RaiBlocks Integration API for Lykke Exchange

# Introduction

The solution developed during the competition “[RaiBlocks - Blockchain Integration API](https://streams.lykke.com/Project/ProjectDetails/raiblocks-blockchain-integration-api)” in accordance with the [requirements](https://docs.google.com/document/d/1KVd-2tg-Ze5-b3kFYh1GUdGn9jvoo7HFO3wH_knpd3U).

# The structure of the solution

|  |  |
| --- | --- |
| **Component** | **Source code location** |
| Blockchain.API | <https://github.com/artem-kruglov/Raiblocks.Api/tree/dev> |
| Blockchain.SignService | <https://github.com/artem-kruglov/Raiblocks.Sign/tree/dev> |
| Node for private (custom) RaiBlocks test network | <https://github.com/artem-kruglov/raiblocks/tree/testnet> |
| Documentation | 1. The original document 2. <https://github.com/artem-kruglov/Raiblocks.Api/blob/dev/README.md> 3. <https://github.com/artem-kruglov/Raiblocks.Sign/blob/dev/README.md> 4. comments in the source code |

# Most distinctive RaiBlocks features

|  |  |
| --- | --- |
| **RaiBlocks feature** | **Impact on an API implementation** |
| Transferring funds from one account to another requires two transactions: a send deducting the amount from the sender’s balance and a receive adding the amount to the receiving account’s balance.  (A source is the [whitepaper](https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf)) | The [POST] /api/transactions/single method implements only send transaction. There is no method that implements receive transactions.  If the node contains a wallet which contains private keys of some accounts, such node can initialize receive transactions for such accounts by itself. |
| The term “block” and “transaction” are often used interchangeably, where a block contains a single transaction.  (A source is the [whitepaper](https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf)) | Operations with many-input/many-output are not supported |
| RaiBlocks is a feeless cryptocurrency.  (A source is the [whitepaper](https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf)) | The method [PUT] /api/transactions is not implemented.  An unsigned transaction and a signed transaction with the same operationId cannot have different values of an amount. |
| Currently the XRB ticker represents 1 million xrb (Mxrb), which is 10^30 raw, the smallest unit of RaiBlocks. RaiBlocks’ smallest unit is 1 raw, while 1 Gxrb is the largest. 1 xrb is 10^24 raw. XRB is the ticker used on exchanges/software to trade Mxrb. 1 XRB does not equal 1 xrb. 1 XRB currently equals 1Mxrb.  (A source is the [FAQ](https://raiblocks.net/page/faq))  Also, the public [RaiBlocks explorer](https://raiblocks.net/page/representatives) displays the balance in XBR with 6 decimal places. | Accuracy = 6  A minimal unit for transaction using this API is a 1 xbr = 10^24 raw (note that 1 xbr <> 1 XBR, 1 XBR = 1 Mxbr = 10^6 xbr). |
| A block does not contain a timestamp in order to reduce the block size.  (a source is the [RaiBlocks forum](https://forum.raiblocks.net/)) | A transaction history shows timestamp when a transaction had become known to the API service: either the timestamp of getting an information of a transaction from the network or the timestamp of broadcasting that transaction by the API Service (only for transactions with operationId). |
| The PoW in RaiBlocks is simply used as an anti-spam tool, similar to Hashcash, and can be computed on the order of seconds. Once a transaction is sent, the PoW for the subsequent block can be precomputed since the previous block field is known; this will make transactions appear instantaneous to an enduser so long as the time between transactions is greater than  the time required to compute the PoW.  In case of executing offline signature value of a “work” must be already computed.  (A source is the [whitepaper](https://raiblocks.net/media/RaiBlocks_Whitepaper__English.pdf)) | The order of transactions sent to the offline signing method should be the same as the order of transactions sent to the broadcast method. |
| There is no a test public network. | We have manually changed a source code of the RaiBlocks node, compiled that, and made the test network with the custom genesis block (see <https://github.com/artem-kruglov/raiblocks/tree/testnet>) |

# Test network

To create a private test network, a genesis block was generated. A modified source code of a test node was committed to the “testnet” branch:

<https://github.com/artem-kruglov/raiblocks/tree/testnet>

Two files, rai\_node1 and rai\_node2, are compiled nodes, which read the settings from directory RaiBlocks1 and RaiBlocks2 respectively. It makes for easy start of the network consisting of two nodes running on the same computer.

Discussion on Reddit: <https://www.reddit.com/r/RaiBlocks/comments/7m47nh/how_to_run_a_private_raiblocks_testnet/>

# Contacts

Email: [artem.kruglov.ru@gmail.com](mailto:artem.kruglov.ru@gmail.com)

telegram: @artem\_kruglov