so there is no risk of losing your collateral. A separate password is required to unlock the masternode private key for voting, so the risk of the site operator voting in your name is minimal.

When you are ready to vote, go to the [budget proposals page](#). Simply click to view the proposals, then click either **Vote YES**, **Vote ABSTAIN** or **Vote NO**.

GoByte Masternode Tool (GMT)

If you started your masternode from a hardware wallet using [GMT](#), you can also use the tool to cast votes. Click **Tools > Proposals** and wait for the list of proposals to load. You can easily see the voting status of each proposal, and selecting a proposal shows details on the **Details** tab in the lower half of the window. Switch to the **Vote** tab to **Vote Yes**, **Vote No** or **Vote Abstain** directly from GMT.

GoByte Core wallet or masternode

If you started your masternode using the GoByte Core Wallet (not recommended), you can vote manually from **Tools > Debug console**, or directly from your masternode via SSH using `gobyte-cli`. First click on the proposal you want to vote on at either [GoByteCentral](#) or [GoByte Ninja](#). You will see a command for manual voting below the proposal description. Copy and paste the command and modify it as necessary. As an example, take this proposal from [GoByte Ninja](#) (or [GoByteCentral](#)). The voting code for GoByte Core Wallet is as follows:



Fig. 167: Voting interface on GoByteCentral



Fig. 168: Voting interface in GMT

```
gobject vote-many 05c89f3a615bdfd7cbbbbd62938ef79e9c0e958e8145dcc998e91e1c0f8fafa3_
↪funding yes
gobject vote-many 05c89f3a615bdfd7cbbbbd62938ef79e9c0e958e8145dcc998e91e1c0f8fafa3_
↪funding no
gobject vote-many 05c89f3a615bdfd7cbbbbd62938ef79e9c0e958e8145dcc998e91e1c0f8fafa3_
↪funding abstain
```

Note that to vote from your masternode directly, you need to prefix the command with `gobyte-cli`, which is usually found in the `.gobytecore` folder. The command should be similar to the following:

```
~/gobytecore/gobyte-cli gobject vote-many_
↪05c89f3a615bdfd7cbbbbd62938ef79e9c0e958e8145dcc998e91e1c0f8fafa3 funding yes
~/gobytecore/gobyte-cli gobject vote-many_
↪05c89f3a615bdfd7cbbbbd62938ef79e9c0e958e8145dcc998e91e1c0f8fafa3 funding no
~/gobytecore/gobyte-cli gobject vote-many_
↪05c89f3a615bdfd7cbbbbd62938ef79e9c0e958e8145dcc998e91e1c0f8fafa3 funding abstain
```

Note this command will trigger a vote from all masternodes configured in `gobyte.conf`. If you have multiple masternodes each with its own `.conf` file, or if you want to vote with only some of your masternodes, you must change the command from `vote-many` to `vote`. If your vote was successful, you should see a confirmation message reading **Voted successfully**.

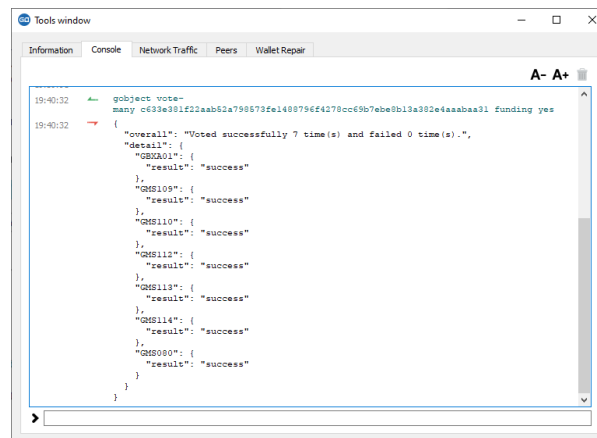


Fig. 169: Voting from the debug console in GoByte Core Wallet

You can also view a list of proposals in JSON format from the console to copy and paste the proposal hash for voting as follows:

```
gobject list
```

Delegating votes

DIP003 masternodes feature a separate voting key, which makes it possible to delegate your vote to a representative. Simply enter a GoByte address provided by the delegate when *registering your masternode*, or update your masternode registration to delegate the vote of a running masternode. The wallet controlling the private key to this address will then cast votes on behalf of this masternode owner simply by following the *GoByte Core voting procedure* described above. No further configuration is required.

1.10.3 8 Steps to a Successful Proposal

Proposals in the GoByte governance system are subject to voting by masternodes. So, like any voting, you need to convince the voters that your proposal should pass. Here are some key points to consider in every proposal:

Keep your proposal clear Your proposal should have a clear title, followed by a short and simple description of the objectives. Explain early in your proposal exactly how it will benefit the GoByte network, how much GoByte you are requesting, how you arrived at this value, and finally who you are and how you plan to do the work. Masternodes should be able to immediately get an idea of what you are proposing from the first few lines of your proposal.

Run a pre-proposal discussion Get feedback from the community before you post your proposal to the blockchain. A discussion period of around two weeks will help you find out if someone has proposed something similar in the past, and whether it succeeded or failed. There are [pre-proposal channels on the forum](#) and [GoByte Core Discord](#), and [Reddit](#) also attracts a lot of views - consider the discussion on these channels to be the research phase of your proposal. Later, you can post a link to the forum discussion when your proposal goes live to show you are including community feedback in your work.

Manage your identity and reputation The GoByte community is one of the network's strongest features, and newcomers are always welcome. However, because of the way proposals work, there needs to be reasonable trust that the work promised in the proposal will be completed if it passes. If you are new, consider starting with a smaller proposal first to prove your ability to deliver on time and budget. Attaching your real name or [Key-base](#) identity to a proposal also helps build trust. If you are making a large proposal, get a team together and nominate (or hire) one person to serve as community liaison, since posting from multiple accounts can be confusing.

Run an enthusiastic campaign for your proposal Proposals with a video or website have a far greater chance of succeeding! Uploading a video gives your proposal a human touch and a chance to convey your enthusiasm in a way that isn't always possible in text. Post your video to the [forum](#), become a regular on Discord or run a webinar to explain the proposal and answer questions. Put some work in before you ask for funding to demonstrate your involvement with GoByte - but don't be annoying and spam many channels asking for votes.

Demonstrate your commitment to the network If you are asking for significant funding to start up or expand a for-profit business built on GoByte, you need to explain why and for how long this funding is required, and what you are offering in return. It can be very helpful to show you have skin in the game by matching the contribution provided in GoByte with funds from your own business or investors. Equity or exclusivity agreements can be reached with [GoByte Core Group, Inc.](#), but must be clarified in writing before the proposal is posted.

Post your proposal early and make yourself available for questions The voting window closes 1728 blocks (or just under 3 days) before the superblock. To give the masternode operators enough time to consider, discuss and vote on your proposal, you must post it well in advance of the voting deadline - it's better to wait for the next superblock than to rush! Most masternodes will see your proposal for the first time once it appears on the blockchain or when you claim it on GoByteCentral. The first few hours of discussion between masternode owners typically bring up a lot of questions and can be critical to influence opinion and voting, so make yourself available during this time.

Keep the community updated when your proposal passes Your proposal should include details of how you plan to keep the community and network informed of your work. Meet your commitments and post regular reports so your output is clear, and make yourself available on social channels to answer questions. Remember, your ability to pass future proposals depends on your demonstrated ability to deliver and communicate.

Consider arrangements for large requests If you are requesting a significant amount of funding, there is an understandable concern that you will deliver on your promises to the network. Reach out to trusted intermediaries such as [GoByte Core Group, Inc.](#) in advance for advice on escrow, and make the conditions for escrow release public and part of the proposal. If your proposal is so large that uses a significant percentage of the budget, there is a risk that approving your proposal will bump smaller proposals out of the budget. Consider breaking your proposal into smaller monthly payments instead.

See [this documentation](#) for specific instructions on how to create a proposal when you are ready. Good luck!

A few additional points:

1. It is currently not possible to pay a budget proposal to a multisig address, or to change the payment address after the proposal is posted to the blockchain.
2. To avoid accusations of favouritism and inside trading, GoByte Core cannot promote your proposal for you. If your proposal is an integration, reach out to the business development team in advance. Once your product is live, it may be possible to announce it from GoByte Core channels.
3. If your proposal is for news, promotion or marketing, make sure you synchronise your efforts with major existing organisations such as GoByte News or marketing firms contracted by GoByte Core.
4. You are responsible for your own planning to hedge against price volatility. If your proposal involves significant payments to third parties in fiat currency, reach out to [GoByte Core Group, Inc.](#) for advice on escrow, price maintenance, converting currencies and hedging against volatility.
5. For the same reason, it is not recommended to request funding for period of longer than six months. Masternodes don't want to see and vote on the same proposal without updates several months in a row, and price volatility makes it a risky proposition both to the network and yourself.
6. Before entering your budget proposal on the blockchain, check how many proposals already exist for the current budget cycle. If it is likely to become very crowded or if some proposals are requesting a significant portion of the budget, voting is likely to be very competitive with weaker projects being forced out of the budget, even if they collect sufficient votes to pass the 10% threshold. See [here](#) for more details.

1.11 Masternodes

GoByte is best known as the first cryptocurrency with a focus on anonymity and transaction speed. What many people do not know is that these features are implemented on top of a network of dedicated servers known as masternodes, which gives rise to many exciting features not available on conventional blockchains. These features include anonymous and instant transactions, as well as governance of the development of the GoByte network through a monthly budget and voting. This in itself is a first in the crypto world, and the masternodes are necessary to achieve the privacy and speed that GoByte offers.

This documentation focuses on understanding the services masternodes provide to the network, but also includes guides on how to run a masternode, using either a hosting provider or by setting up and maintaining your own hosting solution. The primary requirement to run a masternode on the GoByte network is 1000 GBX. This is known as the collateral, and cannot be spent without interrupting operation of the masternode. The second requirement is the actual server running the GoByte masternode software.

Option 1: Hosted masternode

Since operating your own server requires a certain level knowledge of blockchains and Linux server operating systems, several community members offer dedicated hosting solutions for a fee. Taking advantage of these services means the user only needs to provide the masternode collateral and pay the hosting fee in order to receive payment from the block reward. See [these pages](#) for information on how to set up a hosted masternode.

Option 2: Self-operated masternode

Users with a deeper understanding (or curiosity) about the inner workings of the GoByte network may choose to operate their own masternode on their own host server. Several steps are required, and the user must assume responsibility for setting up, securing and maintaining both the server and collateral. See these pages for information on how to set up a self-operated masternode.

1.11.1 Understanding Masternodes

Masternodes, once unique to the Dash network, are now becoming popular as the technology is forked into other blockchains, including GoByte. This section of the documentation describes the principles and mechanisms of masternodes and the services they provide to the GoByte network specifically.

Simply put, a masternode is a server with a full copy of the GoByte blockchain, which guarantees a certain minimum level of performance and functionality to perform certain tasks related to block validation, as well as PrivateSend and InstantSend, as the anonymity and instant transaction features in GoByte are called. The masternodes are paid for this service, using a concept known as Proof of Service. This is in addition to the Proof of Work done by miners to secure the blockchain. Masternodes are also allowed to vote on *governance and funding proposals*, with each masternode receiving one vote (yes/no/abstain) on each proposal submitted to the system.

Anyone can run a masternode. The objective is to have enough decentralization to ensure that no single person controls a significant fraction of the masternodes. However, to avoid bloating the network with unnecessary masternodes or encouraging reckless operators, there is one condition that needs to be fulfilled: proof of ownership of 1000 GBX. The coins don't need to be in the masternode, but they need to be kept in a certain way that is transparent to the entire network. If the owner moves or spends those coins, the masternode stops working and payment ceases.

Masternodes are paid by the network for the PrivateSend, InstantSend and governance services they provide. 65% of the block reward is paid out to the masternodes, 25% to miners and 10% to the budget. In practice, half of the reward from a normal block goes to the miner and half to the masternode. Then, every 17,280 blocks (approximately 30 days), a superblock is created that contains the entire 10% payout to the budget proposal winners. Masternodes are randomly selected for payment in each block (approximately every 2.5 minutes) from a list once they reach the top 10% of the total count of masternodes, and moved to the back of the list after payment. As more masternodes are created, the duration between payments increases. Due to the selection algorithm, there is always an aspect of randomness to payment selection, but in the long term all masternode owners should receive similar payments. If the collateral behind a masternode is spent, or if a masternode stops providing services to the network for more than one hour, it is removed from the list until normal service resumes. In this way, masternodes are given incentive to provide efficient and reliable services to the network.

Having so many servers holding a full copy of the blockchain and working for the coin can be extremely useful. Thanks to the reward system, there is no risk of not having enough masternodes, and the developers can rely on them quickly deploying any new decentralized feature they want to implement. This is where the true strength of GoByte lies - an incentivized system of thousands of distributed servers working 24x7 means that GoByte can scale more efficiently and deploy services more quickly than a blockchain run entirely by unpaid volunteers. The more masternodes, the better and safer the GoByte network.

As of August 2020, the GoByte network has [over 2000 masternodes located](#) in over [41 countries](#) and hosted on [over 100 ISPs](#). The block reward is approximately 11.34375 GoByte, so the selected masternode receives 7.940625 GBX per payment or approximately 67.0470 GBX per month. The block reward decreases by 8.33333% approximately once per year, so the annual earnings for a masternode owner is approximately 81.57% of the collateral, and will decrease over time *as calculated here*. See [this tool](#) to calculate real-time payment rates, and [this site](#) for various real-time statistics on the masternode network.

Masternodes vs. mining

GoByte, like Bitcoin and most other cryptocurrencies, is based on a decentralized ledger of all transactions, known as a blockchain. This blockchain is secured through a consensus mechanism; in the case of both GoByte and Bitcoin, the consensus mechanism is Proof of Work (PoW). *Miners* attempt to solve difficult problems with specialized computers, and when they solve the problem, they receive the right to add a new block to the blockchain. If all the other people running the software agree that the problem was solved correctly, the block is added to the blockchain and the miner is rewarded.

GoByte works a little differently from Bitcoin, however, because it has a two-tier network. The second tier is powered by masternodes (Full Nodes), which enable financial privacy (PrivateSend), instant transactions (InstantSend), and the

decentralized governance and budget system. Because this second tier is so important, masternodes are also rewarded when miners discover new blocks. The breakdown is as follows: 25% of the block reward goes to the miner, 65% goes to masternodes, and 10% is reserved for the budget system (created by superblocs every month).

The masternode system is referred to as Proof of Service (PoSe), since the masternodes provide crucial services to the network. In fact, the entire network is overseen by the masternodes, which have the power to reject improperly formed blocks from miners. If a miner tried to take the entire block reward for themselves or tried to run an old version of the GoByte software, the masternode network would orphan that block, and it would not be added to the blockchain.

In short, miners power the first tier, which is the basic sending and receiving of funds and prevention of double-spending. Masternodes power the second tier, which provide the added features that make GoByte different from other cryptocurrencies. Masternodes do not mine, and mining computers cannot serve as masternodes. Additionally, each masternode is “secured” by 1000 GBX. Those GBX remain under the sole control of their owner at all times, and can still be freely spent. The funds are not locked in any way. However, if the funds are moved or spent, the associated masternode will go offline and stop receiving rewards.

Payment logic

Masternode payments in GoByte version 12 are determined using a completely decentralized deterministic queue with probabilistic selection.

Global list

Every masternode appears in the global list. Their position in this list is determined by their time since the last payment according to the network, not the blockchain. New masternodes joining the network and masternodes receiving payment are placed at the end of the list. Running, active masternodes which are restarted using the rpc commands ‘masternode start’ or ‘masternode start-alias’ are also placed at the end of the list. Using the new rpc command ‘masternode start-missing’ avoids this. As masternodes are moved to the end of the global list, the remaining masternodes slowly migrate towards the top of the list. Once a masternode reaches the top 10% of the global list, it is eligible for selection from the selection pool.

Selection pool

The selection pool is the top 10% of the global list. Its size is determined by the total masternode count. As an example, if there are 4500 active masternodes, the top 450 masternodes in the global list are eligible for selection. Once in the selection pool, selection for payment is determined by block hash entropy. The block hash 100 blocks ago determines which masternode will be selected for payment. A double SHA256 of the funding transaction hash and index for all masternodes in the selection pool is compared with the proof of work hash 100 blocks ago. The masternode with the closest numeric hash value to that block hash is selected for payment.

Probabilities

Because selection is determined by block hash entropy, it is impossible to predict when a payment will occur. Masternode operators should expect considerable variance in payment intervals over time. Once a masternode enters the selection pool, payments become a probability. The probabilities in this example are calculated using an assumed current pool size of 450 (at 4500 total masternodes). Nodes in the selection pool are selected for rewards randomly, i.e. the probability of being selected on any given block is 1/450.

The table below shows the probability of an eligible node being selected for payment over a particular period of time. For example, the probability that an eligible node is selected within 12 hours is about 46%. The table does **not** (and cannot) tell us the probability of being selected **after** a given period of time. For example, if you haven’t been selected within the past 12 hours — and we know from this table there is about a 54% chance of that happening — the probability of

being selected on the next block is **not** 46%. It remains 1/450. Putting these together, if you have an eligible node, and, say, 48 hours have passed without payment, then you've been very unlucky, as there's less than a 10% chance of that happening. But, your chances of being selected on the next block remain the same as for any block, i.e. 1/450.

Once a node is selected for payment, it is moved to the back of the list and cannot be selected again until it re-enters the selection pool.

Hours	Blocks	Probability
1	24	4.69%
2	48	9.16%
3	72	13.42%
4	96	17.48%
6	144	25.04%
8	192	31.91%
10	240	38.15%
12	288	43.81%
18	432	57.89%
24	576	68.43%
30	720	76.34%
36	864	82.86%
42	1008	86.70%
48	1152	90.03%
72	1728	96.85%
96	2304	99.00%

You can run the code (written by community member sirbond used to create the table above *here*).

Quorum selection

InstantSend transactions in GoByte version 12 are secured using a consensus of deterministically selected masternodes. This set of masternodes is informally termed a quorum and must be in a majority agreement, at least six out of ten, for a successful lock of the transaction inputs. Multiple quorums are self-selected for each input in an InstantSend transaction using the mathematical distance between the hash of each input and of the set of masternode funding transactions.

Each masternode receiving the InstantSend transaction lock request compares the hash of the masternode's funding transaction to the hash of the input requesting the lock. After validating the inputs are not spent, the ten masternodes furthest from this hash broadcast their acceptance of the lock.

All InstantSend inputs must be at least six blocks old or the transaction will be rejected.

Masternode requirements

- 1000 GBX: Arguably the hardest part. GoByte can be obtained from exchanges such as HitBTC, STEX, Bi-rakeDEX and Sistemkoin. InstaSwap's service is also an excellent way.
- A server or VPS running Linux: Most recent guides use Ubuntu 16.04 LTS. We recommend VPS services such as Vultr and DigitalOcean, although any decent provider will do. Generally an instance with low to average specifications will do, although performance requirements will increase according to this roadmap.
- A dedicated IP address: These usually come with the VPS/server.
- A little time and (heart): Masternodes used to require complex setup, but tools such as gobyteman now greatly simplify the process.

In addition to the 1000 GBX held in collateral, masternodes also have minimum hardware requirements. As of version 12.1, these requirements are as follows:

	Minimum	Recommended
CPU	1x 1 GHz	1x 2 GHz
RAM	1 GB	2 GB
Disk	8 GB	16 GB
Network	400 GB/mth	1 TB/mth

Masternode bandwidth use ranges between 300-500 GB per month and will grow as the network does.

GoByte Evolution

The exact hardware requirements for GoByte Evolution masternodes have yet to be determined, although some pointers can be taken from the Dash's [roadmap](#) and this [blog post](#). It should be possible to run GoByte masternodes on normal VPS servers until the block size reaches approximately 20 MB, after which custom hardware such as GPUs and eventually ASICs will be required.

1.11.2 Hosting Services

Several GoByte community members offer masternode hosting services. This service can be realized securely without the customer ever giving up control of the 1000 GBX required for collateral. For security reasons, it is highly recommended to keep the collateral on a hardware wallet when taking advantage of a hosting service. A list of currently available masternode hosting services is available below.

List of hosting services

Disclaimer: GoByte Core may be affiliated with these community members, but is not involved in the provision of any of these services.

AllNodes



<https://www.allnodes.com>

- Operated by: Sephiroth
- Services: Hosting

- [Site](#)
- [Email](#)
- [Twitter](#)
- [Telegram](#)
- [Discord](#)

PosMN.com



<https://www.allnodes.com>

- Operated by: Kris
- Services: Hosting
- [Site](#)
- [Email](#)
- [Twitter](#)
- [Telegram](#)
- [Discord](#)

iHostMN



<https://ihostmn.com/>

- Operated by: iHostMN
- Services: Hosting
- [Site](#)
- [Email](#)
- [Twitter](#)
- [Telegram](#)
- [Discord](#)

Masternodehosting

<https://masternodehosting.com>

- Operated by: flare (GoByte Core team member)
- Services: Hosting
- [Site](#)
- [Email](#)
- [Forum](#)

Starting a hosted masternode

Starting a hosted masternode is done in just a few easy steps:

1. Send 1000 GBX to an address you control in a single transaction and wait for 15 confirmations
2. Communicate the address to your hosting provider, who will provide you with a masternode IP address and private key
3. Enter this information in your wallet and start the masternode

It is **highly recommended** to store the keys to your masternode collateral on a *hardware wallet* for added security against hackers. This documentation will use a Trezor as an example, but KeepKey and Ledger are also supported. For instructions on using GoByte Core wallet to start the masternode (no longer recommended), contact your hosting provider.

Send the collateral

Set up your Trezor using the Trezor wallet at <https://wallet.trezor.io>, update the firmware if necessary and send a test transaction to verify that it is working properly. For help on this, see *this guide*. Create a new account in your Trezor wallet by clicking **Add account**. Then click the **Receive** tab and send exactly 1000 GBX to the address displayed. You should see the transaction as soon as the first confirmation arrives, usually within a few minutes.

Once the transaction appears, click the QR code on the right to view the transaction on the blockchain. Keep this window open as we complete the following steps, since we will soon need to confirm that 15 confirmations exist, as shown in the following screenshot.



masternodes/img/setup-collateral-trezor.png

Fig. 170: Trezor Wallet Receive tab showing successfully received collateral of 1000 GBX



Fig. 171: Trezor blockchain explorer showing 15 confirmations for collateral transfer

Correspond with your hosting provider

Once 15 confirmations exist, send the address holding the 1000 GBX to your hosting provider. Payment for operating the masternode will generally also be requested at this point - if paying in GoByte, be careful not to pay from the address holding the collateral. You will receive a reply with the following data:

- A server IP address
- A masternode private key
- The collateral transaction ID (optional)

Start the masternode

The GoByte Masternode Tool (GMT) is required to combine all of this data and issue the command to the network to start the masternode. Download the appropriate version of GMT for your computer from the [GitHub releases page](#) [here](#). Unzip the file and run the executable. The following window appears.



Fig. 172: GoByte Masternode Tool startup screen

We will now do the final preparation in GoByte GMT. Carry out the following sequence of steps as shown in this screenshot from GMT developer SirBound:

1. Enter the name of your masternode here.
2. Enter the IP address of your masternode, as provided by your host.
3. Enter the TCP port number. This should be 12455.
4. Instead of clicking **Generate new**, simply enter the masternode private key provided by your host.



Fig. 173: GoByte Masternode Tool configuration steps

5. Copy the collateral address where you sent the 1000 GBX collateral from your Trezor Wallet and paste it in this field.
6. Click the **arrow** → to derive the BIP32 path from your collateral address. You can verify this against the BIP32 path shown on the receive tab in your Trezor Wallet for the transaction.
7. Click **Lookup** to find the collateral TX ID for the transaction which transferred the collateral to the address. You can verify this against the TXID shown on the confirmation page of the blockchain explorer for your collateral address.



Fig. 174: GoByte Masternode Tool with configuration ready to start masternode

Click **Start Masternode using Hardware Wallet**. Enter your PIN and confirm on your hardware wallet that you want to transmit this command. The following messages will appear, confirm each one:

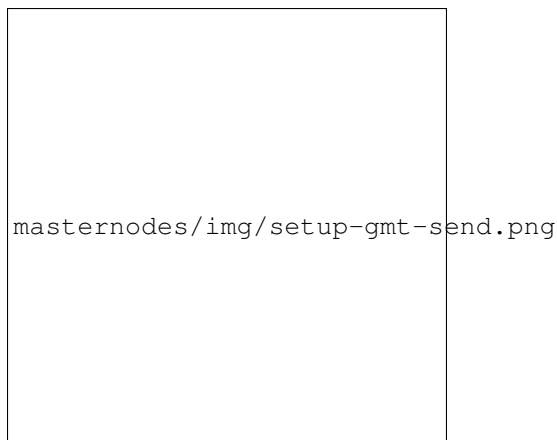




Fig. 175: GoByte Masternode Tool confirmation dialogs to start a masternode

That's it! Your masternode is now running, and you should receive regular payments to your masternode address. You can monitor your masternode's acceptance by the network by entering the collateral address to search the masternode list at <https://masternodes.gobyte.network/> For information on how to withdraw masternode payments without affecting operation of the masternode, see [here](#).

1.11.3 Setup

Setting up a masternode requires a basic understanding of Linux and blockchain technology, as well as an ability to follow instructions closely. It also requires regular maintenance and careful security, particularly if you are not storing your GoByte on a hardware wallet. There are some decisions to be made along the way, and optional extra steps to take for increased security.

If you prefer to use a masternode hosting service, several community members provide hosting at posmn.com, nodehub.io, iHostMn.com or [zCore Central](https://zCoreCentral.com). When using these hosting services, all you have to do is send a single transaction of 1000 GBX to a specific address and communicate the transaction ID to the hosting service. Simply follow the steps [here](#).

This guide is heavily based on previous Dash guides written by [Bertrand256](#), [moocowmoo](#), [tao](#), [BolehVPN](#) and [tungfa](#). Tao's hugely popular original guide and support thread is available [here](#), as well many more guides for specific cases in this forum.

Before you begin

This guide assumes you are setting up a single masternode for the first time. If you are updating a masternode, see [here](#) instead. You will need:

- 1000 GoByte
- A wallet to store your GoByte, preferably a hardware wallet, although GoByte Core wallet is also supported
- A Linux server, preferably a Virtual Private Server (VPS)

We also assume you will be working from a Windows computer. However, since most of the work is done on your Linux VPS, alternative steps for using macOS or Linux will be indicated where necessary.

Set up your VPS

A VPS, more commonly known as a cloud server, is a fully functional installation of an operating system (usually Linux) operating within a virtual machine. The virtual machine allows the VPS provider to run multiple systems on one physical server, making it more efficient and much cheaper than having a single operating system running on the “bare metal” of each server. A VPS is ideal for hosting a GoByte masternode because they typically offer guaranteed uptime, redundancy in the case of hardware failure and a static IP address that is required to ensure you remain in the masternode payment queue. While running a masternode from home on a desktop computer is technically possible, it will most likely not work reliably because most ISPs allocate dynamic IP addresses to home users.

We will use [Vultr](#) hosting as an example of a VPS, although [DigitalOcean](#), [Amazon EC2](#), [Google Cloud](#), [Choopa](#) and [OVH](#) are also popular choices. First create an account and add credit. Then go to the Servers menu item on the left and click + to add a new server. Select a location for your new server on the following screen:

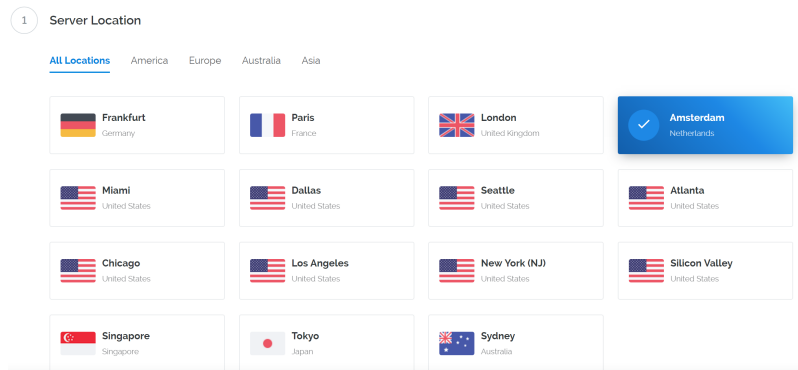


Fig. 176: Vultr server location selection screen

Select Ubuntu 16.04 x64 as the server type. We use 16.04 instead of the latest version because 16.04 is an LTS release of Ubuntu, which will be supported with security updates for 5 years instead of the usual 9 months.

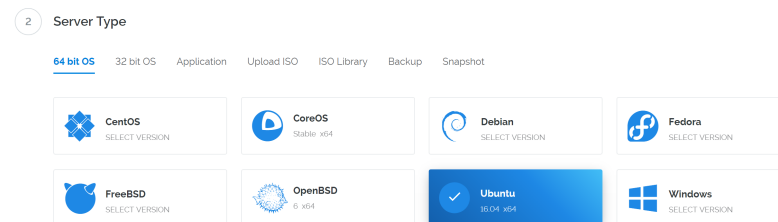


Fig. 177: Vultr server type selection screen

Select a server size offering at least 2GB of memory.

Enter a hostname and label for your server. In this example we will use `gobytemn1` as the hostname.

Vultr will now install your server. This process may take a few minutes.

Click **Manage** when installation is complete and take note of the IP address, username and password.

Set up your operating system

We will begin by connecting to your newly provisioned server. On Windows, we will first download an app called PuTTY to connect to the server. Go to the [PuTTY download page](#) and select the appropriate MSI installer for your

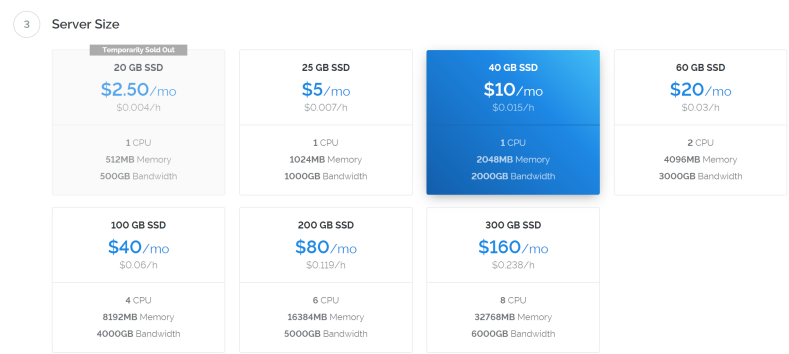


Fig. 178: Vultr server size selection screen



Fig. 179: Vultr server hostname & label selection screen



Fig. 180: Vultr server installation screen



Fig. 181: Vultr server management screen

Download PuTTY: latest release (0.69)

[Home](#) | [FAQ](#) | [Feedback](#) | [Licence](#) | [Updates](#) | [Mirrors](#) | [Keys](#) | [Links](#) | [Team](#)
 Download: [Stable](#) · [Snapshot](#) | [Docs](#) | [Changes](#) | [Wishlist](#)

This page contains download links for the latest released version of PuTTY. Currently this is 0.69, released on 2017-04-29.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a [permanent link to the 0.69 release](#).

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the [development snapshots](#), to see if the problem has already been fixed in those versions.

Package files

You probably want one of these. They include all the PuTTY utilities.

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

MSI ('Windows Installer')

32-bit:	putty-0.69-installer.msi	(or by FTP)	(signature)
64-bit:	putty-64bit-0.69-installer.msi	(or by FTP)	(signature)

Unix source archive

.tar.gz:	putty-0.69.tar.gz	(or by FTP)	(signature)
----------	-----------------------------------	-----------------------------	-----------------------------

Fig. 182: PuTTY download page

system. On Mac or Linux you can ssh directly from the terminal - simply type `ssh root@<server_ip>` and enter your password when prompted.

Double-click the downloaded file to install PuTTY, then run the app from your Start menu. Enter the IP address of the server in the **Host Name** field and click **Open**. You may see a certificate warning, since this is the first time you are connecting to this server. You can safely click **Yes** to trust this server in the future.

You are now connected to your server and should see a terminal window. Begin by logging in to your server with the user `root` and password supplied by your hosting provider.

You should immediately change the root password and store it in a safe place for security. You can copy and paste any of the following commands by selecting them in your browser, pressing **Ctrl + C**, then switching to the PuTTY window and right-clicking in the window. The text will paste at the current cursor location:

```
passwd root
```

Enter and confirm a new password (preferably long and randomly generated). Next we will create a new user with the following command, replacing `<username>` with a username of your choice:

```
adduser <username>
```

You will be prompted for a password. Enter and confirm using a new password (different to your root password) and store it in a safe place. You will also see prompts for user information, but this can be left blank. Once the user has been created, we will add them to the sudo group so they can perform commands as root:

```
usermod -aG sudo <username>
```

Now, while still as root, we will update the system from the Ubuntu package repository:

```
apt update
apt upgrade
```

The system will show a list of upgradable packages. Press **Y** and **Enter** to install the packages. We will now install a firewall (and some other packages we will use later), add swap memory and reboot the server to apply any necessary

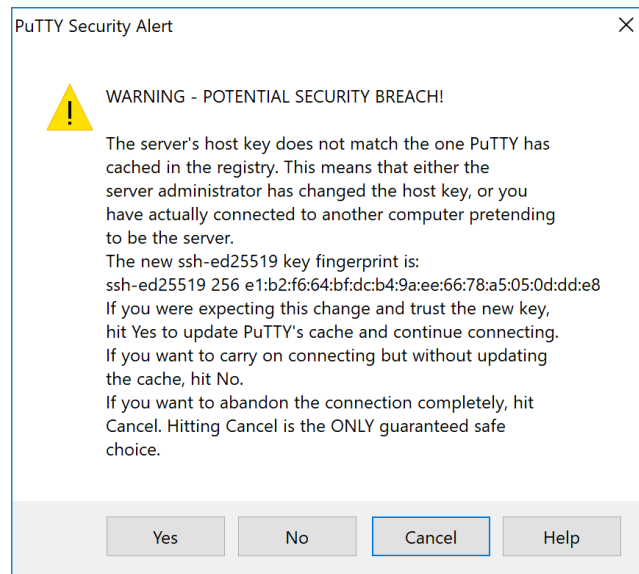


Fig. 183: PuTTY security alert when connecting to a new server

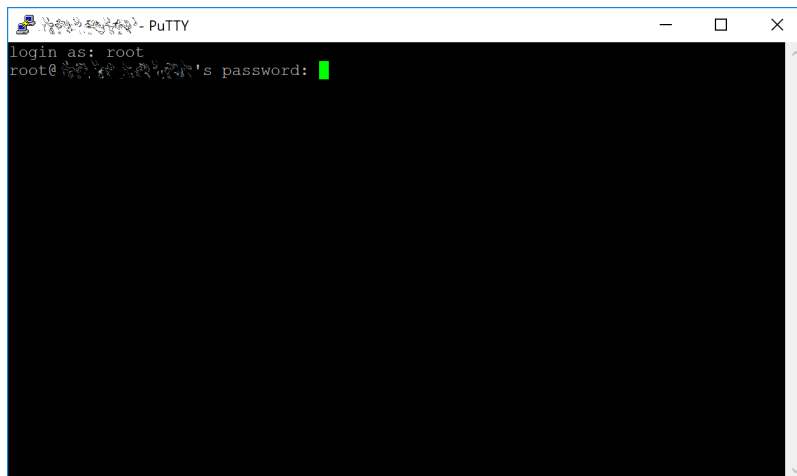


Fig. 184: Password challenge when connecting to your VPS for the first time

kernel updates, and then login to our newly secured environment as the new user:

```
apt install ufw python virtualenv git unzip pv
```

(press **Y** and **Enter** to confirm)

```
ufw allow ssh/tcp
ufw limit ssh/tcp
ufw allow 12455/tcp
ufw logging on
ufw enable
```

(press **Y** and **Enter** to confirm)

```
fallocate -l 4G /swapfile
chmod 600 /swapfile
mkswap /swapfile
swapon /swapfile
nano /etc/fstab
```

Add the following line at the end of the file (press tab to separate each word/number), then press **Ctrl + X** to close the editor, then **Y** and **Enter** save the file.

```
/swapfile none swap sw 0 0
```

Finally, in order to prevent brute force password hacking attacks, open the SSH configuration file to disable root login over SSH:

```
nano /etc/ssh/sshd_config
```

Locate the line that reads `PermitRootLogin yes` and set it to `PermitRootLogin no`. Directly below this, add a line which reads `AllowUsers <username>`, replacing `<username>` with the username you selected above. The press **Ctrl + X** to close the editor, then **Y** and **Enter** save the file.

Then reboot the server:

```
reboot now
```

PuTTY will disconnect when the server reboots.

While this setup includes basic steps to protect your server against attacks, much more can be done. In particular, [authenticating with a public key](#) instead of a username/password combination, [installing fail2ban](#) to block login brute force attacks and [enabling automatic security updates](#) is advisable. More tips are available [here](#). However, since the masternode does not actually store the keys to any GoByte, these steps are considered beyond the scope of this guide.

Send the collateral

A GoByte address with a single unspent transaction output (UTXO) of exactly 1000 GBX is required to operate a masternode. Once it has been sent, various keys regarding the transaction must be extracted for later entry in a configuration file as proof that the transaction was completed successfully. A masternode can be started from a hardware wallet or the official GoByte Core wallet, although a hardware wallet is highly recommended to enhance security and protect yourself against hacking. This guide will describe the steps for both hardware wallets and GoByte Core.

Option 1: Sending from a hardware wallet

Set up your Trezor using the Trezor wallet at <https://wallet.trezor.io/> and send a test transaction to verify that it is working properly. For help on this, see [this guide](#) - you may also choose to (carefully!) [add a passphrase](#) to your Trezor to further protect your collateral. Create a new account in your Trezor wallet by clicking **Add account**. Then click the **Receive** tab and send exactly 1000 GBX to the address displayed. If you are setting up multiple masternodes, send 1000 GBX to consecutive addresses within the same new account. You should see the transaction as soon as the first confirmation arrives, usually within a few minutes.

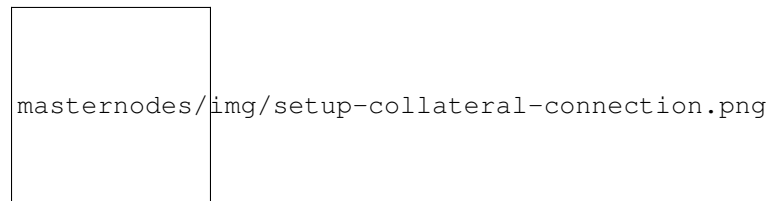


Fig. 185: Trezor Wallet Receive tab showing successfully received collateral of 1000 GBX

Once the transaction appears, click the QR code on the right to view the transaction on the blockchain. Keep this window open as we complete the following steps, since we will soon need to confirm that 15 confirmations exist, as shown in the following screenshot.

While we are waiting for 15 confirmations, download the latest version of the GoByte Masternode Tool (GMT) from the GitHub releases page [here](#). Unzip and run the file. The following window appears.

Click **Check RPC connection** in the top left corner of the main window to verify that the connection is working. Then connect your Trezor device and click **Test HW** to verify the Trezor connection is working.



We will now use GMT to extract the transaction ID. Carry out the following sequence of steps as shown in this

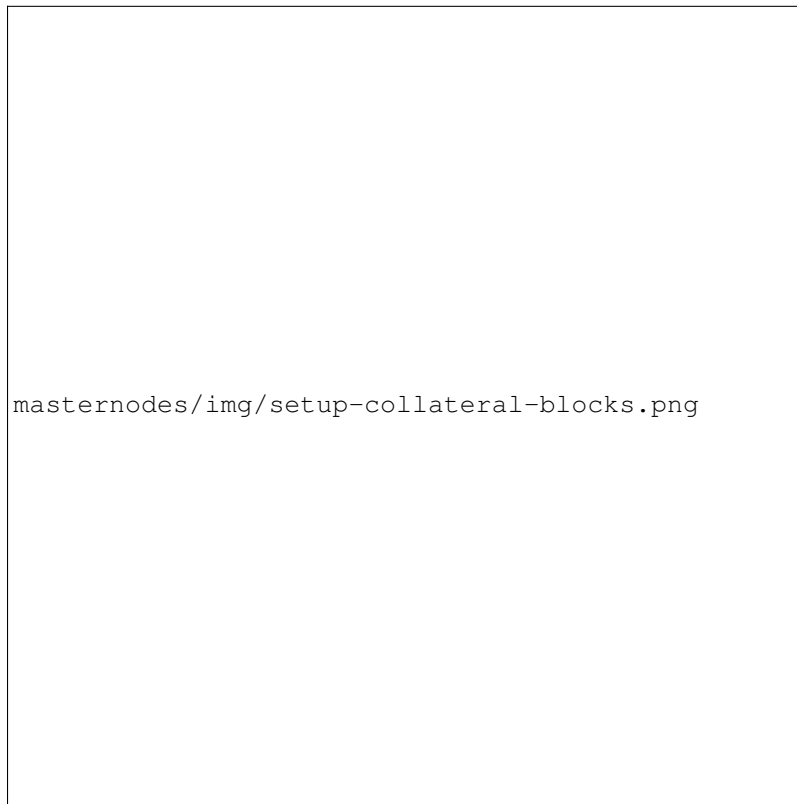


Fig. 186: Trezor blockchain explorer showing 15 confirmations for collateral transfer



Fig. 187: GoByte Masternode Tool startup screen



Fig. 188: GoByte Masternode Tool successful connection confirmations

screenshot from GMT developer Bertrand256:



Fig. 189: GoByte Masternode Tool configuration steps

1. Enter the name of your masternode here. This should match the hostname as defined when setting up your server, gobytemn1 in this case. You can view this in the first line of the output of gobyteman/gobyteman status.
2. Enter the IP address of your masternode here. This was given to you by the VPS provider when you set up the server.
3. Enter the TCP port number. This should be 12455.
4. Click Generate new to generate a new masternode private key.
5. Copy the collateral address where you sent the 1000 GBX collateral from your Trezor Wallet and paste it in this field.
6. Click the arrow → to derive the BIP32 path from your collateral address. You can verify this against the BIP32 path shown on the receive tab in your Trezor Wallet for the transaction.
7. Click Lookup to find the collateral TX ID for the transaction which transferred the collateral to the address. You can verify this against the TX ID shown on the confirmation page of the blockchain explorer for your collateral address.

Leave GMT open, take note of the masternode private key and collateral address and continue with the next step: *installing GoByte Core on your VPS*.

Option 2: Sending from GoByte Core wallet

Open GoByte Core wallet and wait for it to synchronize with the network. It should look like this when ready:



Fig. 190: GoByte Masternode Tool with configuration ready to start masternode



Fig. 191: Fully synchronized GoByte Core wallet

Click **Tools > Debug console** to open the console. Type the following two commands into the console to generate a masternode key and get a fresh address:

```
masternode genkey  
getaccountaddress 0
```

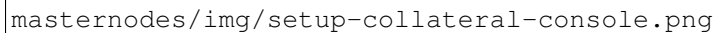
The image is a placeholder for a screenshot of a terminal window. The text 'masternodes/img/setup-collateral-console.png' is centered within a large rectangular box, indicating where the screenshot should be placed.

Fig. 192: Generating a masternode private key in GoByte Core wallet

Take note of the masternode private key and collateral address, since we will need it later. The next step is to secure your wallet (if you have not already done so). First, encrypt the wallet by selecting **Settings > Encrypt wallet**. You should use a strong, new password that you have never used somewhere else. Take note of your password and store it somewhere safe or you will be permanently locked out of your wallet and lose access to your funds. Next, back up your wallet file by selecting **File > Backup Wallet**. Save the file to a secure location physically separate to your computer, since this will be the only way you can access our funds if anything happens to your computer. For more details on these steps, see [here](#).

Now send exactly 1000 GBX in a single transaction to the account address you generated in the previous step. This may be sent from another wallet, or from funds already held in your current wallet. Once the transaction is complete, view the transaction in a [blockchain explorer](#) by searching for the address. You will need 15 confirmations before you can start the masternode, but you can continue with the next step at this point already: installing GoByte Core on your VPS.

Install GoByte Core

GoByte Core is the software behind both the GoByte Core GUI wallet and GoByte masternodes. If not displaying a GUI, it runs as a daemon on your VPS (gobyted), controlled by a simple command interface (gobyte-cli).

Open PuTTY or a console again and connect using the username and password you just created for your new, non-root

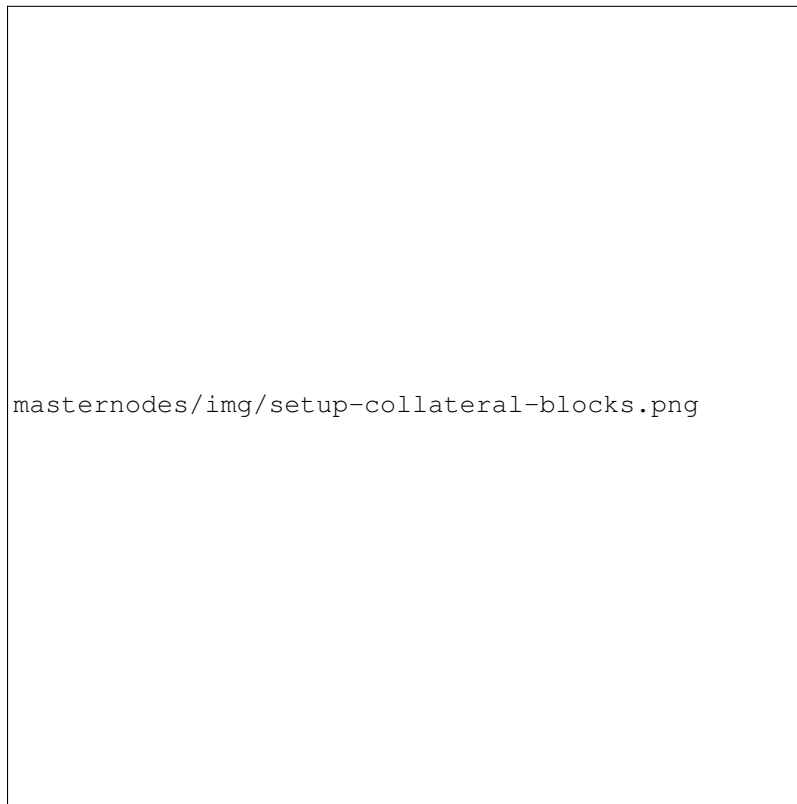


Fig. 193: Trezor blockchain explorer showing 15 confirmations for collateral transfer

user. There are two options to install GoByte Core, an automated option using a script utility called gobyteman by GoByte Core Team member sirbound, and a more complicated option which will allow you to understand all of the key steps involved in preparing your masternode.

Option 1: Automated installation using gobyteman

To install GoByte using gobyteman, enter the following commands after logging in:

```
cd ~
git clone https://github.com/gobytecoin/gobyteman
~/gobyteman/gobyteman install
```

(press **Y** and **Enter** to confirm)

gobyteman will download the latest version of GoByte Core for your system, as well as an initial snapshot of the blockchain to speed up the bootstrapping process. Next download and install sentinel, which is required for masternodes at version 12.1 or higher:

```
~/gobyteman/gobyteman install sentinel
```

Your system is now running as a standard GoByte node, and is busy completing synchronisation with the blockchain. We now need to enter the masternode private key generated in the previous step. Edit the configuration file using the following command:

```
nano ~/.gobytecore/gobyte.conf
```

Uncomment the last two lines by deleting the # symbol at the start of the line, then paste the masternode private key you generated after `masternodeprivkey=`. You can simply click the right mouse button to paste into the terminal window. Press **Ctrl + X** to close the editor and **Y** and **Enter** save the file.

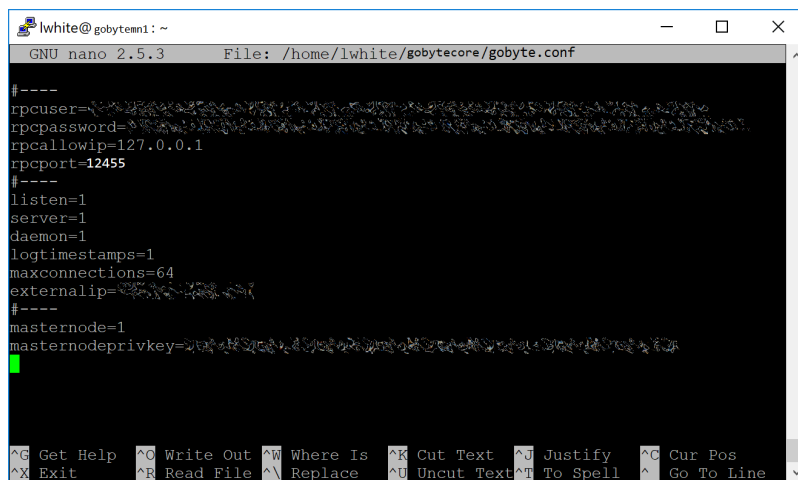


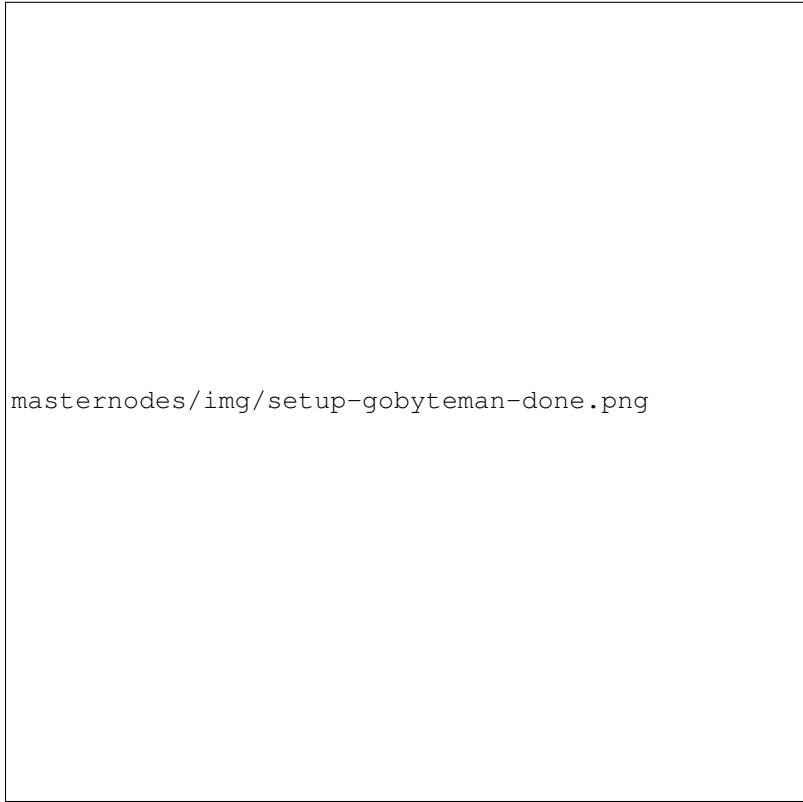
Fig. 194: Entering masternodeprivkey in gobyte.conf on the masternode

At this point you should restart gobyte to load the new configuration file by typing the following:

```
~/gobyteman/gobyteman restart
```

Press **Y** and **Enter** to confirm. Then check the sync status and wait until all blockchain synchronisation and the 15 confirmations for the collateral transaction are complete:

```
~/gobyteman/gobyteman status
```



```
masternodes/img/setup-gobyteman-done.png
```

Fig. 195: gobyteman status output showing masternode ready to be started

gobyteman does not automatically restart your masternode in the event of a system error. Add a check function to crontab to make sure it checks every minute to ensure your masternode is still running:

```
crontab -e
```

Choose nano as your editor and enter the following line at the end of the file, after the line for sentinel:

```
* * * * * pidof gobyted || ~/.gobytecore/gobyted
```

Press enter to make sure there is a blank line at the end of the file, then press **Ctrl + X** to close the editor and **Y** and **Enter** save the file.

Continue with the *next step to start your masternode*.

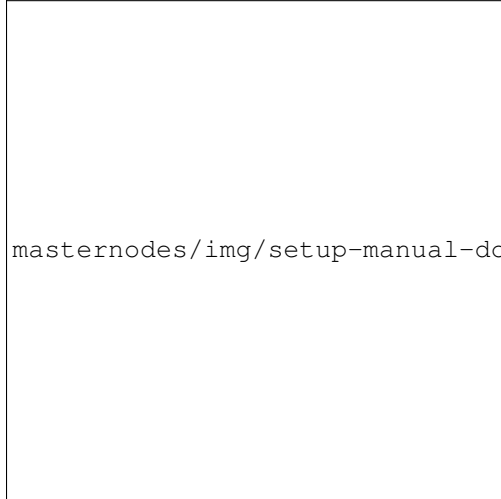
Option 2: Manual installation

To manually download and install the components of your GoByte masternode, visit <https://www.gobyte.network/wallets> on your computer to find the link to the latest GoByte Core wallet. Click **Linux**, then right-click on **Download TGZ for GoByte Core Linux 64 Bit** and select **Copy link address**. Go back to your terminal window and enter the following command, pasting in the address to the latest version of GoByte Core by right clicking or pressing **Ctrl + V**:

```
cd ~
wget https://github.com/gobytecoin/gobyte/releases/download/v0.12.2.4/GoByteCore-0.12.
↪2.4_Linux64.tar.gz
```

You can optionally verify the integrity of your download by running the following command and comparing the output against the value for the file as shown on the GoByte website under **Hash File**:

```
sha256sum GoByteCore-0.12.2.4_Linux64.tar.gz
```



masternodes/img/setup-manual-download.png

Fig. 196: Link to the hash file to verify download integrity

You can also optionally verify the authenticity of your download as an official release by GoByte Core Team. All releases of GoByte are signed using GPG by Antonio Moratti with the key 9ED8 A2B0 A016 6C55, [verifiable here on Keybase](#). Import the key, download the ASC file for the current release of GoByte and verify the signature as follows:

```
curl https://keybase.io/antoniomoratti/pgp_keys.asc | gpg --import
wget https://github.com/gobytecoin/gobyte/releases/download/v0.12.2.4/SHA256SUMS.asc
gpg --verify SHA256SUMS.asc
```

Create a working directory for GoByte, extract the compressed archive, copy the necessary files to the directory and set them as executable:

```
mkdir .gobytecore
tar xfvz GoByteCore-0.12.2.4_Linux64.tar.gz
cp GoByteCore-0.12.2.4/bin/gobyted .gobytecore/
cp GoByteCore-0.12.2.4/bin/gobyte-cli .gobytecore/
chmod 777 .gobytecore/gobyte*
```

Clean up unneeded files:

```
rm GoByteCore-0.12.2.4_Linux64.tar.gz
rm -r GoByteCore-0.12.2.4/
```

Create a configuration file using the following command:

```
nano ~/.gobytecore/gobyte.conf
```




Fig. 197: Downloading the PGP key and verifying the signed binary

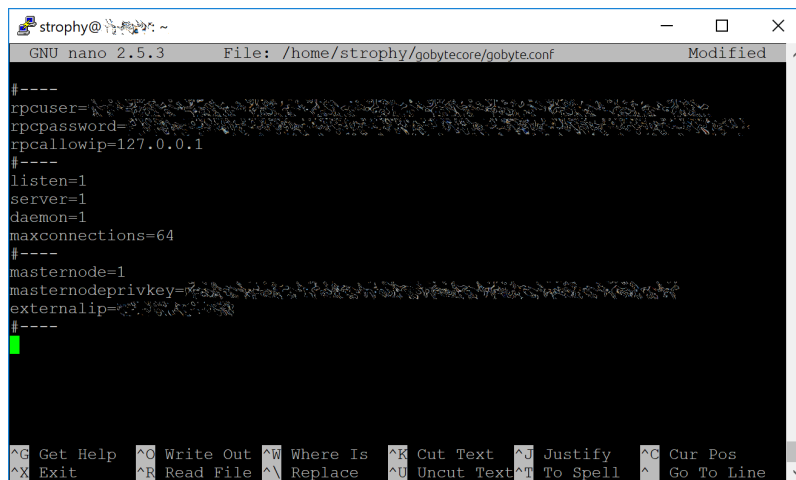
An editor window will appear. We now need to create a configuration file specifying several variables. Copy and paste the following text to get started, then replace the variables specific to your configuration as follows:

```
#----
rpcuser=XXXXXXXXXXXXX
rpcpassword=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
rpcallowip=127.0.0.1
#----
listen=1
server=1
daemon=1
maxconnections=64
#----
masternode=1
masternodeprivkey=XXXXXXXXXXXXXXXXXXXXX
externalip=XXX.XXX.XXX.XXX
#----
```

Replace the fields marked with XXXXXXXX as follows:

- `rpcuser`: enter any string of numbers or letters, no special characters allowed
- `rpcpassword`: enter any string of numbers or letters, no special characters allowed
- `masternodeprivkey`: this is the private key you generated in the previous step
- `externalip`: this is the IP address of your VPS

The result should look something like this:



```
strophy@ ~
GNU nano 2.5.3 File: /home/strophy/gobytecore/gobyte.conf Modified
#----
rpcuser=XXXXXXXXXXXXX
rpcpassword=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
rpcallowip=127.0.0.1
#----
listen=1
server=1
daemon=1
maxconnections=64
#----
masternode=1
masternodeprivkey=XXXXXXXXXXXXXXXXXXXXX
externalip=XXX.XXX.XXX.XXX
#----
```

Fig. 198: Entering key data in `gobyte.conf` on the masternode

Press **Ctrl + X** to close the editor and **Y** and **Enter** save the file. You can now start running GoByte on the masternode to begin synchronization with the blockchain:

```
~/gobytecore/gobyted
```

You will see a message reading **GoByte Core server starting**. We will now install Sentinel, a piece of software which operates as a watchdog to communicate to the network that your node is working properly:

```
cd ~/gobytecore
git clone https://github.com/gobytecoin/sentinel.git
```

(continues on next page)

(continued from previous page)

```
cd sentinel
virtualenv venv
venv/bin/pip install -r requirements.txt
venv/bin/python bin/sentinel.py
```

You will see a message reading **gobyted not synced with network! Awaiting full sync before running Sentinel**. Add gobyted and sentinel to crontab to make sure it runs every minute to check on your masternode:

```
crontab -e
```

Choose nano as your editor and enter the following lines at the end of the file:

```
* * * * * cd ~/.gobytecore/sentinel && ./venv/bin/python bin/sentinel.py 2>&1 >>_
↪sentinel-cron.log
* * * * * pidof gobyted || ~/.gobytecore/gobyted
```

Press enter to make sure there is a blank line at the end of the file, then press **Ctrl + X** to close the editor and **Y** and **Enter** save the file. We now need to wait for 15 confirmations of the collateral transaction to complete, and wait for the blockchain to finish synchronizing on the masternode. You can use the following commands to monitor progress:

```
~/.gobytecore/gobyte-cli mnsync status
```

When synchronisation is complete, you should see the following response:

```
{
  "AssetID": 999,
  "AssetName": "MASTERNODE_SYNC_FINISHED",
  "Attempt": 0,
  "IsBlockchainSynced": true,
  "IsMasternodeListSynced": true,
  "IsWinnersListSynced": true,
  "IsSynced": true,
  "IsFailed": false
}
```

Continue with the next step to start your masternode.

Start your masternode

Depending on how you sent your masternode collateral, you will need to start your masternode with a command sent either by your hardware wallet or by GoByte Core wallet. Before you continue, you must ensure that your 1000 GBX collateral transaction has at least 15 confirmation, and that gobyted is running and fully synchronized with the blockchain on your masternode. See the previous step for details on how to do this. During the startup process, your masternode may pass through the following states:

- **MASTERNODE_SYNC**: This indicates the data currently being synchronised in the masternode
- **MASTERNODE_SYNC_FAILED**: Synchronisation could not complete, check your firewall and restart gobyted
- **WATCHDOG_EXPIRED**: Waiting for sentinel to restart, make sure it is entered in crontab
- **NEW_START_REQUIRED**: Start command must be sent from wallet
- **PRE_ENABLED**: Waiting for network to recognize started masternode
- **ENABLED**: Masternode successfully started

If your masternode does not seem to start immediately, do not arbitrarily issue more start commands. Each time you do so, you will reset your position in the payment queue.

Option 1: Starting from a hardware wallet

Go back to GMT and ensure that all fields are filled out correctly. Click **Lookup** to find the collateral TX ID for the transaction which transferred the collateral to the address if you were not able to do so earlier. Then click **Start Masternode using Hardware Wallet** and confirm the following two messages:



Fig. 199: GoByte Masternode Tool confirmation dialogs to start a masternode

At this point you can monitor your masternode using `gobyteman/gobyteman status`, by entering `~/ . gobytecore/gobyte-cli masternode status` or using the **Get status** function in GMT. You will probably need to wait around 30 minutes as the node passes through the `PRE_ENABLED` stage and finally reaches `ENABLED`. Give it some time, the final result should appear as follows:

At this point you can safely log out of your server by typing `exit`. Congratulations! Your masternode is now running.

Option 2: Starting from GoByte Core wallet

If you used an address in GoByte Core wallet for your collateral transaction, you now need to find the txid of the transaction. Click **Tools > Debug console** and enter the following command:



Fig. 200: gobyteman status output showing successfully started masternode

```
masternode outputs
```

This should return a string of characters similar to this:

```
{
"06e38868bb8f9958e34d5155437d009b72dff33fc28874c87fd42e51c0f74fdb" : "0",
}
```

The first long string is your transaction hash, while the last number is the index. We now need to create a file called *masternode.conf* for this wallet in order to be able to use it to issue the command to start your masternode on the network. Open a new text file in Notepad (or TextEdit on macOS, gedit on Linux) and enter the following information:

- **Label:** Any single word used to identify your masternode, e.g. MN1
- **IP and port:** The IP address and port (usually 12455) configured in the *gobyte.conf* file, separated by a colon (:)
- **Masternode private key:** This is the result of your `masternode genkey` command earlier, also the same as configured in the *gobyte.conf* file
- **Transaction hash:** The txid we just identified using `masternode outputs`
- **Index:** The index we just identified using `masternode outputs`

Enter all of this information on a single line with each item separated by a space, for example:

```
MN1 52.14.2.67:12455 GrxSr3fXpX3dZcU7CoiFuFWqeHYw83r28btCFfIHqf6zkMp1PZ4_
↪06e38868bb8f9958e34d5155437d009b72dff33fc28874c87fd42e51c0f74fdb 0
```

Save this file in the **GoByteCore** data folder on the PC running the GoByte Core wallet using the filename *masternode.conf*. You may need to enable **View hidden items** to view this folder. Be sure to select **All files** if using Notepad so you don't end up with a *.conf.txt* file extension by mistake. For different operating systems, the GoByteCore folder can be found in the following locations (copy and paste the shortcut text into the **Save** dialog to find it quickly):

Now close your text editor and also shut down and restart GoByte Core wallet. GoByte Core will recognize *masternode.conf* during startup, and is now ready to activate your masternode. Go to **Settings > Unlock Wallet** and enter your wallet passphrase. Then click **Tools > Debug console** again and enter the following command to start your masternode (replace MN1 with the label for your masternode):

```
masternode start-alias MN1
```

At this point you can go back to your terminal window and monitor your masternode using `gobyteman/gobyteman status`, by entering `~/.gobytecore/gobyte-cli masternode status` or using the **Get status** function in GMT. You will probably need to wait around 30 minutes as the node passes through the PRE_ENABLED stage and finally reaches ENABLED. Give it some time, the final result should appear as follows:

At this point you can safely log out of your server by typing `exit`. Congratulations! Your masternode is now running.

1.11.4 Maintenance

Masternodes require regular maintenance to ensure you do not drop off the payment queue. This includes promptly installing updates to GoByte, as well as maintaining the security and performance of the server. In addition, masternodes should vote on proposals and perform other tasks in the interest of the network and the value of the GoByte they hold.



Fig. 201: gobyteman status output showing successfully started masternode

How to update a masternode

Periodically, the GoByte Core development team will release updates to GoByte. Since normal nodes rely on them for services and copies of the blockchain, masternodes are expected to update to new versions of GoByte and Sentinel promptly. In some cases, hardware upgrades (more CPU, RAM or disk space, or even custom GPU/ASIC hardware) may also be necessary. Not updating will eventually result in your masternode being removed from the payment queue. If you run a hosted masternode, your host will take care of updates for you. If not, the method of updating depends on how you installed GoByte.

Minor version updates to GoByte (e.g. from 0.12.3.1 to 0.12.3.2) do not make changes to the protocol version, while major version updates (e.g. from 0.12.2.3 to 0.12.3.0) will usually increase the network protocol version. If the protocol version did not change, you DO NOT need to restart your masternode if you complete the update within 60 minutes. If the protocol version did change, you must issue a start command from your wallet. Do not send start commands to your masternode if not necessary, as it will send you to the back of the payment queue.

Option 1: Automated update using gobytelman

To update GoByte using gobytelman, log in to your server and enter the following commands:

```
~/gobytelman/gobytelman sync
~/gobytelman/gobytelman update
```

Check the status of your masternode:

```
~/gobytelman/gobytelman status
```

If you are doing a major version update and need to restart your masternode, update the software version of the wallet holding the collateral to the latest version now by following the instructions [here](#). Continue monitoring your masternode. After some time, all statuses should turn green, in particular **masternode started: YES** and **masternode network state: ENABLED**.

Option 2: Manual update

To update GoByte manually, log in to your server using ssh or PuTTY. First we need to stop GoByte running:

```
~/gobytecore/gobyte-cli stop
```

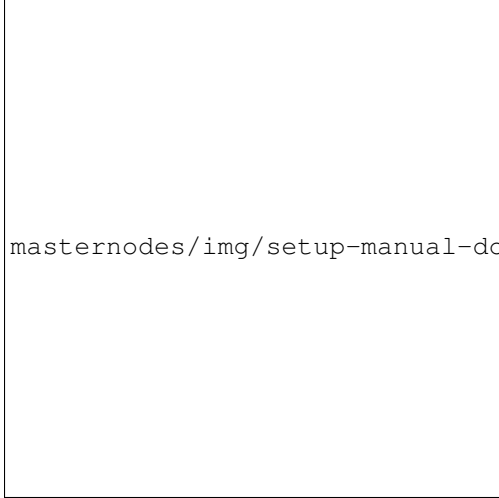
To manually download and install the components of your GoByte masternode, visit <https://www.gobyte.network/wallets/> on your computer to find the link to the latest GoByte Core wallet. Click **Linux**, then right-click on **Download TGZ for GoByte Core Linux 64 Bit** and select **Copy link address**. Go back to your terminal window and enter the following command, pasting in the address to the latest version of GoByte Core by right clicking or pressing **Ctrl + V**:

```
cd ~
wget https://github.com/gobytecoin/gobyte/releases/download/v0.12.2.4/GoByteCore-0.12.
↪2.4_Linux64.tar.gz
```

Verify the integrity of your download by running the following command and comparing the output against the value for the file as shown on the GoByte website under **Hash File**:

```
sha256sum GoByteCore-0.12.2.4_Linux64.tar.gz
```

Remove the old binaries from the working directory, extract the compressed archive, copy the new files to the directory and set them as executable:



masternodes/img/setup-manual-download.png

Fig. 202: Link to the hash file to verify download integrity

```
rm ~/.gobytecore/gobyted
rm ~/.gobytecore/gobyte-cli
tar xfvz GoByteCore-0.12.2.4_Linux64.tar.gz
cp GoByteCore-0.12.2.4_Linux64/bin/gobyted ~/.gobytecore/
cp GoByteCore-0.12.2.4_Linux64/bin/gobyte-cli ~/.gobytecore/
```

Clean up unneeded files:

```
rm GoByteCore-0.12.2.4_Linux64.tar.gz
rm -r GoByteCore-0.12.2.4_Linux64/
```

Restart GoByte:

```
~/.gobytecore/gobyted
```

You will see a message reading “GoByte Core server starting”. We will now update Sentinel:

```
cd ~/.gobytecore/sentinel/
git pull
```

If the protocol version changed during this update, you will need to issue a start command from your wallet. If you are using a hardware wallet, you can issue the start command by simply clicking the button in GMT. If you are using GoByte Core wallet, update it to the latest version, then open the debug console and enter this command, where MN1 is the alias for your masternode:

```
masternode start-alias MN1
```

Monitor the status of your masternode as it starts up:

```
~/.gobytecore/gobyte-cli getblockcount
~/.gobytecore/gobyte-cli getnetworkinfo
~/.gobytecore/gobyte-cli mnsync status
~/.gobytecore/gobyte-cli masternode status
```

In particular, the last command should return the status **Masternode successfully started**. If you see an error similar to **Invalid protocol version**, then the protocol version has changed and you must send a start command from your

wallet again. You can also monitor the status of your masternode from Sentinel. If Sentinel detects a functioning masternode, the following command should return nothing:

```
cd ~/.gobytecore/sentinel
venv/bin/python bin/sentinel.py
```

Finally, you can check for your masternode by its collateral address using [GoByteNinja](#), or search the consensus list of masternodes using this command and entering your masternode IP address:

```
~/.gobytecore/gobyte-cli masternode list full | grep <your ip address>
```

Payment withdrawals

Once your masternode has been accepted by the network, it will enter the masternode payment queue and slowly begin moving up. A masternode within the top 10% of the list is selected and receives a payment each time a new GoByte block is mined. For more details on this process, see [here](#). These payments are sent to the same address you used to start your masternode, which means you need to be careful when withdrawing the payments. The original 1000 GBX payment you used to start your masternode must remain untouched in a single unspent transaction output (utxo) or your masternode will drop off the payment list - you may have seen this ID when preparing to send the start masternode command. Payments appear in separate UTXOs, so we need a method of only spending those UTXOs and not the one containing the 1000 GBX. Note that masternode payouts can only be spent after 100 confirmations.


Option 1: Withdrawals using a hardware wallet

If you used a hardware wallet such as Trezor to start your masternode, you must also use this process to make payout withdrawals. Once again, we will be using SirBond's GoByte Masternode Tool (GMT) to select the correct outputs. With GMT, we can select specific UTXOs to withdraw the payments without touching the original collateral transaction. This is not possible using the Trezor web wallet alone.

Open GMT and verify the RPC and HW connections are working. From the **Tools** menu, select **Transfer funds from current masternode's address** or **Transfer funds from all masternode's addresses**, if you use GMT to control multiple masternodes.

GMT will load for a moment, then display a window showing the available UTXOs you can use in your withdrawal. By default, all UTXOs not used as masternode collateral are checked. The masternode collateral UTXOs are not only unchecked but also hidden in order to avoid unintentionally sending funds associated with collateral and stopping your masternode. You can show these hidden entries by unchecking the **Hide collateral utxos** option. Enter your destination address for the transaction. The window should appear as follows:

Verify the transaction fee and click **Send**. Your Trezor will prompt to enter your PIN and confirm the transaction on the device. Once this is done, confirm one more time to GMT that you want to broadcast the transaction to the network by clicking Yes. A confirmation with the transaction ID will appear.



masternodes/img/maintenance-gmt-broadcast.png



Fig. 203: Selecting the transfer funds function in GMT



Fig. 204: Selecting the UTXOs to use as inputs in the withdrawal transaction



Fig. 205: Confirming broadcast of the transaction to the network

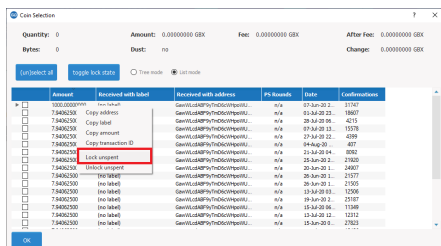
Option 2: Withdrawals from GoByte Core wallet

Similar to GMT as described above, we need a method in GoByte Core wallet to restrict which UTXOs are spent when making withdrawals from a masternode address to ensure that the collateral UTXO is not touched. In GoByte Core wallet, this feature is known as Coin Control, and it must be enabled before you can use it. Click **Settings > Options > Wallet > Enable coin control features**. Now, when you go to the **Send** tab in your wallet, a new button labelled **Inputs...** will appear. Click this button to select which UTXOs can be used as input for any transactions you create. The following window appears:



Fig. 206: Coin Selection window in GoByte Core wallet, showing two masternodes (testnet)

Right click on the transaction(s) showing an amount of 1000 GBX, then select **Lock unspent**. A small lock will appear next to the transaction. Then click **OK**. You can now safely create transactions with your remaining funds without affecting the original collateral UTXOs.



Finding your position in the payment queue

Prior to GoByte 0.12.4, each masternode may have a slightly different view of the network. In addition, selection from the top 10% of masternodes in the list in the *selection pool* is random. This means that there is no definite answer to


	Amount	Received with label
<input type="checkbox"/> 	1000.00000000	(no label)
<input type="checkbox"/>	7.94062500	(no label)
<input type="checkbox"/>	7.94062500	(no label)
<input type="checkbox"/>	7.94062500	(no label)
<input type="checkbox"/>	7.94062500	(no label)
<input type="checkbox"/>	7.94062500	(no label)
<input type="checkbox"/>	7.94062500	(no label)

Fig. 207: Locking UTXOs in GoByte Core wallet

when your masternode will be selected for payment. However, it is possible to make an approximation based on the time your masternode entered the back of the queue.

Community members erfano07p, SirBond and marksgnl have all published various tools you can run to determine your approximate position in the payment queue, and bots are available on Discord and Slack as well.

- mn_queue: <https://github.com/#>
- GoByteman: <https://github.com/gobytecoin/gobyteman>
- GMT: <https://github.com/gobytecoin/gobyte-masternode-tool>

GoByteCentral voting, verification and monitoring

GoByteCentral is a community-supported website managed by community member Rango. It has become a *de facto* site for discussion of budget proposals and to facilitate voting from a graphical user interface, but also offers functions to monitor masternodes.

Adding your masternode to GoByteCentral

GoBytecentral allows you to vote on proposals from the comfort of your browser. After completing [registration](#), go to the [masternodes](#) page and click the **Add masternode now** button. Enter your collateral address on the following screen:

Click **Add masternode**. Your masternode has now been added to GoByteCentral.

Enabling voting from GoByteCentral

Click **Edit** under **Voting privkeys** to enter your masternode private key to enable voting through the GoByteCentral web interface. Enter a voting passphrase (not the same as your login password, but equally important to remember!) and enter the private key (the same key you used in the gobyte.conf file on your masternode) on the following screen:

It is important to note that the private key to start your masternode is unrelated to the private keys to the collateral address storing your 1000 GBX. These keys can be used to issue commands on behalf of the masternode, such as voting, but cannot be used to access the collateral. The keys are encrypted on your device and never stored as plain text on GoByteCentral servers. Once you have entered the key, click **Store encrypted voting privkeys on server**. You can now vote on proposals from the GoByteCentral web interface.

Verifying ownership

You can also issue a message from your address to verify ownership of your masternode to GoByteCentral. Click **Unverified** under **Ownership** and the following screen will appear:

Instructions on how to sign your collateral address using a software wallet appear. If you are using a hardware wallet other than Trezor, you will need to use the GMT app to sign the address. If you are using the Trezor hardware wallet,



Fig. 208: Adding a masternode to GoByteCentral



Fig. 209: Adding voting privkeys to GoByteCentral



Fig. 210: Verifying ownership of your masternode to GoByteCentral

go to your [Trezor wallet](#), copy the collateral address and click **Sign & Verify**. The following screen will appear, where you can enter the message provided by GoByteCentral and the address you wish to sign:



Fig. 211: Signing a message from the Trezor Wallet

Click **Sign**, confirm on your Trezor device and enter your PIN to sign the message. A message signature will appear in the **Signature** box. Copy this signature and paste it into the box on GoByteCentral and click **Verify ownership**. Verification is now complete.

Installing the GoByteCentral monitoring script

GoByteCentral offers a service to monitor your masternode, automatically restart gobyted in the event of a crash and send email in the event of an error. Go to the [Account settings](#) page and generate a new API key, adding a PIN to your account if necessary. Scroll to the following screen:

Copy the link to the current version of the gobytecentral script by right- click and selecting **Copy link address**. Open PuTTY and connect to your masternode, then type:

```
wget https://www.gobytecentral.network/downloads/gobytecentral-updater-v6.tgz
```

Replace the link with the current version of gobytecentral-updater as necessary. Decompress the archive using the following command:

```
tar xvzf gobytecentral-updater-v6.tgz
```

View your masternode configuration details by typing:



Fig. 212: Masternode ownership has been successfully verified



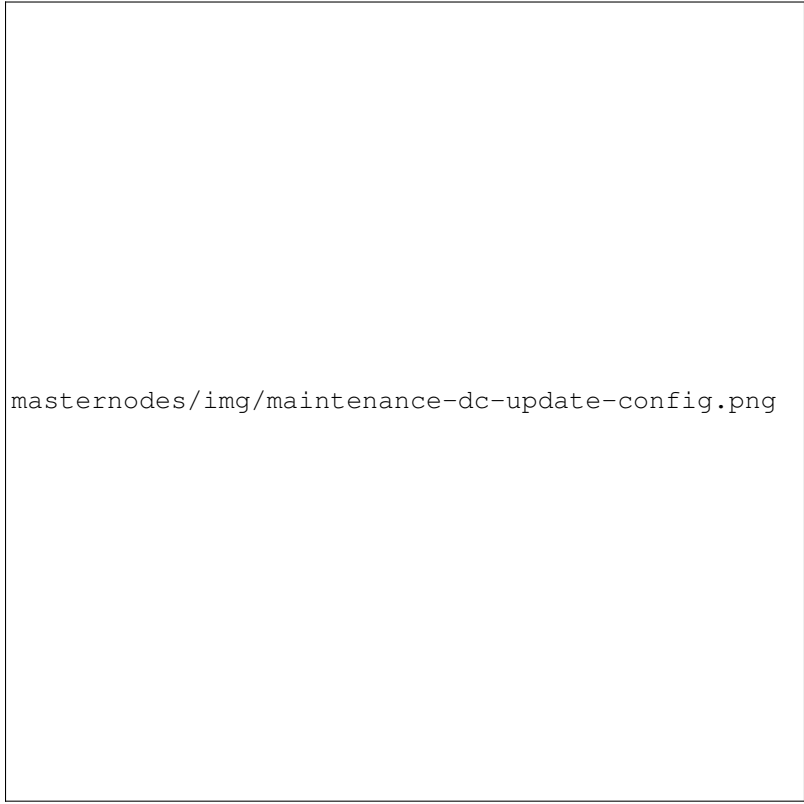
Fig. 213: Setting up the GoByteCentral monitoring script

```
cat .gobytecore/gobyte.conf
```

Copy the values for `rpcuser` and `rpcpassword`. Then edit the gobytecentral configuration by typing:

```
nano gobytecentral-updater/gobytecentral.conf
```

Replace the values for `api_key`, your masternode collateral address, `rpc_user`, `rpc_password`, `daemon_binary` and `daemon_datadir` according to your system. A common configuration, where `lwhite` is the name of the Linux user, may look like this:



masternodes/img/maintenance-dc-update-config.png

Fig. 214: GoByteCentral updater configuration file

```
#####
# gobytecentral-updater configuration
#####

our %settings = (
    # Enter your GoByteCentral api key here
    'api_key' => 'api_key_from_gobytecentral'
);

our %masternodes = (
    'masternode_collateral_address' => {
        'rpc_host'          => 'localhost',
        'rpc_port'          => 12455,
        'rpc_user'          => 'rpc_user_from_gobyte.conf',
        'rpc_password'      => 'rpc_password_from_gobyte.conf',
        'daemon_autorestart' => 'enabled',
    },

```


(continues on next page)

(continued from previous page)

```
'daemon_binary'    => '/home/<username>/.gobytecore/gobyted',
'daemon_datadir'   => '/home/<username>/.gobytecore'
}
);
```

Press **Ctrl + X** to exit, confirm you want save with **Y** and press **Enter**. Test your configuration by running the `gobytecentral` script, then check the website. If it was successful, you will see that an update has been sent:

```
gobytecentral-updater/dcupdate
```



```
masternodes/img/maintenance-dc-update.png
```

Fig. 215: Manually testing the GoByteCentral updater

Once you have verified your configuration is working, we can edit the crontab on your system to schedule the `dcupdate` script to run every 2 minutes. This allows the system to give you early warning in the event of a fault and will even restart the `gobyted` daemon if it hangs or crashes. This is an effective way to make sure you do not drop off the payment queue. Type the following command:

```
crontab -e
```

Select an editor if necessary and add the following line to your crontab after the line for `sentinel`, replacing `lwhite` with your username on your system:

```
*/2 * * * * /home/lwhite/gobytecentral-updater/dcupdate
```

Press **Ctrl + X** to exit, confirm you want save with **Y** and press **Enter**. The `dcupdate` script will now run every two minutes, restart `gobyted` whenever necessary and email you in the event of an error.



Fig. 216: GoByteCentral updater has successfully sent data to the GoByteCentral site



masternodes/img/maintenance-dc-crontab.png

Fig. 217: Editing crontab to run the GoByteCentral updater automatically

Masternode monitoring tools

Several sites operated by community members are available to monitor key information and statistics relating to the masternode network.

Block Explorers

Since GoByte is a public blockchain, it is possible to use block explorers to view the balances of any GoByte public address, as well as examine the transactions entered in any given block. Each unique transaction is also searchable by its txid. A number of block explorers are available for the GoByte network.

- [CryptoID](#) offers a [GoByte blockchain explorer](#) and a [function](#) to view and map GoByte masternodes.
- [BitInfoCharts](#) offers a [page](#) of price statistics and information and a [blockchain explorer](#).
- [CoinCheckup](#) offers a range of statistics and data on most blockchains, including GoByte.
- [GoByte.Network](#) includes two blockchain explorers at [explorer.gobyte.network](#) and [insight.gobyte.network](#).
- [Trezor](#) operates a [blockchain explorer](#) powered by a [GoByte fork](#) of [insight](#), an advanced blockchain API tool

GoByte Masternode Tool

<https://github.com/gobytecoin/gobyte-masternode-tool>

Written and maintained by community member SirBond, GoByte Masternode Tool (GMT) allows you to start a masternode from all major hardware wallets such as Trezor, Ledger and KeepKey. It also supports functions to vote on proposals and withdraw masternode payments without affecting the collateral transaction.

GBX Ninja

<https://masternodes.gobyte.network/>

GBX Ninja, operated by forum member and GoByte Core developer SirBond, offers key statistics on the adoption of different versions of GoByte across the masternode network. Several features to monitor governance of the GoByte, the masternode payment schedule and the geographic distribution of masternodes are also available, as well as a simple blockchain explorer.

GoByteCentral

<https://www.gobytecentral.network>

GoByteCentral, operated by forum member marksgnl, offers an advanced service to monitor masternodes and vote on budget proposals through an advanced web interface. An *Android app* will also be available.

1.11.5 Advanced Topics

Installing GoByte on Fedora Linux

Dash developer t0dd has developed packages and written an excellent guide on installing and running Dash as a node, masternode or on testnet. (these can be adapted for GoByte without any effort)

- <https://github.com/taw00/dashcore-rpm>

Installing GoByte on Ubuntu Linux

GoByte binaries are under development for distribution through the Ubuntu Linux Launchpad repository system. Check back here for details once a release announcement is made.

1.12 Mining

Mining in the context of cryptocurrency such as gobyte refers to the process of searching for solutions to cryptographically difficult problems as a method of securing blocks on the blockchain. The process of mining creates new currency tokens as a reward to the miner. Mining is possible on a range of hardware. gobyte implements an algorithm known as *neoscrypt*, which the miner must solve in order to earn rewards.

The simplest and most general hardware available for mining is the general purpose CPU present in every computer. A CPU is designed to be versatile but offers less efficiency than a GPU, which is designed to rapidly calculate millions of vectors in parallel. While specific CPU instruction enhancements related to cryptography such as AES or AVX can provide a decent boost, GPUs offer a significant performance increase due to their multiple pipelines capable of processing the predictably repetitive calculations associated with cryptocurrency mining. Finally, ASICs are relatively inflexible and can only process the specific function(s) for which they were designed, but at an even faster rate than the more general purpose GPUs and CPUs. A number of neoscrypt ASICs are now available on the market, which have quickly made CPU and GPU mining uneconomic due to the increased difficulty of hashing arising from the rapidly increasing hash rate. The result is a currency which is more secure against brute force attacks on the gobyte blockchain.

The profitability of mining is determined by the hashrate of your mining device, the current network difficulty and the costs of your hardware and electricity. The following links provide up to date information:

- [Hashrate](#)
- [Mining difficulty](#)
- [Profitability calculation tool](#)

1.12.1 Masternodes vs. Mining

GoByte, like Bitcoin and most other cryptocurrencies, is based on a decentralized ledger of all transactions, known as a blockchain. This blockchain is secured through a consensus mechanism; in the case of both GoByte and Bitcoin, the consensus mechanism is Proof of Work (PoW). Miners attempt to solve difficult problems with specialized computers, and when they solve the problem, they receive the right to add a new block to the blockchain. If all the other people running the software agree that the problem was solved correctly, the block is added to the blockchain and the miner is rewarded.

GoByte works a little differently from Bitcoin, however, because it has a two-tier network. The second tier is powered by *masternodes* (Full Nodes), which enable financial privacy (PrivateSend), instant transactions (InstantSend), and the decentralized governance and budget system. Because this second tier is so important, masternodes are also rewarded when miners discover new blocks. The breakdown is as follows: 25% of the block reward goes to the miner, 65% goes to masternodes, and 10% is reserved for the budget system (created by superblocks every month).

The masternode system is referred to as Proof of Service (PoSe), since the masternodes provide crucial services to the network. In fact, the entire network is overseen by the masternodes, which have the power to reject improperly formed blocks from miners. If a miner tried to take the entire block reward for themselves or tried to run an old version of the gobyte software, the masternode network would orphan that block, and it would not be added to the blockchain.

In short, miners power the first tier, which is the basic sending and receiving of funds and prevention of double-spending. Masternodes power the second tier, which provide the added features that make gobyte different from other cryptocurrencies. Masternodes do not mine, and mining computers cannot serve as masternodes. Additionally, each

masternode is “secured” by 1000 GBX. Those gobyte remain under the sole control of their owner at all times, and can still be freely spent. The funds are not locked in any way. However, if the funds are moved or spent, the associated masternode will go offline and stop receiving rewards.

1.12.2 Mining Pools

Mining GoByte in pools is more likely to generate rewards than solo mining directly on the blockchain. Mining GoByte using P2Pool is strongly encouraged, since it is a good way to distribute, rather than centralize, the hashing power. The following site lists GoByte P2Pool mining pools near you, simply choose a pool with favourable fees and ping time and enter your GoByte payment address as username and anything as password.

- <http://www.p2poolmining.us/p2poolnodes/>

If you would like to set up your own P2Pool, documentation of the process is available [here](#) and the code for p2pool-gobyte is available on [GitHub](#). Other mining pools are listed below and may be advantageous for different reasons such as ping latency, uptime, fee, users, etc. A guide to using a typical mining pool can be found [here](#).

- <https://bsod.pw/en/pool/dashboard/GBX/>
- <https://unimining.net/site/mining?algo=neoscrypt>
- <https://prohashing.com/live.html>
- <https://www.mining-dutch.nl/pools/gobyte.php>
- <https://fairpool.pro/site/coins?coin=GBX>
- <https://www.ahashpool.com/>
- <http://blazepool.com/>
- <http://zergpool.com/site/mining?algo=neoscrypt>
- <https://gobyte.suprnova.cc>
- <https://www.nicehash.com>
- <https://gobyte.miningpoolhub.com>
- <https://www.multipool.us>
- <https://www.f2pool.com>
- <https://aikapool.com/gbx/>
- <http://zpool.ca>

DISCLAIMER: This list is provided for informational purposes only. Services listed here have not been evaluated or endorsed by the gobyte developers and no guarantees are made as to the accuracy of this information. Please exercise discretion when using third-party services. If you’d like to be added to this list please reach out to leon.white@gobyte.network

In addition to joining a pool, you will also need to create a gobyte address to receive your payout. To do this in gobyte Core wallet, see [here](#).

Mining Pools

Mining GoByte in pools is more likely to generate rewards than solo mining directly on the blockchain. This is because you are extremely unlikely to mine a block as an individual given the current mining difficulty, so pooling efforts results in more frequent incremental rewards than extremely rare large rewards.

Miner Hosting

Some miners may decide to host the machines directly in their homes. This can work if you have a good shed or basement. But keep in mind that FPGA machines can be intrusive in your home. They are quite loud when operating (think a vacuum cleaner). In addition, mining hardware consumes a very significant amount of electricity, so running multiple mining FPGAs in your home may damage your wiring or fuse board, may require additional ventilation or power and network distribution systems, and bears a risk of starting a fire. Ensure you are properly informed about these challenges before mining at home.

Depending on the above considerations and your electricity rate, it may make sense to send your mining machine to a mining colocation site for hosting on your behalf. The mining colocation business takes a “hosting fee” in return for setup, ensuring proper operations, electricity and ongoing maintenance.

There are many hosting options available in the US / Canada such as [Compute North](#), [Frontier](#), [Core Scientific](#), [Lightspeed](#), and many more. [Crypto Mining Tools](#) recently came out with a [mining map](#) with more options in North America. [F2Pool](#) has a [list of hosting locations](#) in China and the CIS region. [BC Mining](#) is an upcoming facility in Paraguay, South America. Please make sure to do your own research before making any decision on engaging a colocation business.

Setting up your miner

Powering up the Miner

Connect your machine to the power outlet via the built-in PSU. Then connect your miner to your internet-connected router or switch using a standard network cable. You are now ready to power up your miner.

Scanning for Devices

To get started, use a scanning tool like [AngryIP](#) or [Locator](#) to scan every device on your local network and see its IP address. After you have identified your miner’s IP address, you are ready to proceed with Configuration.

Configuration

Enter the local network IP address of your miner in the URL bar of any web browser. A dashboard similar to the one shown below will pop up.

Choosing a Mining Pool

To finish the configuration, you will need to choose a mining pool.

To help with the security of the network and decentralization we recommend using P2Pool. To set up your own P2Pool server, follow the setup documentation available [here](#). The code for p2pool-gobyte is [available on GitHub](#).

If you do not want to set up your own pool, you can check out a list of pools [here](#). In this guide we will set up a miner with [BSOD.pw](#), a international mining pool that operates based on the pay-per-share (PPS) payment method. Other pools can be found [here](#).

Completing Configuration

Once you have completed the [How to Start Tutorial](#) with the mining pool, you will have all the information you need to complete the setup process.

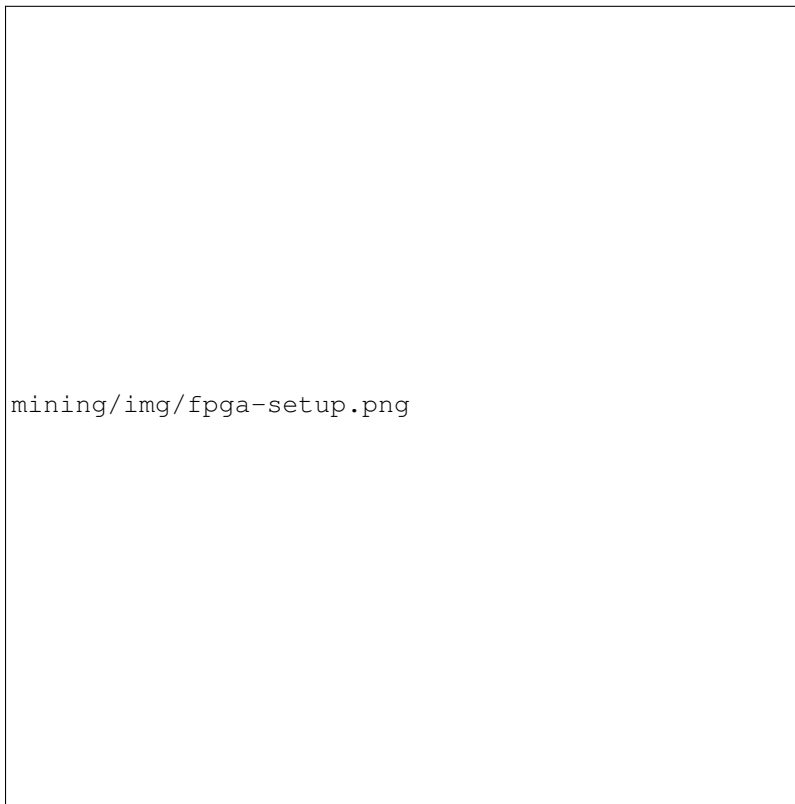


Fig. 218: Setting up a mining FPGA to use a pool

- URL: `stratum+tcp://pool.bsod.pw:1932`
- Worker: `WorkerName` Workername can be anything, but avoid using symbols or special characters as it may be invalid.
- Password: `123`

For the other nodes (Pool RU and Pool EU), feel free to use any of the other region nodes or other pools, such as `eu.bsod.pw:1932`.

Once you have filled out the details, click **Save & Apply**. Setup is now complete.

Monitoring Results

It will take about 5 minutes for your workers to appear on the stats page. To find your user, simply go to BSOD.pw, and navigate to the Wallet tab. You should see something like this:

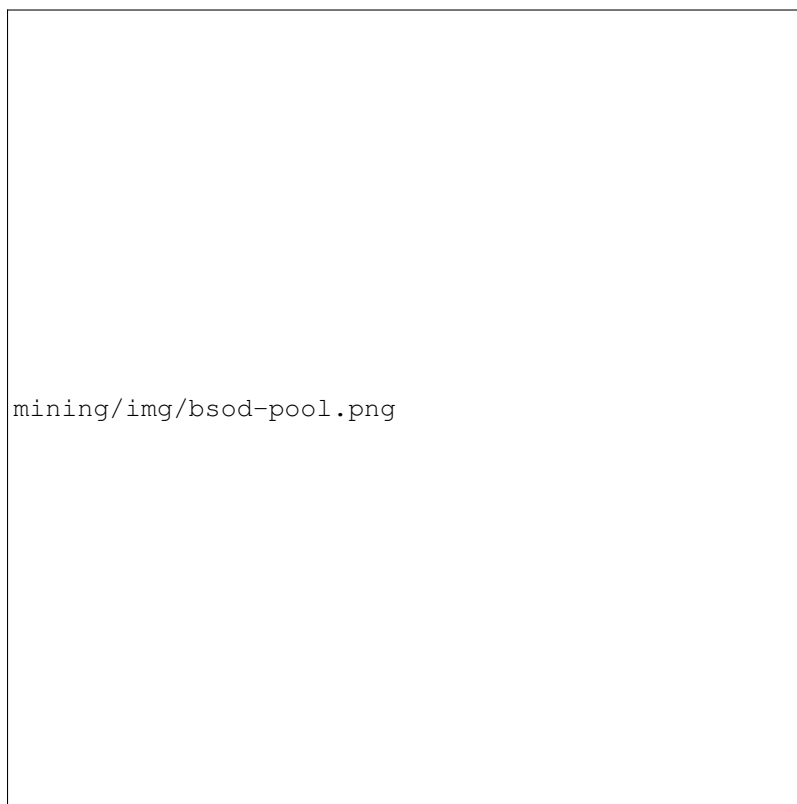



Fig. 219: Monitoring mining results with BSOD.pw pool

Note: For P2Pool, there is no central point to check balance. To learn more about P2Pool, see the [P2Pool documentation](#).

Setting up the GoByte Wallet

The last step is to set up your GoByte address where you will receive your miner payouts. To do this in the GoByte Core wallet, see [here](#).

Once you have your address, you can enter your address and set a custom payout threshold.



mining/img/bsod-threshold.png

Fig. 220: Setting the payout threshold with BSOD.pw pool

P2Pool Node Setup

This guide describes how to set up a GoByte P2Pool node to manage a pool of miners. Unlike centralized mining pools, P2Pool is based on the same peer-2-peer (P2P) model as GoByte, making the pool as a whole highly resistant to malicious attacks, and preserving and protecting the decentralized nature of GoByte. When you launch a P2Pool node, it seeks out, connects with, and shares data with a decentralized network of other P2Pool nodes (also known as peers). P2Pool nodes share a cryptographic chain of data representing value, similar to GoByte's blockchain. The P2Pool version is called the sharechain. The decentralized and fair nature of this mining model means mining with P2Pool is strongly encouraged. P2Pool for GoByte uses the [p2pool-gobyte](#) software on GitHub, which is a fork of p2pool for Bitcoin. For more information, see [here](#).

Because of the way P2Pool manages difficulty adjustments on the sharechain, it is important to maintain low latency between the miners and the P2Pool node to avoid miners submitting shares too late to enter the sharechain. When setting up your node, you need to consider its physical and network location relative to the miners you intend to connect to the node. If you operate a mining farm, your P2Pool node should probably be a physical machine on the same local network as your miners. If you plan to operate a public node, it may be best to set up your P2Pool node as a virtual machine in a data center with a high speed connection so geographically close miners can mine to your pool with relatively low latency.

This following section describes the steps to setup an Ubuntu Server running P2Pool for GoByte. It has been tested with Ubuntu 18.04 LTS and GoByte 0.13.1.0. While a reasonable effort will be made to keep it up to date, it should be possible to modify the instructions slightly to support different versions or operating systems as necessary.

Setting up the host server

Download a copy of Ubuntu Server LTS from <https://www.ubuntu.com/download/server> and install it on your system according to the steps described [here](#). If you are using a VPS such as Vultr or AWS, your provider will most likely provide an option to install this system during provisioning. Ensure you enable OpenSSH server during setup so you can control your server from a remote console. Once you have access to your server, create a new non-root user if you have not already done so using the following command, replacing `<username>` with a username of your choice:

```
adduser <username>
```

You will be prompted for a password. Enter and confirm using a new password (different to your root password) and store it in a safe place. You will also see prompts for user information, but this can be left blank. Once the user has been created, we will add them to the sudo group so they can perform commands as root:

```
usermod -aG sudo <username>
```

Reboot your server and log in as the new user. At this point it is recommended to connect remotely using **PuTTY** (for Windows) or **ssh** (for Linux and macOS) if you have not already done so.

Setting up port forwarding

If you are on a private network behind a router, you will need to set up port forwarding for at least port 8999 (UDP/TCP) for access to the sharechain, as well as port 7903 (UDP/TCP) if you want your node to be accessible to the public. How this is done depends on your particular network router and is therefore beyond the scope of this documentation. An example from the popular DD-WRT open source router distribution is shown below. Guides to setting up port forwarding can be found [here](#) and [here](#).

Take note of your IP address either from your router management interface or by visiting <https://www.whatismyip.com>

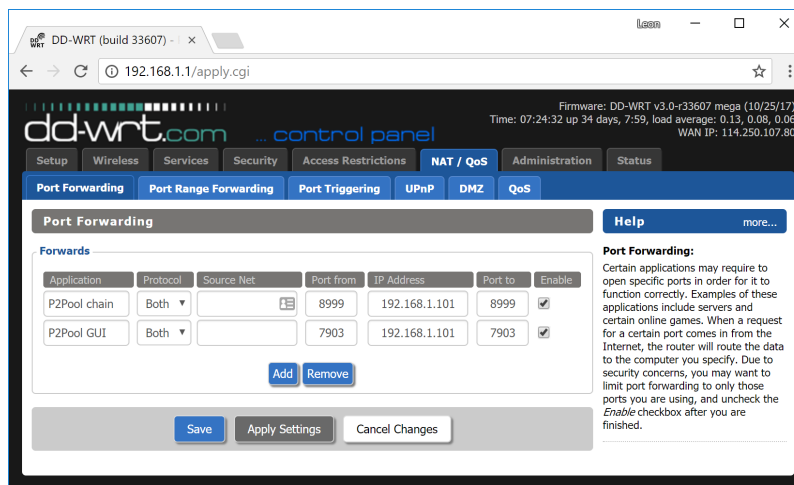


Fig. 221: Setting up port forwarding under DD-WRT

Option 1: Automated script setup

GoByte community member **sirbond** has generously donated a script to automatically deploy everything required to run a p2pool-gobyte node under Ubuntu Server 16.04 and higher. For more details, see [this forum post](#), or simply follow these instructions to get the script. To get fetch the script and get started, type:

```
sudo apt install git
git clone https://github.com/gobytecoin/p2pool-gobyte-deploy
```

The files will be created in the p2pool-gobyte-deploy folder. We now need to configure a few variables specific to your system:


```
nano ./p2pool-gobyte-deploy/p2pool.deploy.sh
```

Scroll down to the section labeled `#Variables` and enter the following information, replacing the `<xxx>` placeholders after the `=` sign. Note that it may also be necessary to update the `GBX_WALLET_URL`, `GBX_WALLET_ZIP` and `GBX_WALLET_LOCAL` values if they do not match the current version of GoByte:

- `PUBLIC_IP` = your public IP address from the previous step
- `EMAIL` = your email address
- `PAYOUT_ADDRESS` = your GBX wallet address to receive fees
- `USER_NAME` = linux user name
- `RPCUSER` = enter a random alphanumeric rpc user name
- `RPCPASSWORD` = enter a random alphanumeric rpc password

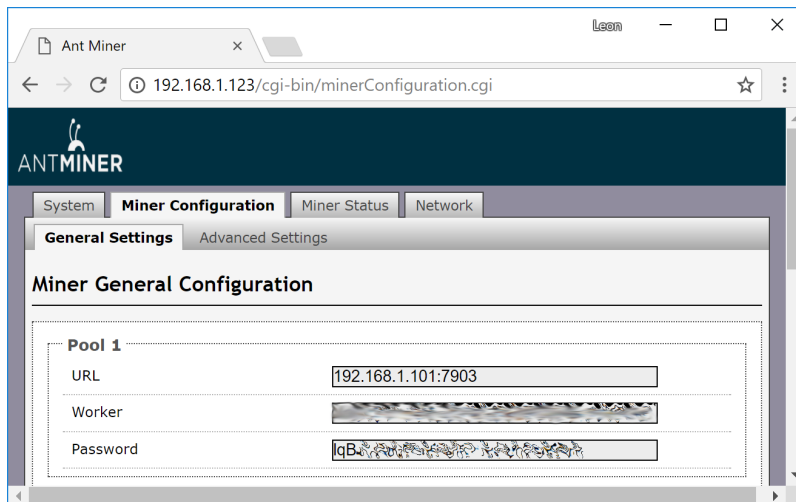
Press **Ctrl + X** to close the editor and **Y** and **Enter** save the file. Then run the script:

```
bash ./p2pool-gobyte-deploy/p2pool.deploy.sh
```

The script will carry out all steps necessary to set up P2pool on Ubuntu Server and start gobyted synchronisation. When setup is complete, you should see a message reading **Installation Completed**. You can now run a second script to start p2pool-gobyte:

```
bash ~/p2pool.start.sh
```

Your P2Pool node is now running. If you see errors similar to **Error getting work from gobyted** or **-10 GoByte Core is downloading blocks...** then you must wait until GoByte finishes synchronisation. Once this is done, you can point your miners to `<ip_address>:7903` to begin mining.



Option 2: Manual setup

First update your operating system as follows:

```
sudo apt update
sudo apt upgrade
```

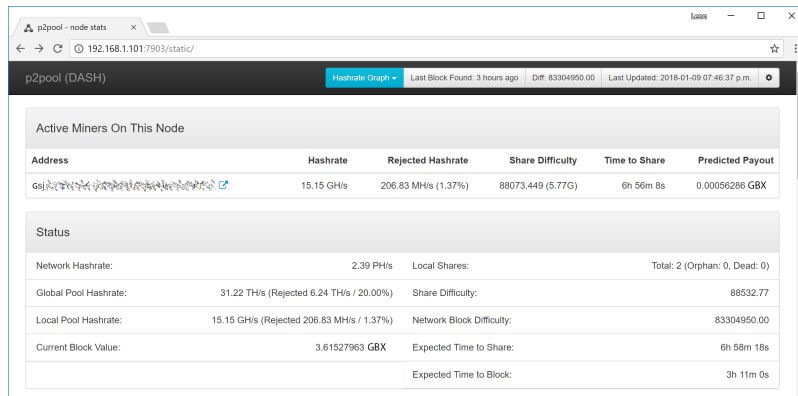


Fig. 222: Example configuration showing a single miner connected to a p2pool-gobyte node on the local network

Setting up gobyted

P2Pool requires a full GoByte node to be running to get block and transaction data. To download and install GoByte, visit <https://www.gobyte.network/downloads> on your computer to find the link to the latest GoByte Core wallet. Click **Linux**, then right-click on **Download TGZ** for **GoByte Core x64** and select **Copy link address**. Go back to your terminal window and enter the following command, pasting in the address to the latest version of GoByte Core by right clicking or pressing **Ctrl + V**:

```
cd ~
wget https://github.com/gobytecoin/gobyte/releases/download/v0.12.2.4/GoByteCore-0.12.2.4_Linux64.tar.gz
```

Verify the integrity of your download by running the following command and comparing the output against the value for the file as shown on the GoByte website under **Hash File**:

```
sha256sum GoByteCore-0.12.2.4_Linux64.tar.gz
```

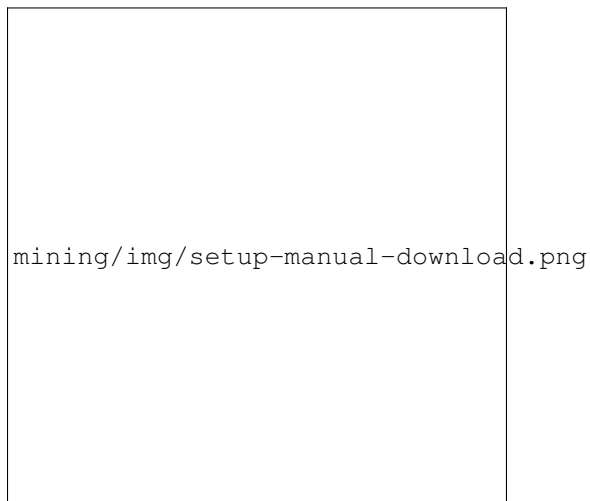


Fig. 223: Link to the hash file to verify download integrity

Create a working directory for GoByte, extract the compressed archive, copy the necessary files to the directory and set them as executable:

```
mkdir .gobytecore
tar xfvz GoByteCore-0.12.2.4_Linux64.tar.gz
cp GoByteCore-0.12.2.4/bin/gobyted .gobytecore/
cp GoByteCore-0.12.2.4/bin/gobyte-cli .gobytecore/
```

Clean up unneeded files:

```
rm GoByteCore-0.12.2.4_Linux64.tar.gz
rm -r GoByteCore-0.12.2.4/
```

Create a configuration file using the following command:

```
nano ~/.gobytecore/gobyte.conf
```

An editor window will appear. We now need to create a configuration file specifying several variables. Copy and paste the following text to get started, then replace the variables specific to your configuration as follows:

```
#----
rpcuser=XXXXXXXXXXXXX
rpcpassword=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
rpcallowip=127.0.0.1
#----
listen=1
server=1
daemon=1
maxconnections=64
#----
```

Replace the fields marked with XXXXXXXX as follows:

- rpcuser: enter any string of numbers or letters, no special characters allowed
- rpcpassword: enter any string of numbers or letters, no special characters allowed

The result should look something like this:

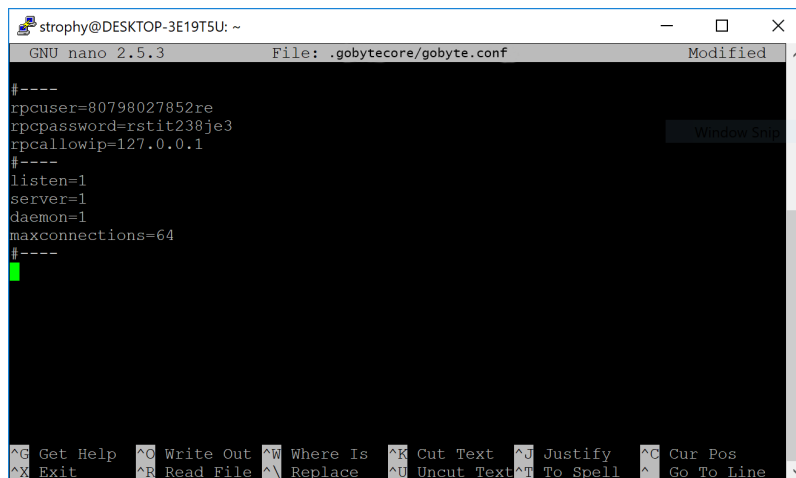


Fig. 224: Entering key data in gobyte.conf on the P2Pool node

Press **Ctrl + X** to close the editor and **Y** and **Enter** save the file. You can now start running GoByte on the masternode to begin synchronization with the blockchain:

```
~/gobytecore/gobyted
```

You will see a message reading **GoByte Core server starting**. You can continue with the following steps and check synchronization periodically using the following command. Synchronization is complete when the number of `blocks` is equal to the current number of blocks in the GoByte blockchain, as can be seen from any synchronized GoByte wallet or [block explorer](#):

```
~/gobytecore/gobyte-cli getblockcount
```

Setting up P2Pool

We will now set up the P2Pool software and its dependencies. Begin with the dependencies:

```
sudo apt install python-zope.interface python-twisted python-twisted-web python-dev_  
→ gcc g++ git
```

Create working directories and set up `p2pool-gobyte`:

```
mkdir git  
cd git  
git clone https://github.com/gobytecoin/p2pool-gobyte  
cd p2pool-gobyte  
git submodule init  
git submodule update  
cd gobyte_hash  
python setup.py install --user
```

We will add some optional extra interfaces to the control panel:

```
cd ..  
mv web-static web-static.old  
git clone https://github.com/justino/p2pool-ui-punchy web-static  
mv web-static.old web-static/legacy  
cd web-static  
git clone https://github.com/hardcpp/P2PoolExtendedFrontEnd ext
```

You can now start `p2pool` and optionally specify the payout address, external IP (if necessary), fee and donation as follows:

```
python ~/git/p2pool-gobyte/run_p2pool.py --external-ip <public_ip> -f <fee> --give-  
→ author <donation> -a <payout_address>
```

You can then monitor your node by browsing to the following addresses, replacing `<ip_address>` with the IP address of your P2Pool node:

- Punchy interface: http://ip_address:7903/static
- Legacy interface: http://ip_address:7903/static/legacy
- Status interface: http://ip_address:7903/static/status
- Extended interface: http://ip_address:7903/static/ext

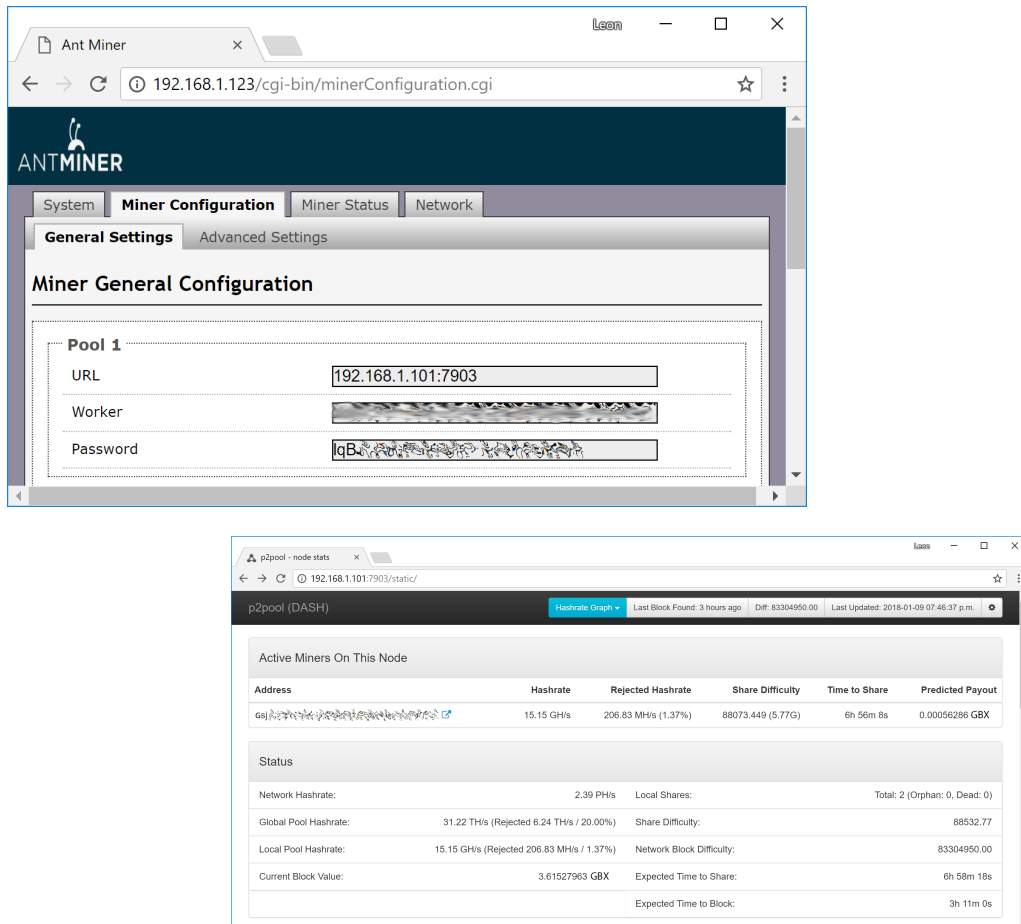


Fig. 225: Example configuration showing a single Bitmain Antminer D3 connected to a p2pool-gobyte node on the local network

1.12.3 CPU Mining

This documentation describes how to mine gobyte under the Windows operating system using just the CPU in your computer. Please note that the prevalence of GPU and ASIC miners mean that unless you have free electricity, this is highly unlikely to be profitable! Since this is the case, the software in this guide has not been updated in several years, and is intended for experimental purposes and testnet only.

This is a fairly simple procedure and examples will be given in order to achieve the fastest possible hash rate for your CPU, but remember that more optimized miners do exist, so we advise you to keep an eye out on mining sites such as these in order to keep up with the latest information and releases.

- [Crypto Mining Blog](#)
- [GoByte Forum Mining Discussions](#)
- [Bitcoin Talk Altcoin Mining Discussions](#)

Mining software

The first step is to download appropriate mining software. A good basic miner for modern CPUs can be found here:

- <https://github.com/tpruvot/cpuminer-multi>

- <https://github.com/JayDDee/cpuminer-opt/releases>

Our goal here is to choose mining software that supports the maximum possible instruction sets available on your CPU, and then try to increase the hash speed. Once you have made your choice, click **Releases** and download and extract the zip file. The different *.exe files indicate which specific processor optimizations they support. The folder should look something like this:



Fig. 226: Executable CPU miners for gobyte

Configuration

Begin by selecting a mining pool and generating a gobyte address as described in the *Mining Pools* section above. Keep all your mining files in a single folder. In this example we will work from the Desktop. The node selected for this example is from the BSOD.pw list and is located in US, EU, RU or ASIA:

```
https://bsod.pw/en/pool/dashboard/GBX/
```

Next, open **Notepad** and type in on one line the command we will use to start the miner, followed by pause on the second line. The general format is as follows:

```
<minerd> -a <algorithm> -o <url> -u <username> -p <password> -t <threads>  
pause
```

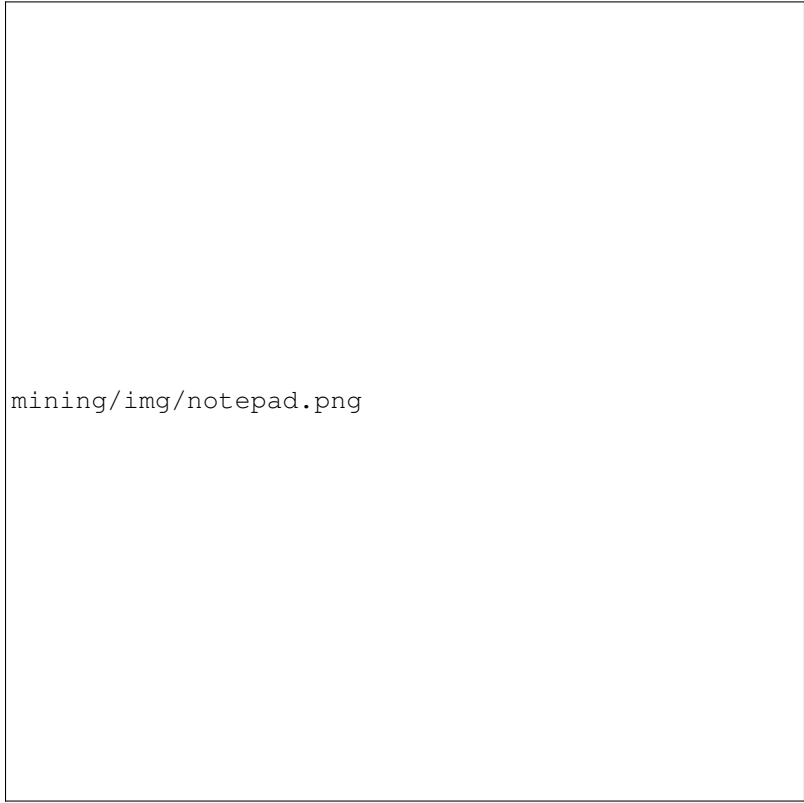
Where:

- minerd = the executable miner daemon file you choose to use
- a = algorithm, which is neoscrypt for gobyte
- o = URL of your mining pool, including the protocol and port

- `u` = username, usually the gobyte receiving address of your wallet or worker
- `p` = password, can often be set to `x`
- `t` = number of threads used
- `pause` = keeps the window open in the case of errors

For the CPU in the example above, the command may be (for EU):

```
cpuminer.exe -a neoscrypt -o stratum+tcp://eu.bsod.pw:1932 -u_
↪GM9ygYxt8HHp4vVHHndv4cLPXDMv6FFgdw -p x
pause
```



mining/img/notepad.png

Fig. 227: Notepad file showing an example command to start a CPU miner

Click **File**, then **Save As**. Change **Save as type** to **All Files**, then type the file name as *startminer.bat* and save it in the same folder as the unzipped *minerd* files.

Testing

You are now ready to start! Keep an eye on your CPU usage in **Task Manager** (right click the taskbar to open this) and be careful that the CPU temperature does not exceed your maximum rating (around 64°C). If you have temperature or desktop stability problems, reduce `t` to ~2 threads and try that first. If `t` is left out, the machine will default to the maximum number of threads. After running the miner for a while, take a look at the hash speed and payouts in your mining pool. You can identify your miner by the wallet address on the page.

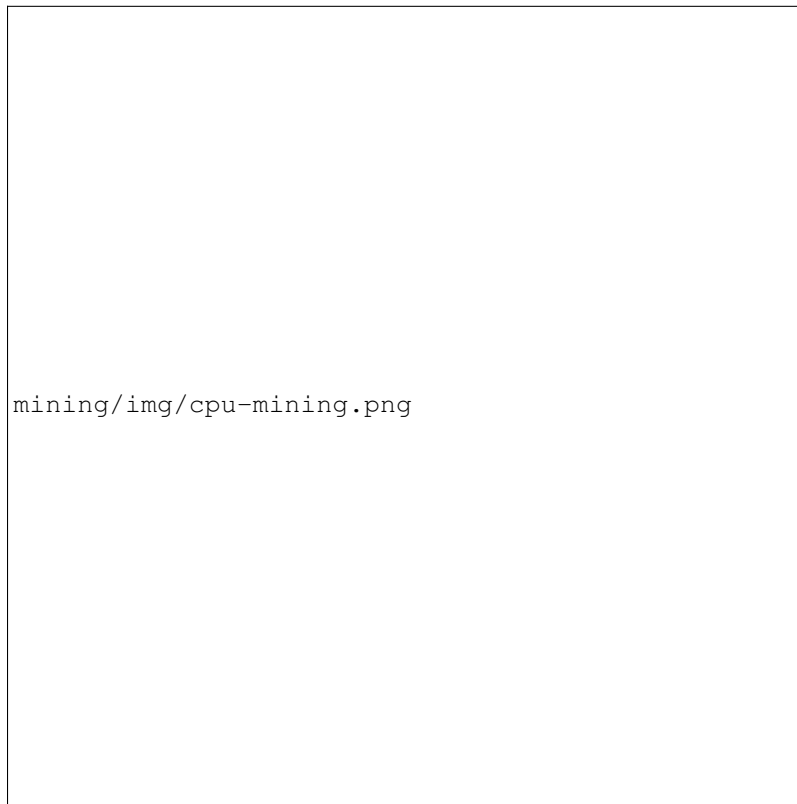


Fig. 228: Example of CPU mining using CPUMiner on Intel Core i7

Tips

Reduce the number of threads for added desktop usability and heat reduction. If the CPU temperature is too high, consider fitting a new fan and check that the heat sink thermal paste on the CPU is adequate. Tweak the processor clock speed for added performance using a motherboard controller like [AI Suite](#) for Asus motherboards. Reduction of CPU core voltage will result in lower temperature but increased instability.

Try to select a pool that is nearby to reduce network latency. If the node appears slow, switch to another location. Please distribute the hashing power globally to different pools to avoid forking.

1.12.4 GPU Mining

This guide consolidates several other guides on how to use your GPU (the processor on your graphics card) to mine gobyte using the neoscript algorithm on Windows. Please note that the growing market for ASIC miners means that this is probably not going to be profitable! A lot of the software and binaries described here also have not been updated for several years, so this guide should be used for experimental purposes only.

This guide will cover the process of downloading and configuring the mining software, followed by some suggestions for optimizations. This technology can change rapidly, so we advise you to keep an eye out on mining sites such as these in order to keep up with the latest information and releases.

- [Crypto Mining Blog](#)
- [GoByte Forum Mining Discussions](#)
- [Bitcoin Talk Altcoin Mining Discussions](#)

Mining software

As for CPU mining, a range of mining software is available for GPU mining. Most of it is based on sgminer compiled with different optimizations specific to different hardware. A good approach is to identify your graphics hardware, then choose an appropriate build of sgminer. You can use [GPU-Z](#) to identify your GPU hardware:

Next, download the mining software. Most of these are based on the original [sgminer](#), but this is not suitable for the neoscript algorithm, offers no compiled binaries and hasn't been updated in years. We will describe using pre-compiled binary software maintained by newer developers only.

AMD

- <https://github.com/nicehash/sgminer/releases>
- <https://github.com/ghostlander/nsghminer>

NVIDIA

- <https://github.com/tpruvot/ccminer/releases> (focus on core application)
- <https://github.com/sp-hash/ccminer/releases> (sp-mod, optimized CUDA kernels for Windows)
- <https://github.com/KlausT/ccminer/releases> (similar to SP version, more clean)
- <https://github.com/ghostlander/nsghminer>

Download your chosen release and extract the zip file to a known location. The folder should look something like this:

The sgminer file is the executable file, while the various files with .cl extensions define the various algorithms supported by sgminer. In this case, we are interested in the neoscript.cl and neoscript-mod.cl implementations of neoscript. Note that the name of the executable file may be different for miners with different optimizations, for example ccminer for NVIDIA cards.

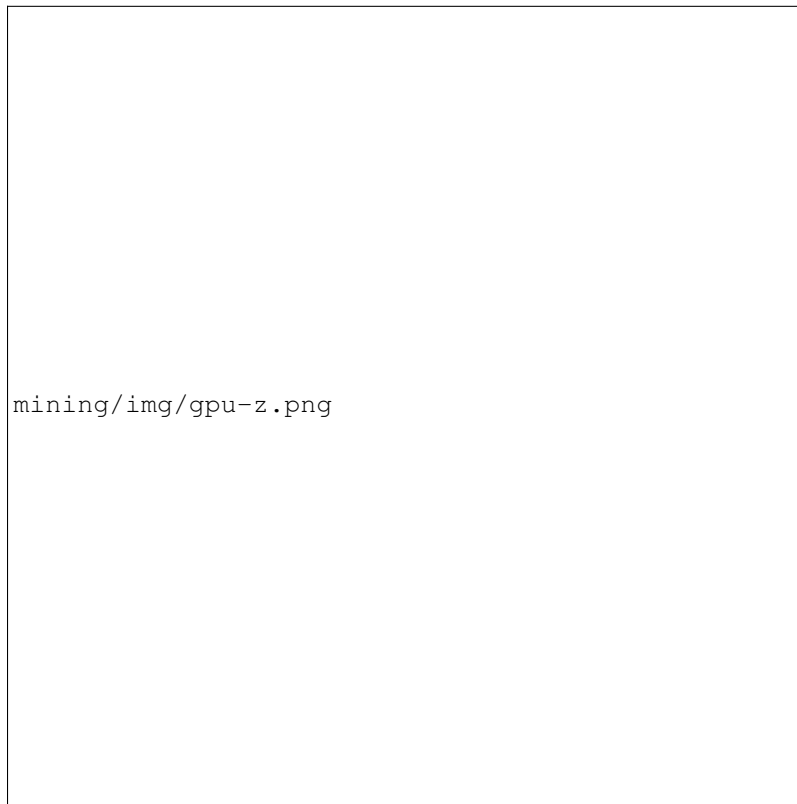


Fig. 229: GPU-Z showing details for AMD Radeon Turks and NVIDIA Quadro GK104 class GPUs



Fig. 230: Executable GPU miners for gobyte

Configuration

Begin by selecting a mining pool and generating a gobyte address as described in the [Mining Pools](#) section above. Keep all your mining files in a single folder. In this example we will work from the Desktop. The node selected for this example is from the BSOD.pw list and is located in US, EU, RU or ASIA:

```
https://bsod.pw/en/pool/dashboard/GBX/
```

Next, open **Notepad** and create the basic configuration. The general format is as follows:

```
{
  "pools" : [
    {
      "url" : "stratum+tcp://eu.bsod.pw:1932",
      "user" : "walletaddress",
      "pass" : "x",
      "algorithm": "neoscrypt"
    }
  ]
}
```

Where:

- pools = defines a list of pools (in this case, only one) towards which the hashing power is directed
- url = URL of your mining pool, including the protocol and port
- user = username, usually the gobyte receiving address of your wallet or worker
- pass = password, can often be set to x
- algorithm = hashing algorithm to use, in this case neoscrypt (for historic reasons) or neoscrypt-mod

For the pool above, the configuration may be:

Click **File**, then **Save As**. Change **Save as type** to **All Files**, then type the file name as *sgminer.conf* and save it in the same folder as the unzipped *sgminer* files.

Testing

Double click your *sgminer.exe* and a **Command Prompt** window should appear immediately. If it disappears too quickly, check your configuration for missing commas, unclosed brackets or incorrect file name. The program will compile a special binary specific to your GPU and store it in the folder, then begin hashing.

1.12.5 FPGA Mining

FPGA stands for *Field-Programmable Gate Array* is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term “field-programmable”. FPGAs are a popular choice for mining cryptocurrency because they can offer a higher efficiency than CPU or GPU miners, resulting in higher profit.

Please note that the information on this page may become obsolete very quickly due to the rapidly changing market and difficulty of mining GoByte. You are responsible for carrying out your own research and any listing on this page should not be considered an endorsement of any particular product. A good place to begin your research is the [mining section of the GoByte Forums](#).

The following NeoScript FPGA miners are available on the market today, click the product name to visit the manufacturer’s website:

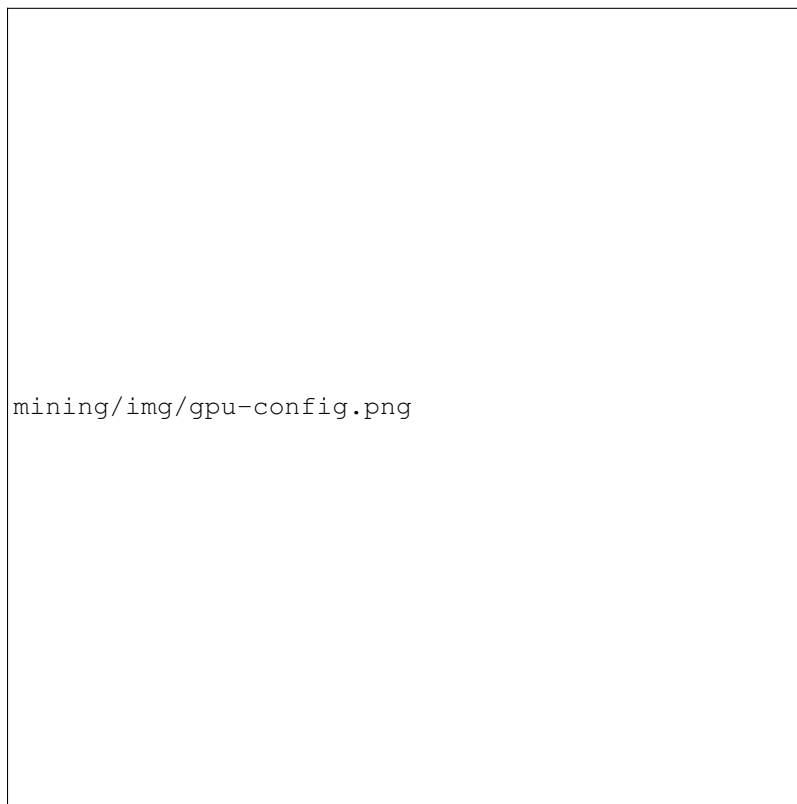


Fig. 231: Configuration file for a gobyte GPU miner

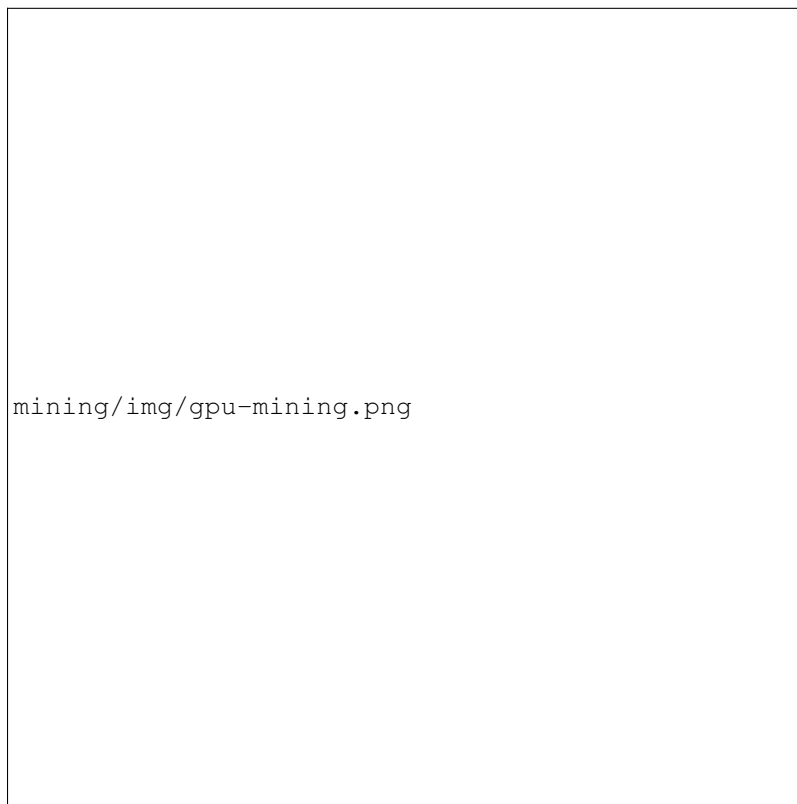


Fig. 232: Example of GPU mining using sgminer on nVidia Graphics card

Name	Hash rate	Power	Weight	Dimensions (mm)	Price
CVP-13 by Bittware	10.5 MH/s	1600 W	?	Standard Size	\$\$5,495

FPGA resellers may also have miners available:

- <https://shop.fpga.guide/>
- <https://www.bittware.com/fpga>

1.13 Developers

GoByte Core has published an extensive [Developer Guide](#) to help new developers get started with the GoByte code base, and as a reference for experienced developers. This guide can be leveraged to quickly and efficiently integrate external applications with the GoByte ecosystem. Anyone can contribute to the guide by submitting an issue or pull request on GitHub. The documentation is available at: <https://gobyte-docs.github.io/en/>

The GoByte Core Team also maintains the [GoByte Roadmap](#), which sets out delivery milestones for future releases of GoByte and includes specific technical details describing how the development team plans to realise each challenge. The GoByte Roadmap is complemented by the [GoByte Improvement Proposals](#), which contain detailed technical explanations of proposed changes to the GoByte protocol itself.

The remaining sections available below describe practical steps to carry out common development tasks in GoByte.

1.13.1 Translating GoByte

Translations of all GoByte products are managed courtesy of Transifex, which offers its own detailed documentation of all functions and features. Within Transifex, GoByte maintains an organization which contains multiple projects and one team of translators assigned to all of the projects. Each project is assigned with one or more target languages for translation by the project maintainer. When a translator joins the team, they are able to choose the languages they feel able to translate. They can then work on any projects specifying this language as a target language. <TO SET UP>

- [Transifex](#)
- [Transifex Documentation](#)
- [GoByte translation projects](#) <TO SET UP>
- [GoByte translators team](#) <TO SET UP>

In general, languages with minimal regional variation are to be translated into the common language (rather than regional) target. Portuguese, for example, is simply translated into the `pt` target language, rather than two separate target languages `pt_BR` and `pt_PT`, for Portuguese as spoken in Brazil and Portugal, respectively. As GoByte grows, these languages may be separated out into their regional variants by proofreaders, depending on demand. Exceptions to this rule apply where the same spoken language is written differently, such as `zh_CN` and `zh_TW` for Simplified Chinese and Traditional Chinese. <TO SET UP>

Keeping translations consistent over time as multiple translators work on each target language is a very important part of delivering a quality user experience. For this reason, if you come across any GoByte-specific terminology such as *masternodes*, you should use the **Concordance** search function to see how the term has been translated in the past. Transifex will also provide **Suggestions** and **History** if it recognizes a similar string in the database of past translations. Stay consistent with past language use, but also ensure your terminology is up to date with current use! <TO SET UP>

[Suggestions](#) 1[History](#) 1[Q Concordance](#)

The following documentation describes the various projects and any special features specific to the programming language in which the product is written.

GoByte Core

<https://www.transifex.com/gobyte/gobyte/> <TO SET UP>

This project contains a file named `gobyte_en.ts`, which is an export of all translatable user-facing content in the *GoByte Core Wallet*. Languages with 80% or more of the translations complete will be integrated in the next release. Note that the software will often replace placeholders in the text with actual numbers, addresses or usernames. If you see a placeholder in the source text, it must also appear in the target text. If it does not, your translation cannot be used. The **Copy source string** button can help you copy everything over, so all you need to do is replace the English words surrounding the placeholders. You can change the order of the placeholders as necessary, according to the grammar of your target language. <TO SET UP>

Placeholders **Source:** `E&xit`

Target: `&Beenden`

Note that the `&` character is placeholder used to indicate a keyboard shortcut in a program menu, and must appear next to the appropriate character in your target language with no adjacent space. Placeholders such as `%1` or `%s` will be replaced by the software as it is running to indicate a name or number of something relating to the message. You must insert these placeholders in the grammatically appropriate position in your target text.

Punctuation **Source:** `change from %1 (%2)`

Target: `Wechselgeld von %1 (%2)`

Note that any brackets `()` and punctuation such as full stops `.` at the end of a sentence must also exist in the target text.

GoByte Docs

https://www.transifex.com/gobyte/gobyte-docs <TO SET UP>

This project contains all content from the GoByte Documentation hosted at <https://docs.gobyte.network> (probably the site you are reading now). Each `.html` page in the documentation appears as a file in the resources section, named according to the navigation steps required to open the page. The GoByte Documentation is written in a documentation language called `reStructuredText` and built using the open-source `Sphinx Documentation Generator`. To simplify layout, most of the text has no markup or code marks at all, but hyperlinks and certain formatting must be reproduced in the target language as follows: <TO SET UP>

Inline literals **Source:** Type `“./gobyte-qt”` to run the file.

Target: Escriba `“./gobyte-qt”` para correr el archivo.

Note that two backticks ```` before and after a word or phrase will cause that text to appear as an `inline literal`. This is commonly used to highlight code or commands to be typed by the user.

Bold and italic **Source:** To encrypt your wallet, click `**Settings**` > `**Encrypt**` wallet.

Target: Para encriptar su billetera, haga click en `**Settings**` > `**Encrypt**` billetera.

A single `*` before and after a word or phrase will render it in an *italic* font, while a double `**` will render it in **bold**.

Internal hyperlinks Source: See `:ref:`here <sporks>`` for a brief introduction to sporks.

Target: Ver `:ref:`aquí <sporks>`` para una breve introducción a sporks

An internal hyperlink consists of the phrase `:ref:`, followed by a single backtick ```, followed by some text which must be translated, followed by angle brackets with the link target `< >`, followed by another backtick ```. Translate the text, but do not translate the text inside the angle brackets.

External hyperlinks Source: The ``official GoByte website <https://www.gobyte.network>`` also provides a list of major exchanges offering GoByte.

Target: El ``sitio web oficial de GoByte <https://www.gobyte.network>`` también proporciona una lista de las principales Casas de cambio o Exchanges que ofrecen GoByte.

An external hyperlink consists of a single backtick ```, followed by some text which must be translated, followed by angle brackets with the link target `< >`, followed by another backtick and a single or double underscore: ``_` or ``__`. Translate the text, but do not translate the hyperlink (unless you want to link to a version of the page in the target language).

GoByte Graphics

<https://www.transifex.com/gobyte/gobyte-graphics> <TO SET UP>

GoByte visual products such as infographics, flyers and conference handouts are produced using Adobe InDesign, Adobe Illustrator or Microsoft Word and are available for use in the *Marketing section* of the GoByte Documentation. It is important to view the finished English layout during translation in order to understand the context of the text you are translating. For example, many words should be translated differently depending if they are a heading, a sentence or an item in a diagram. <TO SET UP>

Because these proprietary file formats are not easily handled by Transifex, the language content is exported to a text or Microsoft Excel file and uploaded to Transifex for processing. If you translate GoByte Graphics, please send an email to antoniom@gobyte.network or @AntonioMoratti on [Discord](#) when you are finished to request layout in the visual design. <TO SET UP>

GoByte iOS Wallet

<https://www.transifex.com/gobyte/gobyte-ios-wallet> <TO SET UP>

All language content from the *GoByte iOS Wallet* are available for translation in this project. Please have a device running the iOS wallet available during translation to understand the context of the text you are translating. Note that any placeholders in the source text segment must also appear in the target language, similar to the instructions above for GoByte Core Wallet. <TO SET UP>

GoByte Android Wallet

<https://www.transifex.com/gobyte/gobyte-wallet> <TO SET UP>

All language content from the *GoByte Android Wallet* are available for translation in this project. Please have a device running the Android wallet available during translation to understand the context of the text you are translating. Note

that any placeholders in the source text segment must also appear in the target language, similar to the instructions above for GoByte Core Wallet. <TO SET UP>

GoByte Videos

<https://www.transifex.com/gobyte/gobyte-videos> <TO SET UP>

This section primarily contains language content from the popular [GoByte Academy](#) video series. Please translate with the videos open in YouTube to properly understand the context of the source text. Once your translation is complete, please send an email to antoniom@gobyte.network or @AntonioMoratti on [Discord](#) to request inclusion of the subtitles on YouTube. <TO SET UP>

GoByte Website

<https://www.transifex.com/gobyte/gobyte-website> <TO SET UP>

The GoByte website at <https://www.gobyte.network> is available for translation in Transifex. Please have the website open while you translate to correctly understand the context of the source text. Once your translation is complete, please send an email to antoniom@gobyte.network or @AntonioMoratti on [Discord](#) to request a build of your translation onto the website. <TO SET UP>

1.13.2 Compiling GoByte Core

While GoByte offers stable binary builds on the [website](#) and on [GitHub](#), and development builds using *Jenkins*, many users will also be interested in building GoByte binaries for themselves. The following guides are available:

- *Building on Linux*
- *Building on macOS*
- *Building on Windows*
- *Gitian deterministic builds*

These guides describe how to build the current stable version. To build the latest version from the develop branch, replace the normal `git clone` command with the following command when pulling from GitHub:

```
git clone https://github.com/gobytecoin/gobyte.git -b develop
```

Linux

This guide describes how to build GoByte Core wallet without the GUI from source under Ubuntu Linux. For a more detailed guide, see the [Unix Build Notes](#). The content on this page is intended to serve as a simple guide for general compilation of non-deterministic binary files from the stable source code. A standard installation of Ubuntu Linux 18.04 LTS will be used as an environment for the build. We assume you are running as a user with sudo permissions. First add the necessary extra repository and update all packages:

```
sudo add-apt-repository ppa:bitcoin/bitcoin
sudo apt update
sudo apt upgrade
```

Now install the dependencies as described in the installation documentation:

```
sudo apt install build-essential libtool autotools-dev automake pkg-config libssl-dev
↳libevent-dev bsdmainutils git libdb4.8-dev libdb4.8++-dev curl
sudo apt install libboost-system-dev libboost-filesystem-dev libboost-chrono-dev
↳libboost-program-options-dev libboost-test-dev libboost-thread-dev libzmq3-dev
```

Optionally install the Qt dependencies if you want to build the GoByte GUI:

```
sudo apt install libqt5gui5 libqt5core5a libqt5dbus5 qttools5-dev qttools5-dev-tools
↳libprotobuf-dev protobuf-compiler
```

Download the stable GoByte repository:

```
git clone https://github.com/gobytecoin/gobyte.git
```

And build:

```
cd gobyte
./autogen.sh
./configure
make
make install
```

/usr/local/bin now contains the compiled GoByte binaries.

macOS

GoByte can be built for macOS either using a cross-compiler under Linux or natively under macOS.

Linux cross-compile

This guide describes how to build GoByte Core wallet from source under Ubuntu Linux. It is intended to serve as a simple guide for general compilation of non-deterministic binary files from the stable source code. For a more detailed guide, see the [macOS Build Notes](#). A standard installation of Ubuntu Linux 18.04 LTS will be used as an environment for the build. We assume you are running as a user with sudo permissions. First add the necessary extra repository and update all packages:

```
sudo add-apt-repository ppa:bitcoin/bitcoin
sudo apt update
sudo apt upgrade
```

Now install the dependencies as described in the installation documentation:

```
sudo apt install build-essential libtool autotools-dev automake pkg-config libssl-dev
↳libevent-dev bsdmainutils git libdb4.8-dev libdb4.8++-dev curl
sudo apt install libboost-system-dev libboost-filesystem-dev libboost-chrono-dev
↳libboost-program-options-dev libboost-test-dev libboost-thread-dev libzmq3-dev
sudo apt install ca-certificates curl g++ git-core pkg-config autoconf librsvg2-bin
↳libtiff-tools libtool automake faketime bsdmainutils cmake imagemagick libcap-dev
↳libz-dev libbz2-dev python python-dev python-setuptools fonts-tuffy p7zip-full
↳sleuthkit
```

Optionally install the Qt dependencies if you want to build the GoByte GUI:

```
sudo apt install libqt5gui5 libqt5core5a libqt5dbus5 qttools5-dev qttools5-dev-tools
↳libprotobuf-dev protobuf-compiler
```

Download the stable GoByte repository:

```
git clone https://github.com/gobytecoin/gobyte.git
cd gobyte
```

A copy of the macOS SDK is required during the build process. To download this, use a Google Chrome in a desktop environment to go to <https://appleid.apple.com> and create or sign in to your Apple account. Then go to <https://developer.apple.com> and open the Chrome Developer Tools from the **Menu -> More tools -> Developer tools**. Click on the **Network** tab, then go back to your main browser window and copy in the following URL:

```
https://developer.apple.com/services-account/download?path=/Developer_Tools/Xcode_7.3.
↳1/Xcode_7.3.1.dmg
```

Cancel the download as soon as it begins and go back to your the **Network** tab in the developer tools. Right click on the network request at the bottom of the list labeled **Xcode_7.3.1.dmg** and select **Copy -> Copy as cURL (bash)**. Paste this long string of text into your Linux terminal, append `-o Xcode_7.3.1.dmg` at the end and then press enter to begin the download. Once it is complete, extract the required files from the disc image as follows:

```
contrib/macdeploy/extract-osx-sdk.sh
rm -rf 5.hfs MacOSX10.11.sdk
mkdir depends/SDKs
mv MacOSX10.11.sdk/ depends/SDKs/
```

And build:

```
make -C depends HOST=x86_64-apple-darwin11
./autogen.sh
./configure --prefix=`pwd`/depends/x86_64-apple-darwin11
make
```

`~/gobyte/src` now contains the compiled GoByte binaries, and `~/gobyte/src/qt` contains the GoByte GUI wallet.

macOS Native

This guide describes how to build GoByte Core wallet from source under macOS. It is intended to serve as a simple guide for general compilation of non-deterministic binary files from the stable source code. For a more detailed guide, see the [macOS Build Notes](#). A standard installation of macOS 10.13 High Sierra will be used as an environment for the build. We assume you are running as a user with sudo permissions. First, open a the **Terminal** app and enter the following command to install the OS X command line tools:

```
xcode-select --install
```

When the popup appears, click **Install**. Then install [Homebrew](#):

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
↳master/install)"
```

Install dependencies:

```
brew install automake berkeley-db4 libtool boost --c++11 miniupnpc openssl pkg-config
↳protobuf qt libevent librsvg
```

Clone the GoByte Core source code and change to the `gobyte` directory:

```
git clone https://github.com/gobytecoin/gobyte
cd gobyte
```

Build GoByte Core. Configure and build the headless gobyte binaries as well as the GUI (if Qt is found). You can disable the GUI build by passing `--without-gui` to configure:

```
./autogen.sh
./configure
make
```

It is recommended to build and run the unit tests:

```
make check
```

You can also create a `.dmg` that contains the `.app` bundle (optional):

```
make deploy
```

GoByte Core is now available at `./src/gobyted`.

Windows

This guide describes how to build GoByte Core wallet from source for 64-bit Windows. Most developers use cross-compilation from Linux to build executables for Windows. The content on this page is intended to serve as a simple guide for general compilation of non-deterministic binary files from the stable source code. For a more detailed guide, see the [Windows Build Notes](#). A standard installation of Ubuntu Linux 18.04 LTS will be used as an environment for the build. We assume you are running as a user with `sudo` permissions. First add the necessary extra repository and update all packages:

```
sudo add-apt-repository ppa:bitcoin/bitcoin
sudo apt update
sudo apt upgrade
```

Now install the dependencies as described in the installation documentation:

```
sudo apt install build-essential libtool autotools-dev automake pkg-config libssl-dev
↳ libevent-dev bsdmainutils git libdb4.8-dev libdb4.8++-dev curl
sudo apt install libboost-system-dev libboost-filesystem-dev libboost-chrono-dev
↳ libboost-program-options-dev libboost-test-dev libboost-thread-dev libzmq3-dev
sudo apt-get install g++-mingw-w64-x86-64 mingw-w64-x86-64-dev
```

Optionally install the Qt dependencies if you want to build the GoByte GUI:

```
sudo apt install libqt5gui5 libqt5core5a libqt5dbus5 qttools5-dev qttools5-dev-tools
↳ libprotobuf-dev protobuf-compiler
```

Download the stable GoByte repository:

```
git clone https://github.com/gobytecoin/gobyte.git
```

Build and link the depends system:

```
cd gobyte/depends
make HOST=x86_64-w64-mingw32
cd ..
```

(continues on next page)

(continued from previous page)

```
sudo update-alternatives --set x86_64-w64-mingw32-gcc /usr/bin/x86_64-w64-mingw32-  
↳gcc-posix  
sudo update-alternatives --set x86_64-w64-mingw32-g++ /usr/bin/x86_64-w64-mingw32-  
↳g++-posix
```

And build:

```
./autogen.sh  
CONFIG_SITE=$PWD/depends/x86_64-w64-mingw32/share/config.site ./configure --prefix=/  
make
```

~/gobyte/src now contains the compiled GoByte binaries, and ~/gobyte/src/qt contains the GoByte GUI wallet.

Gitian

Gitian is the deterministic build process that is used to build the GoByte Core executables. It provides a way to be reasonably sure that the executables are really built from the source on GitHub. It also makes sure that the same, tested dependencies are used and statically built into the executable. Multiple developers build the source code by following a specific descriptor (“recipe”), cryptographically sign the result, and upload the resulting signature. These results are compared and only if they match, the build is accepted and uploaded to gobyte.network.

More independent Gitian builders are needed, which is why this guide exists. It is preferred you follow these steps yourself instead of using someone else’s VM image to avoid ‘contaminating’ the build.

Setup the host environment

Gitian builds are known to be working on Debian 8.x. If your machine is already running this system, you can perform Gitian builds on the actual hardware. Alternatively, you can install it in a virtual machine. Follow the guide for [setting up a VPS for masternodes](#), selecting a Debian 8.x image during the installation process and naming your non-root user gitianuser. Selecting a VPS with two processors will also greatly speed up the build process. If you cannot login to your VPS over SSH as root, access the terminal and issue the following command:

```
sed -i 's/^PermitRootLogin.*/PermitRootLogin yes/' /etc/ssh/sshd_config  
/etc/init.d/ssh restart
```

Log in to your new environment by SSH as root. Set up the dependencies first by pasting the following in the terminal:

```
apt-get install git ruby sudo apt-cacher-ng qemu-utils debootstrap lxc python-cheetah  
↳parted kpartx bridge-utils make ubuntu-archive-keyring curl  
adduser gitianuser sudo
```

Then set up LXC and the rest with the following, which is a complex jumble of settings and workarounds:

```
# the version of lxc-start in Debian needs to run as root, so make sure  
# that the build script can execute it without providing a password  
echo "%sudo ALL=NOPASSWD: /usr/bin/lxc-start" > /etc/sudoers.d/gitian-lxc  
echo "%sudo ALL=NOPASSWD: /usr/bin/lxc-execute" >> /etc/sudoers.d/gitian-lxc  
# make /etc/rc.local script that sets up bridge between guest and host  
echo '#!/bin/sh -e' > /etc/rc.local  
echo 'brctl addbr br0' >> /etc/rc.local  
echo 'ifconfig br0 10.0.3.2/24 up' >> /etc/rc.local  
echo 'iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE' >> /etc/rc.local
```

(continues on next page)

(continued from previous page)

```
echo 'echo 1 > /proc/sys/net/ipv4/ip_forward' >> /etc/rc.local
echo 'exit 0' >> /etc/rc.local
# make sure that USE_LXC is always set when logging in as gitianuser,
# and configure LXC IP addresses
echo 'export USE_LXC=1' >> /home/gitianuser/.profile
echo 'export GITIAN_HOST_IP=10.0.3.2' >> /home/gitianuser/.profile
echo 'export LXC_GUEST_IP=10.0.3.5' >> /home/gitianuser/.profile
reboot
```

At the end Debian is rebooted to make sure that the changes take effect. Re-login as the user gitianuser that was created during installation. The rest of the steps in this guide will be performed as that user.

There is no python-vm-builder package in Debian, so we need to install it from source ourselves:

```
wget http://archive.ubuntu.com/ubuntu/pool/universe/v/vm-builder/vm-builder_0.12.
↳4+bzr494.orig.tar.gz
echo "76cbf8c52c391160b2641e7120dbade5afded713afaa6032f733a261f13e6a8e  vm-builder_0.
↳12.4+bzr494.orig.tar.gz" | sha256sum -c
# (verification -- must return OK)
tar -zxvf vm-builder_0.12.4+bzr494.orig.tar.gz
cd vm-builder-0.12.4+bzr494
sudo python setup.py install
cd ..
```

Set up the environment and compile

Clone the GoByte Core repository to your home directory:

```
git clone https://github.com/gobytecoin/gobyte.git
```

Then create the script file:

```
nano gobyte/contrib/gitian-build.sh
```

And paste the following script in place (this will be automatic if/when the script is pulled into GoByte Core):

```
https://github.com/gobytecoin/gobyte/blob/master/contrib/gitian-build.sh
```

Save the file and set it executable:

```
sudo chmod +x gobyte/contrib/gitian-build.sh
```

Set up the environment, replacing the name and version with your name and target version:

```
gobyte/contrib/gitian-build.sh --setup gobytecoin 0.12.1.5
```

Run the compilation script:

```
gobyte/contrib/gitian-build.sh --build gobytecoin 0.12.1.5
```

Your system will build all dependencies and GoByte Core from scratch for Windows and Linux platforms (macOS if the dependencies were installed according to [these instructions](#)). This can take some time. When complete, you will see the SHA256 checksums, which you can compare against the hashes available on the [GoByte website](#). In this way, you can be sure that you are running original and untampered builds of the code as it exists on GitHub.

1.13.3 Testnet and devnets

With the release of GoByte Core 12.3, GoByte added support for a great new feature—**named devnets**. Devnets are developer networks that combine some aspects of testnet (the global and public testing network) and some aspects of regtest (the local-only regression testing mode that provides controlled block generation). Unlike testnet, multiple independent devnets can be created and coexist without interference. For practical documentation on how to use devnets, see the [developer documentation](#) or this [blog post](#).

Testnet is a fully functioning GoByte blockchain with the one key exception that because the GoByte on the network can be created freely, it has no value. This currency, known as tGBX, can be requested from a faucet to help developers test new versions of GoByte, as well as test network operations using identical versions of the software before they are carried out on the mainnet. There are a few other key differences:

- Testnet operates on port 13455 (instead of 12455)
- Testnet addresses start with “n” instead of “G”, ADDRESSVERSION is 112 (instead of 38)
- Testnet balances are denominated in tGBX (instead of GBX)
- Protocol message header bytes are 0xcee2caff (instead of 0xbf0c6bbd)
- Bootstrapping uses different DNS seeds: testnet-dns.gobyte.network, testnet-dns2.gobyte.network
- Launching GoByte Core in testnet mode shows an orange splash screen

To start GoByte Core in testnet mode, find your gobyte.conf file and enter the following line:

```
testnet = 1
```

Tools and links

The links below were collected from various community sources and may not necessarily be online or functioning at any given time. Please join [GoByte Nation Discord](#) or the [GoByte Forum](#) if you have a question relating to a specific service.

- **Test builds:** #
- **Bugtracker:** <https://github.com/gobytecoin/gobyte/issues/new>
- **Discussion and help:** <https://www.gobyte.network/forum/topic/testing.53/>
- **Masternode tools:** <https://test.masternodes.gobyte.network/masternodes.html>
- **Testnet for Bitcoin:** <https://en.bitcoin.it/wiki/Testnet>

Faucets

- <https://testfaucet.gobyte.network> - by GoByte Core

Explorers

- <http://texplorer.gobyte.network/> - by GoByte Core
- <https://tinsight.gobyte.network> - by GoByte Core

Pools

- <https://testpool.gobyte.network/> [stratum+tcp://testpool.gobyte.network/] - by GoByte Core

Masternodes

Installing a masternode under testnet generally follows the same steps as the *mainnet masternode installation guide*, but with a few key differences:

- You will probably be running a development version of GoByte instead of the stable release. See *here* for a list of builds, then choose the latest successful build and click **Artifacts** to view a list of binaries.
- When opening the firewall, port 13455 must be opened instead of (or in addition to) 12455. Use this command:
`ufw allow 13455/tcp`
- Your desktop wallet must be running in testnet mode. Add the following line to *gobyte.conf*: `testnet = 1`
- When sending the collateral, you can get the 1000 tGBX for free from a faucet (see above)
- You cannot use gobyteman to install development versions of GoByte. See the link to downloadable builds above.
- Your masternode configuration file must also specify testnet mode. Add the following line when setting up *gobyte.conf* on the masternode: `testnet = 1`
- As for mainnet masternodes, the RPC username and password must contain alphanumeric characters only
- When cloning sentinel, you may need to clone the development branch using the `-b` option, for example: `git clone -b core-v0.12.2.x https://github.com/gobytecoin/sentinel.git`
- Once sentinel is installed, modify `~/.gobytecore/sentinel/sentinel.conf`, comment the mainnet line and uncomment: `network=testnet`
- The wallet holding the masternode collateral will expect to find the `masternode.conf` file in `~/.gobytecore/testnet3/masternode.conf` instead of `~/.gobytecore/masternode.conf`.

Testnet 12.3

In September 2020, the GoByte team announced the start of testing of the upcoming GoByte 12.3 release. Extensive internal testing has already been done on the 12.2 code, but there are numerous bugs that can only be revealed with actual use by real people. The GoByte team invites anybody who is interested to download the software and become active on testnet. This release includes:

- Named Devnets, to help developers quickly create multiple independent devnets
- New format of network message signatures
- Governance system improvements
- PrivateSend improvements
- Additional indexes cover P2PK now
- Support for pruned nodes in Lite Mode
- New Masternode Information Dialog

Discussion:

- <https://www.gobyte.network/forum/threads/v12-3-testing.38475>
- Testnet tools: <https://docs.gobyte.network/en/latest/developers/testnet.html>

- Issue tracking: <https://github.com/gobytecoin/gobyte/issues/new>

Latest test binaries:

- <https://github.com/gobytecoin/gobyte/releases/tag/v0.12.3.0-rc3>

Testnet 12.2

In October 2017, the GoByte team announced the launch of a testnet for public testing of the upcoming 12.2 release of the GoByte software. Extensive internal testing has already been done on the 12.2 code, but there are numerous bugs that can only be revealed with actual use by real people. The GoByte team invites anybody who is interested to download the software and become active on testnet. This release includes:

- DIP0001 implementation <https://github.com/dashpay/dips/blob/master/dip-0001.md>
- 10x transaction fee reduction (including InstantSend fee)
- InstantSend vulnerability fix
- Lots of other bug fixes and performance improvements
- Experimental BIP39/BIP44 complaint HD wallet (disabled by default, should be fully functional but there is no GUI yet)

Discussion:

- Testnet 12.2 discussion: <https://www.gobyte.network/forum/threads/v12-2-testing.17412/>
- Testnet tools: <https://www.gobyte.network/forum/threads/testnet-tools-resources.1768/>
- Issue tracking: <https://github.com/gobytecoin/gobyte/issues/new>

Latest successfully built develop branch binaries:

- GoByte Core: #jenkins
- Sentinel: <https://github.com/gobytecoin/sentinel/tree/develop>

1.13.4 Sporks

A multi-phased fork, colloquially known as a “spork”, is a mechanism adopted from DASH used to safely deploy new features to the network through network-level variables to avoid the risk of unintended network forking during upgrades. It can also be used to disable certain features if a security vulnerability is discovered - see [here](#) for a brief introduction to sporks. This documentation describes the meaning of each spork currently existing on the network, and how to check their respective statuses.

Spork functions

Sporks are set using integer values. Many sporks may be set to a particular epoch datetime (number of seconds that have elapsed since January 1, 1970) to specify the time at which they will active. Enabled sporks are set to 0 (seconds until activation). This function is often used to set a spork enable date so far in the future that it is effectively disabled until changed. The following sporks currently exist on the network and serve functions as described below:

SPORK_2_INSTANTSEND_ENABLED Governs the ability of GoByte clients to use InstandSend functionality.

SPORK_3_INSTANTSEND_BLOCK_FILTERING If enabled, masternodes will reject blocks containing transactions in conflict with locked but unconfirmed InstandSend transactions.

SPORK_5_INSTANTSEND_MAX_VALUE Enforces the maximum value in GoByte that can be included in an InstantSend transaction.

SPORK_6_NEW_SIGS Enables a new signature format for GoByte-specific network messages introduced in Dash 12.3. For more information, see [here](#) and [here](#).

SPORK_8_MASTERNODE_PAYMENT_ENFORCEMENT If enabled, miners must pay 65% of the block reward to a masternode currently pending selection or the block will be considered invalid.

SPORK_9_SUPERBLOCKS_ENABLED If enabled, superblocks are verified and issued to pay proposal winners.

SPORK_10_MASTERNODE_PAY_UPDATED_NODES Controls whether masternodes running an older protocol version are considered eligible for payment. This can be used as an incentive to encourage masternodes to update.

SPORK_12_RECONSIDER_BLOCKS Forces reindex of a specified number of blocks to recover from unintentional network forks.

SPORK_13_OLD_SUPERBLOCK_FLAG Deprecated. No network function since block 614820.

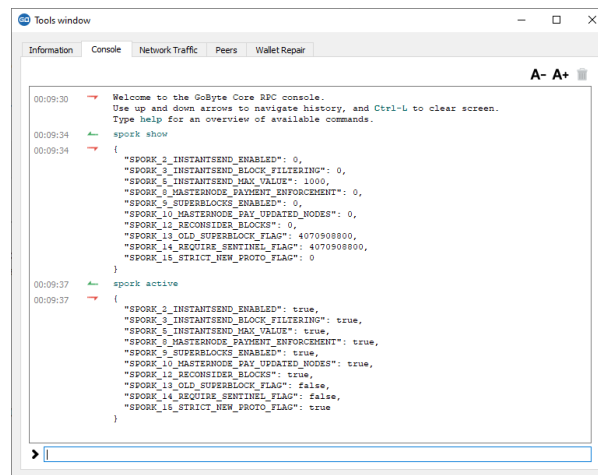
SPORK_14_REQUIRE_SENTINEL_FLAG Toggles whether masternodes with status are eligible for payment if status is WATCHDOG_EXPIRED, i.e. Sentinel is not running properly.

SPORK_15_DETERMINISTIC_MNS_ENABLED Controls whether [deterministic masternodes](#) are required. When activated, the legacy masternode list logic will no longer run and non-updated masternodes will not be eligible for payment.

SPORK_16_INSTANTSEND_AUTOLOCKS Enables automatic transaction locking for transactions with less than a specified number of inputs, and removes the legacy InstantSend fee. Allows any node to request the transaction lock, not just the sending node.

Viewing spork status

The `spork show` and `spork active` commands issued in the debug window (or from `gobyte-cli` on a masternode) allow you to interact with sporks. You can open the debug window by selecting **Tools > Debug console**.



```

00:09:30 Welcome to the GoByte Core RPC console.
          Use up and down arrows to navigate history, and Ctrl-L to clear screen.
          Type help for an overview of available commands.
00:09:34 spork show
00:09:34 {
  "spork_2_instant_send_enabled": 0,
  "spork_3_instant_send_block_filtering": 0,
  "spork_4_instant_send_max_value": 1000,
  "spork_5_masternode_payment_enforcement": 0,
  "spork_6_superblocks_enabled": 0,
  "spork_10_masternode_pay_updated_nodes": 0,
  "spork_12_reconsider_blocks": 0,
  "spork_13_old_superblock_flag": 4070908000,
  "spork_14_require_sentinel_flag": 4070908000,
  "spork_16_strict_new_proto_flag": 0
}
00:09:37 spork active
00:09:37 {
  "spork_2_instant_send_enabled": true,
  "spork_3_instant_send_block_filtering": true,
  "spork_4_instant_send_max_value": true,
  "spork_5_masternode_payment_enforcement": true,
  "spork_6_superblocks_enabled": true,
  "spork_10_masternode_pay_updated_nodes": true,
  "spork_12_reconsider_blocks": true,
  "spork_13_old_superblock_flag": false,
  "spork_14_require_sentinel_flag": false,
  "spork_16_strict_new_proto_flag": true
}

```

Fig. 233: spork show and spork active output in the GoByte Core debug console

1.13.5 Version History

Full release notes and the version history of GoByte are available here:

- <https://github.com/gobytecoin/gobyte/blob/master/doc/release-notes.md>

1.14 Marketing

This page includes downloads of various templates and designs intended for use as office stationary and presentations. For a visual overview of existing web and sticker designs, please see the following links.

- [Rendered designs, badges and stickers](#) <TO SET UP>
- [Vector art package for designers](#) <TO SET UP>

GoByte uses the following color scheme to promote a consistent visual identity.

Color	RGB	CMYK	Hex	Pantone
GoByte Blue	22,102,174	87,41,0,32	#1666AE	2935c
Light Blue	114,144,201	43,28,0,21	#7290C9	535c
Midnight Blue	11,15,59	100,96,41,53	#0b0f3b	5255c
White	255,255,255	0,0,0,0	#ffffff	-
Grey	120,120,120	54,46,45,11	#787878	Cool Gray 9 C
Black	17,25,33	82,71,59,75	#111921	Black 6 C

1.14.1 Design Materials

Brochures

An attractive brochure about GoByte, designed for handing out at conferences and events. <TO SET UP>

This design can be translated into your language at [Transifex here](#). For more information on translating GoByte products on Transifex, see [here](#). Please contact antoniom@gobyte.network once translation is complete to request layout of the completed translation. <TO SET UP>

Language	Download
English	PDF
French	PDF
German	PDF
Spanish	PDF
Thai	PDF

Flyers

An attractive flyer about GoByte, designed to be folded in half and placed on flat surfaces at conferences and events. <TO SET UP>

This design can be translated into your language at [Transifex here](#). For more information on translating GoByte products on Transifex, see [here](#). Please contact antoniom@gobyte.network once translation is complete to request layout of the completed translation. <TO SET UP>

Language	Download
English	PDF
Arabic	PDF
Chinese (Traditional)	PDF
Dutch	PDF
French	PDF
German	PDF
Spanish	PDF
Thai	PDF

Handouts

This handout is ideal for dual-sided printing as a handout for conferences. The current version is **v1.0**; previous versions are available below. <TO SET UP>

This design can be translated into your language at [Transifex here](#). For more information on translating GoByte products on Transifex, see [here](#). Please contact antoniom@gobyte.network once translation is complete to request layout of the completed translation. <TO SET UP>

Language	Download
English	PDF DOCX
Arabic	PDF DOCX
Czech	PDF DOCX
Dutch	PDF DOCX
German	PDF DOCX
Russian	PDF DOCX
Slovak	PDF DOCX
Thai	PDF DOCX
Vietnamese	PDF DOCX

Previous versions (English only):

Version	Download
0.1	PDF DOCX
0.0	PDF DOCX

Infographics

The GoByte Difference

This engaging infographic details the improvements the GoByte network has delivered by building on the Bitcoin code base. Based on an original design by GoByte Core. <TO SET UP>

This design can be translated into your language at [Transifex here](#). For more information on translating GoByte products on Transifex, see [here](#). Please contact antoniom@gobyte.network once translation is complete to request layout of the completed translation. <TO SET UP>

Language	Download
English	PDF PNG
Arabic	PDF PNG
Bulgarian	PDF PNG
Chinese (Simplified)	PDF PNG
Chinese (Traditional)	PDF PNG
Czech	PDF PNG
French	PDF PNG
German	PDF PNG
Greek	PDF PNG
Italian	PDF PNG
Polish	PDF PNG
Russian	PDF PNG
Slovak	PDF PNG
Spanish	PDF PNG
Vietnamese	PDF PNG

Ten Misconceptions About GoByte

This infographic refutes many common yet uninformed arguments made against GoByte. Based on an original design by Antonio Moratti. <TO SET UP>

This design can be translated into your language at [Transifex here](#). For more information on translating GoByte products on Transifex, see [here](#). Please contact antoniom@gobyte.network once translation is complete to request layout of the completed translation. <TO SET UP>

Language	Download
English	PDF PNG
Arabic	PDF PNG
Bulgarian	PDF PNG
Chinese (Traditional)	PDF PNG
Czech	PDF PNG
French	PDF PNG
German	PDF PNG
Greek	PDF PNG
Polish	PDF PNG
Russian	PDF PNG
Slovak	PDF PNG
Spanish	PDF PNG
Thai	PDF PNG
Vietnamese	PDF PNG

Presentations

GoByte Meetup



An attractive presentation about GoByte, designed to guide the audience through the basics of cryptocurrency and advantages of GoByte. <TO SET UP>

Language	Download
English	PPTX PDF
German	PPTX PDF

Simple presentation



A simple presentation about GoByte, available in 5 languages, 3 aspect ratios and 2 formats. Simply click the links to download. Note that the [Noto Sans UI](#) font must be installed if using the PowerPoint files. <TO SET UP>

Browse all files on Dropbox <TO SET UP>

Language	Format	Download
English	PDF	<i>16:9 4:3 A4</i>
	PPTX	<i>16:9 4:3 A4</i>
Chinese (Simplified)	PDF	<i>16:9 4:3 A4</i>
	PPTX	<i>16:9 4:3 A4</i>
Portuguese	PDF	<i>16:9 4:3 A4</i>
	PPTX	<i>16:9 4:3 A4</i>
Russian	PDF	<i>16:9 4:3 A4</i>
	PPTX	<i>16:9 4:3 A4</i>
Spanish	PDF	<i>16:9 4:3 A4</i>
	PPTX	<i>16:9 4:3 A4</i>
Romanian	PDF	<i>16:9 4:3 A4</i>
	PPTX	<i>16:9 4:3 A4</i>

GoByte 101 Presentation

Prepared by GoByte Core following proposal sponsorship for the [GoByte Embassy](#). <TO SET UP>

Language	Download
English	PPTX
French	PPTX
German	PPTX
Spanish	PPTX
Romanian	PPTX

1.14.2 Business Templates

Document templates

Official GoByte document templates. <TO SET UP>

Name	Download
Word document with cover page and paragraph styles	DOCX
Word template with blue watermark	DOTX
Word template with grey watermark	DOTX

Presentation templates

Official GoByte presentation templates. We strongly recommend using predefined presentation slide layouts (check [here](#) for instructions). <TO SET UP>

Name	Download
PowerPoint template with simple blue and white slides	POTX
PowerPoint template with sample layouts, styles and shapes	POTX
Presentation icons	PPTX

Cards

High resolution cards for printing. Great for use as the back of business cards, or to hand out to explain and promote GoByte. <TO SET UP>

Name	Download version
Handout Card	English
	Arabic
	Chinese (Simplified)
	Czech
	French
	Polish
	Portuguese
	Russian
	Spanish

Fonts

Name	Download version
Calibri	6.20
Gunship Bold Italic	5.00
Magistral ATT	1.00
Montserrat	7.20
Noto Sans UI	1.06
FIFA Welcome	1.30

1.15 Legal

1.15.1 How the Law Applies to GoByte

The purpose of the GoByte DAO is to promote, protect and standardize GoByte. In the course of our mission, we have received inquiries into how some aspects of GoByte are treated under United States law. The purpose of this document is to address the most common of these inquiries and explain how we believe the laws apply to GoByte. This is not meant as a legal opinion, and you should consult your own attorneys before relying upon it. However, it is meant to state our position on the law, and how the law should be properly interpreted.

One of the most common questions we receive is *How are masternode operators treated under the US tax laws?*

Tax Treatment

Block rewards

As many already know, block rewards are paid to masternode operators in exchange for validating transactions on the GoByte network. The IRS has stated unequivocally that “when a taxpayer successfully ‘mines’ virtual currency, the fair market value of the virtual currency as of the date of receipt is includible in gross income.” To be sure, masternodes do not “mine”, but the IRS considers using computer resources to validate Bitcoin transactions and maintain the public Bitcoin transaction ledger to constitute “mining”. By analogy, a masternode operator should also treat as regular income the fair market value of the block reward.

GoByte Collateral

A GoByte user may demonstrate to the network his or her control over 1,000 GBX in order to run a masternode. These tokens never leave the user’s control. If at any point during the user’s tenure as a masternode operator, the user disposes of any or all of the 1,000 GBX, the network automatically strips the user of his or her status as a masternode. Under the US Internal Revenue Code, gain or loss is realized only on the “sale or exchange” of property. The term “sale” generally means the transfer of all right, title, and interest in the property transferred. A number of factors typically are considered to determine whether a sale has occurred, the most important being whether the benefits and burdens of ownership of the transferred property have passed from the transferor to the transferee. In GoByte, the masternode operator retains control of the 1,000 GBX and simply demonstrates that control to the network. Therefore, the holding of the 1,000 GBX for purposes of qualifying as a masternode operator should not cause a taxable event to occur because the user has not transferred any of the benefits and burdens of ownership.

Capital Gains

Assuming that the 1,000 GBX are sold, whether that GoByte is a “capital asset” will determine the tax treatment of the sale. Stocks, bonds and other investment property for example, are generally treated as capital assets. Inventory, depreciable property, and stock in trade, though, are not. Assuming the masternode operator held the 1,000 GBX either for investment purposes or for purposes of qualifying as a masternode operator, the IRS would likely treat gain or loss on the sale of those GoByte tokens as capital in nature. Therefore, GoByte held for a long enough period of time could be subject to the lower “long term capital gains” tax rate.

Legal Liability

As with cash or any other currency system, users may use GoByte in connection with illegal activity. A common question we receive is whether masternode operators can also be liable for criminal activity, simply by relaying transactions related to that activity. The fundamental legal requirement of *mens rea* makes criminal liability unlikely for masternode operators.

Primary Liability

Almost all crimes require that a defendant have a defined *mens rea* at the time of an offense. *Mens rea* is a mental state like purposefulness, knowledge, recklessness or negligence. For example, to act with “purpose” is commonly understood as desiring as your “conscious object” the result of a crime. “Knowledge” is a less culpable mindset than “purpose” – acting with “knowledge” requires general awareness that your actions will bring about a particular crime. “Recklessness” requires disregard of a substantial risk. Finally, a person acts “negligently” if they should have been aware of a substantial and unjustifiable risk of a particular consequence of their actions, but were not.

Most masternodes have no awareness, while relaying GoByte transactions, of the identity of the users involved, the ultimate destination of users’ funds, or any other circumstances of GoByte transactions. As such, it would be difficult for a prosecutor to demonstrate that a masternode operator who facilitated an illegal transaction merely by relaying the transaction would have a culpable *mens rea*.

Secondary Liability

Even if someone is not the principal actor in the commission of a crime, that person can be secondarily liable for their involvement in it. As such, we are sometimes asked whether masternode operators, by their involvement in relaying GoByte transactions, could be “aiding and abetting” or “conspiring” to commit a crime that might involve GoByte. Generally speaking, aiding and abetting requires that the defendant (i) seek by his action to make the crime succeed and (ii) act with the same *mens rea* as required for the principal offense.

No matter the requisite *mens rea* of a particular principal offense committed by a GoByte user, it is unlikely that a mere masternode operator, without more, could be found to have “aided and abetted.” To be sure, the masternodes do provide assistance in the principal offense – in that masternode action is required to process all GoByte transactions. However, the masternodes would not have the requisite *mens rea* to satisfy the requirements of aiding and abetting liability. Masternode operators have no readily available information about the purpose or consequences of users’ GoByte transactions, or even the originating identity of the sender of funds. As such, so long as a sufficient diversity of non-criminal transactions occur on the GoByte network, they would not harbor even the least culpable *mens rea* (i.e. negligence) with respect to a user relaying or receiving GoByte in furtherance of a particular crime.

“Conspiracy” liability is even less likely. Conspiracy generally requires i) an agreement to commit a crime, ii) knowledge of the unlawful purpose of the agreement, iii) intent to further the unlawful purpose, and iv) an act in furtherance of the conspiracy. None of these requirements are met by mere masternode operators.

Exchange Liability

Exchanges have asked whether they can be held liable for criminal activity connected with GoByte PrivateSend transactions.

The Bank Secrecy Act (BSA) is the law that primarily governs exchanges in the United States. The BSA does not contain any prohibition on supporting GoByte transactions. Indeed, the BSA take a flexible, risk- based approach to regulation and contemplate that financial institutions will enter into lines of business with new risks. This risk-based approach requires, at the outset, an independent risk assessment. By and large, the risks faced by exchanges who begin to support GoByte will be similar to the risks associated with other virtual currencies. One significant difference concerns PrivateSend transactions, and we focus on this difference below:

- PrivateSend transactions obfuscate the source and destination addresses of funds, thus blockchain forensic techniques like clustering analysis may be less effective. To the extent that exchanges rely on such blockchain forensics tools for their information collection, reporting and reporting obligations under the BSA, they should consider alternative means.
- PrivateSend transactions are used for legitimate purposes and are often required to achieve personal or commercial privacy for sensitive transactions. The use of PrivateSend transactions is not inherently suspicious. Combination with other factors, including those identified in the exchange's own risk assessment, may raise PrivateSend transactions to the level of suspicious activity.
- Exchanges should consider revising their risk assessments and AML policies to account for the unique characteristics of GoByte. For example, including blockchain addresses in Suspicious Activity Reports (SARs) will be less descriptive and effective for investigations based on such addresses. Exchanges might consider adding additional context and explanation in SARs.
- When conducting Enhanced Due Diligence on customers and transactions, exchanges should account for the presence of PrivateSend transactions and update their AML policies accordingly. For example, identifying counterparties to a PrivateSend transaction may be more difficult than identifying counterparties to transactions in other virtual currencies when relying on blockchain forensics.

1.15.2 PrivateSend Legal Position

GoByte's transaction rules are identical to Bitcoin, and therefore for regulatory and compliance purposes GoByte can and should be treated identically to Bitcoin. [BlockchainIntel](#) and [Coinfirm](#) are KYC/AML service providers that offer services covering the GoByte blockchain.

1.15.3 ATM & Fiat Compliance

Introduction

An aspect that required legal research is what are the compliance requirements to facilitate GoByte-fiat exchange. This can be in the form of running ATM kiosks or using other mechanisms to personally offer GoByte to fiat exchange services.

TO UPDATE

GoByte ATM Compliance Program

TO UPDATE