Peer API

Version 4.1

Contents

General information **4**
- Basic structures of data 4
  - Transaction 4
  - Block 5
  - Block head 6
- Transaction types 6
  - Basic operations 6
    - 100 - Registration 6
    - 200 - Payment 6
    - 300 - Deposit 7
  - Multisignature management 7
    - 400 - Delegate 7
    - 410 - Quorum 7
    - 420 - Rejection 7
    - 430 - Publication 8
  - Colored coins management 8
    - 500 - ColoredCoinRegistration 8
    - 510 - ColoredCoinPayment 9
    - 520 - ColoredCoinSupply 9
    - 530 - ColoredCoinRemove 9
  - Complex transaction 10
    - 600 - ComplexPayment 10
- Retrieving account identifier 11
- Transaction signature 12
- Retrieving transaction/block identifier 15

Client-to-peer API **16**
- Functions used for interaction with accounts 16
  - State accounts.get_state(String acc) 16
  - Balance accounts.get_balance(String acc) 16
  - Info accounts.get_information(String acc) 17
- Functions used for interaction with transaction history 18
  - Transaction[] history.get_committed_page(String acc, int page) 18
  - Transaction[] history.get_committed(String acc) 19
  - Transaction[] history.get_uncommitted(String acc) 19
  - BlockHeader[] history.get_signed_block(String acc) 19
  - BlockHeader history.get_last_block() 20
  - BlockHeader history.get_block_with_transaction(String id) 20
- Functions for sending transactions 21
  - void transactions.put_transaction(transaction tr) 21

Service functions 21
int time.get() 21
int properties.get_timestamp() 22
String properties.get_network() 22
ColoredCoinInfo colored.get_info(String id) 22

Peer-to-peer API 24
Blocks synchronization functions 24
Difficulty blocks.get_difficulty() 24
Block blocks.get_last_block() 24
Block[] blocks.get_block_history(String[] blockSequence) 25
Peer list synchronization functions 25
Attributes metadata.get_attributes() 25
String[] metadata.get_well_known_nodes() 26
boolean metadata.add_peer(long peerID, String address) 26
Transactions synchronization functions 27
Transaction[] transactions.get_transactions(String lastBlockId, String[] ignoreList) 27
Snapshot synchronization functions 28
Block snapshot.get_last_block() 28
Block snapshot.get_block_by_height(int height) 28
Block[] snapshot.get_blocks_head_from(int height) 28
Account[] snapshot.get_accounts(String blockID) 29
Account[] snapshot.get_next_accounts(String blockID, String accountID) 30

API block explorer 31
Functions used for interaction with accounts 31
Functions used for interaction with transaction and block history 31
Transaction[] explorer.get_committed_page(String accountId, int page) 31
Transaction[] explorer.get_uncommitted(String id) 31
BlockHeader[] explorer.get_last_blocks() 31
BlockHeader[] explorer.get_last_blocks_from(int height) 32
Block explorer.get_block_by_height(int height) 32
Block explorer.get_block_by_id(String blockId) 33
Transaction explorer.get_transaction_by_id(String id) 33
Transaction[] explorer.get_last_uncommitted_trs() 34

Peer monitoring 35
Prometheus API 35

General information

Basic structures of data

Transaction

- `id:string` - transaction identifier. Calculated by the fields `signature` and `timestamp`, converted into a string according to certain rules. Optional field.
- `type:number` - transaction type.
- `timestamp:number` - time of transaction creation (in seconds, unix-time).
- `deadline:number` - lifetime of unsigned transactions in seconds.
- `fee:number` - transaction fee (in microEON). Minimal value is 10microEON per 1kb of transaction length.
- `sender:string` - transaction sender identifier.
- `signature:string` - transaction signature.
- `attachment:object` - additional data (information embedded into the transaction, its format is determined by transaction type). Optional field.
- `confirmations:object` - multisignature confirmation. Format: "account ID":"signature in HEX". Optional field.
- `version:number` - format version. Must be set to 1.
- `note:string` - transaction note. Maximum length is 128 symbols. Allowed latin alphabet, digits and chars “`-#@*_`” (RegExp `"^[-a-zA-Z0-9 #@*_]+$"`). Optional field.
- `reference:string` - reference to the main transaction (only for the complex transaction).
- `bills:object` - internal transactions. Used in the complex transaction (only for the complex transaction).
- `payer:string` - reference to fee payer. Fee payer must confirm a transaction (in confirmations field). Used as a reference to the author of the complex transaction in the bills field. Optional field.

Examples:


```
{"attachment":{"amount":100,"recipient":"EON-WEUCY-TPM29-EK53X"},"deadline":3600,"fee":10,"id":"EON-T-3CMBX-E3N3T-ERSPU","sender":"EON-NRVQC-892FF-475RR","signature":"59c8c7e11d5a922b04e78a3b6da19daf6c75d06d9612bcda4610bd1a93dbf682db85c21ae48eaf4e3a94505e1ce64648ea0e8e6fdeda1411c83a461dce171103","timestamp":1507118401,"type":200,"version":1}
```


Example with a multisignature:


```
{"attachment":{"amount":100,"recipient":"EON-87VQX-TKLTD-3JJNK"},"confirmations":{"EON-CJZU5-85T68-RDM4W":"bf57e4db9fedba5a4aefcea09769f39b2aa92bd62e657b3d484cc50f031787e7fb9e3f1786e23a580c91bafa85cdc2716eba5410f3
```

f97da2d9c8cb52499c4c0d","EON-87VQX-TKLTD-3JJNK":"e0aa2340f1980b00243a87
8c8f5af39e3b106090b8ee7b1b6e5165d2169b4edc726cefafa75451168d601d8d0f907
c11d39f3411279c120f97132a579af89003"},"deadline":3600,"fee":10,"id":"EO
N-T-TDNBX-ETP79-WQ4BP","sender":"EON-NRVQC-892FF-475RR","signature":"69
d3c70d5b06c24572d4632a9780938a6eb61797b988a80f8ecf251176672b920f904f837
f41f9b5dd058a7c1a00d18996d101d839ba37c42c6098f9d45c8c0b","timestamp":15
07119481,"type":200,"version":1}

Block

- `id:string` - block identifier. Calculated by the fields `signature` and `timestamp`, converted into a string according to certain rules. Optional field.
- `timestamp:number` - transaction creation time (in seconds, unix-time).
- `prev:string` - previous block identifier.
- `generator:string` - identifier of an account which generated the block.
- `generation:string` - signature of a block generation.
- `signature:string` - block signature.
- `height:number` - block number in the blockchain (generation number - the number of preceding blocks)
- `difficulty:string` - cumulative difficulty of the blockchain.
- `transactions:array(Transaction)` - the number of transactions within the block. The reward for the block is equal to the sum of the field “fee” of all the transactions within the block. Optional field.
- `snapshot:string` - top of the account state tree after processing the block. It is indicated since v.2
- `version:number` - Format version. Must be set to 1.

Example:

{"difficulty":"2111","generation":"c21c48851c74cb96cc0e34cb6f4cc0506bab
e76a2079f62f7bde2523d188724c6bfdc837bead183f0d814581a38734ee5d91cfa8b5b
a571f55ef1fefe05e0a0c","generator":"EON-WEUCY-TPM29-EK53X","height":1,"
id":"EON-B-NHMBX-EBJKW-BPBKT","prev":"EON-B-2CMBX-EFV9P-XWBSY","signatu
re":"8db0b630cd70121b380d21b2fcd0a5e9f1af0ff5bfd47df026238f4248ba0cdfe8
726f8a1136897f1e531e5e29bae5305befba5fdd388476cf78c188da647002","snapsh
ot":"c0be50283eb40a662fdf679dc246ac9b7f9fb6ca19ee613c459e8df6f6b81fd8d1
8805231714abc3736ab5a08add5e32dbe518e52c6a9ccce3935fc844087a77","timest
amp":1507118580,"transactions":[{"attachment":{"amount":100,"recipient"
:"EON-WEUCY-TPM29-EK53X"},"deadline":3600,"fee":10,"sender":"EON-NRVQC-
892FF-475RR","signature":"fe3b6e24bb39d0b17db105b90bc0657043d517736b35a
caaf982872a97d73bbf4cba94a6a77a65086a15bebe5edca4622133638200b7a56c99d9
6bc24991d406","timestamp":1507118400,"type":200,"version":1},{"attachme
nt":{"amount":100,"recipient":"EON-WEUCY-TPM29-EK53X"},"deadline":3600,
"fee":10,"sender":"EON-NRVQC-892FF-475RR","signature":"59c8c7e11d5a922b

04e78a3b6da19daf6c75d06d9612bcda4610bd1a93dbf682db85c21ae48eaf4e3a94505e1ce64648ea0e8e6fdeda1411c83a461dce171103","timestamp":1507118401,"type":200,"version":1}],"version":1}

Block head

- `id:string` - block identifier. Calculated by the fields `signature` and `timestamp`, converted into a string according to certain rules.
- `timestamp:number` - transaction creation time (in seconds, unix-time).
- `generator:string` - identifier of an account which generated the block.
- `signature:string` - block signature.
- `height:number` - block number in the blockchain (generation number - the number of preceding blocks).
- `transactions_count:number` - transactions count in the block.
- `transactions_fee:number` - cumulative fee of all transactions in the block.

Example:

{"id":"EON-B-2Y9PG-FBPTR-N79PJ","signature":"2b4757408e7fb88afe6bd25f20240fcf2f3f38a091346ca559491f47ef58a787280cb408b0870c0c4483cad69ca8211a2e84e606a7620f0b37a195bb063fdf03","generator":"EON-7TGC9-NJYF8-P7E6X","height":3232,"timestamp":1525325760,"total_fee":0,"transaction_count":0}

Transaction types

Basic operations

100 - Registration

New account registration in the system.

Format of the attachment (`"ID":"PublicKey"`):

- ID - new account identifier.
- PublicKey:string - new account public key (in HEX).

Transaction example:

{"attachment":{"EON-WHY8W-XTMPD-PEW7J":"87cd5726b968289fd17633c3c765149664d204fa78c3d4c7eea30ca8a612dc4e"},...,"type":100}

200 - Payment

Money transfer between two accounts.

Format of the attachment:

- `amount:number` - transfer amount (in microEON, i.e. 1 EON is specified as 1000000).
- `recipient:string` - transfer recipient identifier.

Transaction example:

```
{"attachment":{"amount":100,"recipient":"EON-RSSGB-C9NRJ-935TW"},...,"type":200}
```

300 - Deposit

Sets the deposit funds to participate in the generation of blocks. If you want to delete the deposit, you should set its size to zero.

Format of the attachment:

- `amount:number` - deposit size (in microEON, i.e. 1 EON is specified as 1000000).

Transaction example:

```
{"attachment":{"amount":100},...,"type":300}
```

Multisignature management

400 - Delegate

Delegation of a right to sign.
A specified "percentage" of the signature is attributed to an account identifier. If you specify it as “0”, the right to sign is withdrawn.

Format of the attachment ("ID":vote):

- ID - account identifier.
- vote:number - percentage share of account signature.

Transaction example:

```
{"attachment":{"EON-WHY8W-XTMPD-PEW7J":50},...,"type":400}
```

410 - Quorum

Transactions quorum setting.

Format of the attachment:

- `all:number` - general quorum (for all transaction types, which do not have a specifically defined quorum value).
- `"transaction type code":quorum value(number)` - specifically defined quorum value for some types of transactions.

Transaction example:

```
{"attachment":{"all":50,"100":20,"200":70},...,"type":410}
```

420 - Rejection

Rejection of a right to sign.

Account, which is the only one having the right of signature, cannot give up the right of signature.

Format of the attachment:

- `account:string` - account identifier, which we refuse to grant with the right of signature.

Transaction example:


```
{"attachment":{"account:"EON-RSSGB-C9NRJ-935TW"},...,"type":420}
```


430 - Publication

Account seed publication.
To make an account public you should:

1. Withdraw rights of signature of all other accounts.
2. Specify signature “percentage” as 0% for the account which is made public (it means withdrawal of the right to sign).
3. Wait 24 hours after applying last change in delegation of a right to sign.
4. Publish account seed.

Format of the attachment:

- `seed:string` - secret seed of the account which is made public.

Transaction example:


```
{"attachment":{"seed":"87cd5726b968219fd17633c3c765142564d204fa78c3d4c7
eea30ca8a612dc4e"},...,"type":430}
```


Colored coins management

The following modes for colored coins are supported:

- Pre-emission. In this mode the total number of colored coins is fixed. All issued colored coins enter on a creator's balance.
- Auto-emission. In this mode colored coins are created when a creator transfers the coins to another account and are burned when the colored coins return to him. The account’s balance of the own colored coins (created by yourself) is always a zero. The total number of colored coins (emission) is the total number of colored coins on all other accounts.

The maximum number of colored coins can be equal 2^63-1 = 9223372036854775807.

500 - ColoredCoinRegistration

Colored coins creation.
You can create coins, which identifier is defined by a user account identifier (with “EON-C” prefix). The user becomes the creator of the colored coin and can manage the whole monetary mass of the coin. One account can create only one colored coin type.

Format of the attachment:

- `emission:number|string` - created colored coins supply.
  - `number` - total emission of colored coins (pre-emission mode).
  - `"auto"(string)` - auto-emission mode.
- `decimal:number` - decimal places (0 to 10).

Pre-emission transaction example:

```
{"attachment":{"emission":1000000,"decimal":3},...,"type":500}
```

Auto-emission transaction example:

```
{"attachment":{"emission":"auto","decimal":3},...,"type":500}
```

510 - ColoredCoinPayment

Colored coins transfer.

Format of the attachment:

- `amount:number` - transfer amount in atomic values (e.g. in microEON).
- `recipient:string` - transfer recipient identifier.
- `color:string` - colored coins type identifier.

Transaction example:

```
{"attachment":{"amount":100,"recipient":"EON-RSSGB-C9NRJ-935TW","color"
:"EON-C-RMNF4-KLGQ7-9Y65X"},...,"type":510}
```

520 - ColoredCoinSupply

Change in the colored coin emission.
New colored coins are sent to the creator's account if the emission grows, and withdrawn from the account when emission decreases.

Format of the attachment:

- `supply:number|string` - new emission.
  - `number` - total emission of colored coins (sets pre-emission mode).
  - `"auto"(string)` - sets auto-emission mode.

Transaction examples:

```
{"attachment":{"supply":200000},...,"type":520}
```

```
{"attachment":{"supply":"auto"},...,"type":520}
```

530 - ColoredCoinRemove

Removes colored coins. All issued colored coins must be on the creator's balance.

Format of the attachment: not used.

Transaction example:

```
{...,"type":530}
```

Complex transaction

600 - ComplexPayment

Complex transaction is a transaction which includes several standard payment transactions. Allowed types of the internal transactions:

- 200 - Payment.
- 510 - ColoredCoinPayment.

The internal transactions are represented in the `bills` field as Key-Value: "transaction ID":Transaction(object).

The internal transaction can be one of two types:

- The main transactions - transactions without `reference` field.
  - `payer:string` - must be set the value as the complex payment's author.
- The referenced transactions - transactions with `reference` field.
  - `reference:string` - contains the ID of one of the referenced transactions (referenced transaction will be executed before the current transaction).
  - `payer:string` - must be empty (not used).

A fee of the internal transactions must be set to 0 (not applicable outside the complex transaction).

Format of the attachment: not used.

Transaction example:


```
{"bills":{"EON-T-WUMBX-EX8PC-QCKQW":{"attachment":{"amount":99,"recipie
nt":"EON-CJHAZ-69PW3-R6VGQ"},"deadline":3600,"fee":0,"id":"EON-T-WUMBX-
EX8PC-QCKQW","reference":"EON-T-WUMBX-E7GSR-CHURN","sender":"EON-NRVQC-
892FF-475RR","signature":"58b2909bc568f72537566878a17e01a6c8f60753460e4
eb3cdc6046216a016b15e3c83064f2e9fc1afee9fc7b06a68ff4fa4a38c846bff966c0f
c36c5212900e","timestamp":1507118940,"type":200,"version":1},"EON-T-WUM
BX-E7GSR-CHURN":{"attachment":{"amount":100,"recipient":"EON-CJHAZ-69PW
3-R6VGQ"},"deadline":3600,"fee":0,"id":"EON-T-WUMBX-E7GSR-CHURN","payer
":"EON-CJHAZ-69PW3-R6VGQ","sender":"EON-NRVQC-892FF-475RR","signature":
"6402e94adc4aabdb094524c77c9426808daddcefcb8669bf07e800c4608d6f45d588e8
ede84933e48a6e60c00aabd6462190955b747f5d8b25cefa758b7b060e","timestamp"
:1507118940,"type":200,"version":1}},"deadline":3600,"fee":30,"id":"EON
-T-XUMBX-EF8Z6-B9C3X","sender":"EON-CJHAZ-69PW3-R6VGQ","signature":"cfc
f8fd16b86edcd5557583c8ebf29a7966105ad45ca9b180ce06ba3b564b4039d8402f91b
1ac9d0b0ae6cab4f6590d557e61b6a230fc384497bbfe73e00cd06","timestamp":150
7118941,"type":600,"version":1}
```

Retrieving account identifier

The following describes an algorithm for obtaining an account identifier:

1. Preparation of the key pair.
    a. The key pair is generated using NaCl library: NaCl.sign.keyPair.fromSeed (seed).
    b. seed - 32 bytes are generally transferred as HEX string.
    c. Can use any means for generating a seed-string, for example:
        i. openssl rand -hex 32
2. Preparation of numerical account identifier (long - 64 bits).
    a. The public key is taken from the key pair (private / public).
    b. Public key is encoded with SHA-512 to get an array of bytes.
    c. The resulting array is converted into a 8 byte number:
        i. The byte array is divided into blocks of 8 bytes, each block generates an 8-byte number, which are then used as operands of XOR operations.
            1. 0th byte - low byte number, 7th byte - high.
3. Converting numeric account ID to a string representation.
    a. The resulting 64 bits are extended to 75 as follows:
        i. the 64 bits are divided into 7 blocks of 10 bits (the missing bits are zeros).
        ii. The obtained blocks (numbers) are then used as operands of XOR operations, and result in one 10-bit block (10-bit check number).
        iii. The bits of the 10-bit number are the bits 65-74 in the required number.
        iv. 75th bit is set to 1.
    b. The resulting 75-bit number is converted into a string according to the BASE32 algorithm (15 characters).
        i. The following set of characters is used for converting: `23456789ABCDEFGHJKLMNPQRSTUVWXYZ` (no similar characters).
    c. The resulting string is broken into 3 groups of 5 elements via a hyphen.
    d. A prefix is added (`"EON-"` for accounts).
    e. The account identifier is represented according to the following form: `EON-XXXXX-XXXXX-XXXXX`.

Example (Java):

Step 1:
https://github.com/EonTechnology/server/blob/3e38dddaa410291f2268a5de71d12eb0809250d9/peer-core/src/main/java/com/exscudo/peer/core/crypto/ed25519/Ed25519Signer.java#L16

Step 2:
https://github.com/EonTechnology/server/blob/3e38dddaa410291f2268a5de71d12eb0809250d9/peer-core/src/main/java/com/exscudo/peer/core/data/identifier/BaseIdentifier.java#L91

Step 3:
https://github.com/EonTechnology/server/blob/3e38dddaa410291f2268a5de71d12eb0809250d9/peer-core/src/main/java/com/exscudo/peer/core/data/identifier/BaseIdentifier.java#L122

Transaction signature

Transaction signature algorithm is described below:

1. To sign a transaction it is necessary to form an object of the following structure:
    a. `type:number` - transaction type.
    b. `timestamp:number` - when the transaction was created (in seconds, unix-time).
    c. `deadline:number` - the lifetime of unsigned transactions in seconds.
    d. `fee:number` - transaction fee (in microEON, i.e 1 EON specified as 1000000).
    e. `sender:string` - sender ID.
    f. `attachment:object` - additional data (or empty object).
    g. `network:string` - network identifier.
    h. `version:number` - format version. Set to 1.
    i. `note:string` - transaction note (if the value is set).
    j. `bills:object` - internal transactions (if the value is set). In structure used for signature calculation.
    k. `payer:string` - fee payer (if the value is set).
    l. `reference:string` - reference to main transaction (if the value is set).
2. This object is converted into BENCODE-string.
3. The resulting string is converted to uppercase.
4. An array of bytes (UTF-8 encoding) is taken from the upper case string.
5. The resulting array of bytes is used to calculate the signature by the private key. Getting the byte array with the signature:
    a. Public and private keys are received on the seed:
        i. `key = nacl.sign.keyPair.fromSeed(seed)`
    b. The data is signed `(data` - data from par.4):
        i. `nacl.sign.detached(data, key.secretKey)`
6. The received signature is converted into HEX-string. This string is recorded in the `signature` transaction field (or added in the `confirmations` for multisignature in the form of "account identifier":"signature").

Example:
For example, we want to make a transfer of 10,000 microEONs (0,1 EON) from the account "EON-NRVQC-892FF-475RR" to the account "EON-87VQX-TKLTD-3JJNK".

Transaction (JSON-formatted):


```
{"attachment":{"amount":1000000,"recipient":"EON-87VQX-TKLTD-3JJNK"},"deadline":3600,"fee":10,"sender":"EON-NRVQC-892FF-475RR","signature":null,"timestamp":1507118581,"type":200,"version":1}
```

In order to confirm a transaction, it is necessary to sign it with a digital signature.
In order to form the signature it is necessary to use:

1. The account seed.
2. Network identifier.
3. The transaction itself.

The account seed of the "EON-NRVQC-892FF-475RR" is shown below (HEX):
`eba54bbb2dd6e55c466fac09707425145ca8560fe40de3fa3565883f4d48779e`

Network identifier can be requested from the peer (or to know in advance - it is a constant value within the network). This parameter is used to prevent a transaction from going to other network, for example, from a TestNet to MainNet or vice versa. For receiving the network identifier you must call "metadata.get_attributes()" (see API/peer) and get the value of the field network_id:


```
>>>{"jsonrpc":"2.0","id":30378,"method":"metadata.get_attributes"}
<<<{"jsonrpc":"2.0","id":30378,"result":{"announced_address":"127.0.0.1
:8080","application":"EON","version":"0.9.0","fork":1,"peer_id":-723206
9433514112627,"network_id":"EON-B-2CMBX-EFV9P-XWBSY"}}
```


An example of a transaction was given above.

Transaction signature:

Step 1 - form the object (represented in JSON format for convenience):


```
{"attachment":{"amount":1000000,"recipient":"EON-87VQX-TKLTD-3JJNK"},"d
eadline":3600,"fee":10,"network":"EON-B-2CMBX-EFV9P-XWBSY","sender":"EO
N-NRVQC-892FF-475RR","timestamp":1507118581,"type":200,"version":1}
```


Step 2 - convert the object into BENCODE-string:


```
d10:attachmentd6:amounti1000000e9:recipient21:EON-87VQX-TKLTD-3JJN
Ke8:deadlinei3600e3:feei10e7:network23:EON-B-2CMBX-EFV9P-XWBSY6:se
nder21:EON-NRVQC-892FF-475RR9:timestampi1507118581e4:typei200e7:ve
rsioni1ee
```


Step 3 - convert it to uppercase:


```
D10:ATTACHMENTD6:AMOUNTI1000000E9:RECIPIENT21:EON-87VQX-TKLTD-3JJN
KE8:DEADLINEI3600E3:FEEI10E7:NETWORK23:EON-B-2CMBX-EFV9P-XWBSY6:SE
NDER21:EON-NRVQC-892FF-475RR9:TIMESTAMPI1507118581E4:TYPEI200E7:VE
RSIONI1EE
```


Step 4 - we get a byte array in UTF-8 (represented in HEX-format for convenience):


```
4431303a4154544143484d454e5444363a414d4f554e544931303030303030304539
3a5245343495049454e5432313a454f4e2d38375651582d544b4c54442d334a4a4e
```

```
4b45383a444541444c494e4549333630304533a4645454931304537 3a4e455457
4f524b32333a454f4e2d422d32434d42582d45465639502d5857425359363a5345
4e44455232313a454f4e2d4e525651432d38393246462d3437355252393a54494d
455354414d5049313530373131383538314534 3a5459504549323030453 73a5645
5253494f4e49314545
```

Step 5 - signing the data:
The first step is to get the key pair from seed (using a high-level API NaCL library):

```
key = nacl.sign.keyPair.fromSeed(seed)
```

public key:

```
c8069cf37abb6723feafe5cd93ba4961aaf74e4ac4837fc3c5ffa7a0acea5672
```

private key (containing a public key):

```
eba54bbb2dd6e55c466fac09707425145ca8560fe40de3fa3565883f4d48779ec8
069cf37abb6723feafe5cd93ba4961aaf74e4ac4837fc3c5ffa7a0acea5672
```

signing the data:

```
signature = nacl.sign.detached(data, key.secretKey)
```

Then we get the following signature (in HEX):

```
833b2723fe43958d87b502b94f66bdbaf4a0e221d29a6c43d0a72423318cabaaf5
e1755def04cb34053ee69b6d154365c143df9c3bf361f28ff35a9a278a3009
```

Step 6 - we receive the signed transaction:
To do this, the resulting array of bytes in HEX-format stores in the `signature` field the following:


```
{"attachment":{"amount":1000000,"recipient":"EON-87VQX-TKLTD-3JJNK"},"d
eadline":3600,"fee":10,"id":"EON-T-PHMBX-EK2XT-CZSCQ","sender":"EON-NRV
QC-892FF-475RR","signature":"833b2723fe43958d87b502b94f66bdbaf4a0e221d2
9a6c43d0a72423318cabaaf5e1755def04cb34053ee69b6d154365c143df9c3bf361f28
ff35a9a278a3009","timestamp":1507118581,"type":200,"version":1}
```


The transaction signature is complete, it can now be sent to the peer for further processing.

Retrieving transaction/block identifier

An algorithm for obtaining a transaction (or a block) identifier is the following:

1. The field `signature` (a byte array) is coded by SHA-512.
2. The resulting hash sum (the byte array) is converted to a 4-byte number:
    a. The array is divided into blocks of 4 bytes, each block forms a 4-byte number, which are then used as operands of XOR operations.
        i. 0th byte is the lowest byte, 3th byte is the senior.
3. An 8-byte identifier is generated.
    a. 4 senior bytes - it is the number which was obtained by converting digital signature (in par.2).
    b. 4 lowest bytes is the transaction creation time (its timestamp).
4. The resulting identifier is converted according to the algorithm, which has been described previously (see "Retrieving account identifier").
    a. "EON-B" is used as a prefix for blocks.
    b. "EON-T" is used as a prefix for transactions.

Implementation case (Java):
https://github.com/EonTechnology/server/blob/3e38dddaa410291f2268a5de71d12eb0809250d9/peer-core/src/main/java/com/exscudo/peer/core/data/identifier/BaseIdentifier.java#L73

Client-to-peer API

JSON-RPC 2.0 protocol is used for the interaction.
Transported via HTTP (POST-requests).
Entry point (TestNet): https://peer.testnet.eontechnology.org:9443/bot/v1

Functions used for interaction with accounts

State accounts.get_state(String acc)

Get account status.UZG5V-JKN8H-7NDVK"],"id":3}

- Input parameters:
  - `acc:string` - the account identifier.
- Return value: `object(State)`
  - `code:number` - status code (via HTTP).
  - `name:string` - text description.
  - Possible states
    - `{"code":200,"name":"OK"}` the account is confirmed.
    - `{"code":102,"name":"Processing"}` the account is being processed.
    - `{"code":404,"name":"Not Found"}` the account is not found.
- In case of an incorrect account identifier an error is returned, for example:
  - `{"jsonrpc":"2.0","error":{"code":-32099,"message":"Unknown format or data corrupted. EON-UZG52-JKN8H-7NDVK"},"id":null}`

Examples:

- **>>>** - request.
- **<<<** - response.

>>>`{"jsonrpc":"2.0","method":"accounts.get_state","params":["EON-UZG5V-JKN8H-7NDVK"],"id":3}`
<<<`{"jsonrpc":"2.0","id":3,"result":{"code":200,"name":"OK"}}`

Balance accounts.get_balance(String acc)

Get account balance.

- Input parameters:
  - `acc:string` - account identifier.
- Return value: `object(Balance)`:
  - `State:object(State)` - account state
    - `code:number` - status code (HTTP).

        - `name:string` - text description.
        - Possible meanings:
            - `{"code":200,"name":"OK"}` the account is verified.
            - `{"code":401,"name":"Unauthorized"}` the account is not verified.
    - `amount:number` - balance in microEON.
    - `colored_coins:object` - balance of different types of colored coins. Dictionary type "colored coin identifier": the balance in minimum units. Optional field.

Examples

>>>`{"jsonrpc":"2.0","method":"accounts.get_balance","params":["EON-QB3PM-Y3MRM-28TFL"],"id":3}`

<<<`{"jsonrpc":"2.0","id":3,"result":{"state":{"code":200,"name":"OK"},"amount":819174151,"colored_coins":{"EON-C-QB3PM-Y3MRM-28TFL":1000000}}}`

Info accounts.get_information(String acc)

Get account information.

- Input parameters:
    - `acc:string` - account identifier.
- Return value: `object(Info)`
    - `state:object(State)` - account state:
        - `code:number` - status code (HTTP).
        - `name:string` - text description.
        - Possible values:
            - `{"code":200,"name":"OK"}` the account is verified.
            - `{"code":401,"name":"Unauthorized"}` the account is not verified.
    - `amount:number` - balance in microEON.
    - `deposit:number` - deposit in microEON (i.e., the amount involved in the generation of coins).
    - `public_key:string` - the public key in HEX-format.
    - `sign_type:string` - the signature type. Possible values:
        - `"normal"` - Simple signature. The account individually signs its own transactions (signature percentage 100%).
        - `"mfa"` - Multisignature. A multisignature may be required for the transaction adoption (depending on the quorum set). Signature percentage may vary from 0% to 99%.
        - `"public"` - Public Account. A multisignature is always required for the transaction adoption. Account seed is published online. The account has no right to sign its own transactions (signature percentage 0%).
    - `voting_rights:object` - the delegation of authority to sign. Format:

        - `weight:number` - signature percentage of the account's own transactions.
        - `delegates:object` - signature percentage delegated to other accounts. Dictionary type "Account identifier":registered signatures percentage.
    - `quorum:object` - the necessary quorum to adopt processing of the transaction. The default value is 100%. Format:
        - `quorum:number` - percentage by default.
        - `quorum_by_types:object` - percentages on certain types of transactions (if they differ from the default value). Dictionary "transaction type code":quorum value.
    - `seed:string` - account seed in HEX-format.
    - `voter:object` - participation in the delegation of authority to sign. Dictionary "Account identifier":signature percentage.
    - `colored_coin:string` - colored coins type identifier which were created by the account. Optional field.
- In case of an error, such as an incorrect account identifier, an error is returned:
    - `{"jsonrpc":"2.0","error":{"code":-32099,"message":"Unknown format or data corrupted. EON-UZG52-JKN8H-7NDVK"},"id":null}`

Examples:

>>>{"jsonrpc":"2.0","method":"accounts.get_information","params":["EON-UZG5V-JKN8H-7NDVK"],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":{"state":{"code":200,"name":"OK"},"public_key":"c3c03262a50ea80b1b8e5be4eb1bc34a22c1ccbda1d666644b5bcc0ef5f88ca5","amount":799999970,"deposit":100000000,"sign_type":"normal"}}

Detailed example:

>>>{"jsonrpc":"2.0","method":"accounts.get_information","params":["EON-UZG5V-JKN8H-7NDVK"],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":{"state":{"code":200,"name":"OK"},"public_key":"b59ea1a634b7a6367755b544ce5df468be9b1a6a9d4f1fce9409468adc0046c9","amount":99999970,"deposit":0,"sign_type":"mfa","voting_rights":{"weight":70,"delegates":{"EON-73MQU-39KHN-9NHQM":100}},"quorum":{"quorum":50,"quorum_by_types":{"310":100,"320":100}},"voter":{"EON-QB3PM-Y3MRM-28TFL":70}}}

Functions used for interaction with transaction history

Transaction[] history.get_committed_page(String acc, int page)

Get a confirmed transactions page associated with the account.

- Input parameters:
  - `acc:string` - account ID.
  - `page:number` - page number, counting from zero (transactions sorted by timestamp, desc).
- Return value: `array(Transaction)`.

Examples

>>>{"jsonrpc":"2.0","method":"history.get_committed_page","params":["EON-NRVQC-892FF-475RR",0],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":[{"attachment":{"amount":100,"recipient":"EON-WEUCY-TPM29-EK53X"},"deadline":3600,"fee":10,"id":"EON-T-3CMBX-E3N3T-ERSPU","sender":"EON-NRVQC-892FF-475RR","signature":"59c8c7e11d5a922b04e78a3b6da19daf6c75d06d9612bcda4610bd1a93dbf682db85c21ae48eaf4e3a94505e1ce64648ea0e8e6fdeda1411c83a461dce171103","timestamp":1507118401,"type":200,"version":1},...]}

Transaction[] history.get_committed(String acc)

Get the latest confirmed transactions associated with the account. Analogous to history.get_committed_page(acc, 0).

Transaction[] history.get_uncommitted(String acc)

Get unconfirmed transactions associated with the account.
The format of input/output is completely similar to history.get_committed(acc).

BlockHeader[] history.get_signed_block(String acc)

Get last block heads signed by the account.

- Input parameters:
  - `acc:string` - account ID.
- Return value: `array(BlockHeader)`.
  - `id:string` - block identifier. Calculated by the fields `signature` and `timestamp`, converted into a string according to certain rules.
  - `timestamp:number` - transaction creation time (in seconds, unix-time).
  - `generator:string` - identifier of an account which generated the block.
  - `signature:string` - block signature.
  - `height:number` - block height in the blockchain.
  - `transactions_count:number` - transactions count in the block.
  - `transactions_fee:number` - cumulative fee of all transactions in the block.

Examples

>>>{"jsonrpc":"2.0","method":"history.get_signed_block","params":["EON-7TGC9-NJYF8-P7E6X"],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":[{"id":"EON-B-WS3QG-FBKW9-78NMT","signature":"38a8fd71aa7428e3e996604f0a90c6e2562d1e3828babaaaf0c80ab33c39b2a5caff4e92891ddd91fe73d6143b03a41195283c781f5d1927d76c96487d9afb0e","generator":"EON-7TGC9-NJYF8-P7E6X","height":3379,"timestamp":1525352220,"transactions_fee":120,"transactions_count":12},...]}

BlockHeader history.get_last_block()

Get header of the last block in the blockchain.

- Input parameters: none.
- Return value: `object(BlockHeader)`.
  - `id:string` - block identifier. Calculated by the fields `signature` and `timestamp`, converted into a string according to certain rules.
  - `timestamp:number` - transaction creation time (in seconds, unix-time).
  - `generator:string` - identifier of an account which generated the block.
  - `signature:string` - block signature.
  - `height:number` - block height in the blockchain.
  - `transactions_count:number` - transactions count in the block.
  - `transactions_fee:number` - cumulative fee of all transactions in the block.

Examples

>>>{"jsonrpc":"2.0","method":"history.get_last_block","id":3}

<<< {"jsonrpc":"2.0","id":3,"result":{"id":"EON-B-6QKYM-FP3J8-BZGLY","signature":"7754643c6908e47a6512022bcd5ce1592fe4d22d1491f60f0c5f51fc5422f8f7f9aab9ef15b4a5f40b137bf9a4146a805b3a98d443fcf0748500a49996f95609","generator":"EON-GY9GJ-XRY7R-RQCDT","height":29253,"timestamp":1530873540,"transactions_fee":0,"transactions_count":0}}

BlockHeader history.get_block_with_transaction(String id)

Get header of the block by transaction identifier (block with this transaction).

- Input parameters:
  - `id:string` - transaction ID.
- Return value: `object(BlockHeader)`.
  - `id:string` - block identifier. Calculated by the fields `signature` and `timestamp`, converted into a string according to certain rules.
  - `timestamp:number` - transaction creation time (in seconds, unix-time).
  - `generator:string` - identifier of an account which generated the block.
  - `signature:string` - block signature.
  - `height:number` - block height in the blockchain.

- `transactions_count:number` - transactions count in the block.
- `transactions_fee:number` - cumulative fee of all transactions in the block.

Examples

```
>>>{"jsonrpc":"2.0","method":"history.get_block_with_transaction","id":3,"params":["EON-T-P8DFH-FKJ6H-U2GSW"]}
<<<{"jsonrpc":"2.0","id":3,"result":{"id":"EON-B-JADFH-F3E5V-GTMFM","signature":"fb10c0926c15790870807cf4360f7db39cad8d1fb086a5973603a0c063534724bebd94985a0e145d0b56bd9b2324ea13104357597c005a2aa77168ada2419c07","generator":"EON-V9S5M-34Z7X-UX97X","height":2820,"timestamp":1526115600,"transactions_fee":10,"transactions_count":1}}
```

Functions for sending transactions

void transactions.put_transaction(transaction tr)

Send a transaction to a peer.

- Input parameters:
  - `tr:object(Transaction)` - transaction.
- Return value: none
  - in case of success: "result":"success".
  - In case of transaction processing error an JsonRPC error returns.

Examples

```
>>>{"jsonrpc":"2.0","id":3,"method":"transactions.put_transaction","params":[{"attachment":{"amount":100,"recipient":"EON-WEUCY-TPM29-EK53X"},"deadline":3600,"fee":10,"id":"EON-T-3CMBX-E3N3T-ERSPU","sender":"EON-NRVQC-892FF-475RR","signature":"59c8c7e11d5a922b04e78a3b6da19daf6c75d06d9612bcda4610bd1a93dbf682db85c21ae48eaf4e3a94505e1ce64648ea0e8e6fdeda1411c83a461dce171103","timestamp":1507118401,"type":200,"version":1}]}
<<<{"jsonrpc":"2.0","id":3,"result":"success"}
```

Validation error

```
<<<{"jsonrpc":"2.0","error":{"code":-32602,"message":"Invalid timestamp or other params for set the time."},"id":null}
```

Service functions

int time.get()

Get server time (unix-timestamp).

- Input parameters: none.
- Return value: `number`.

Examples:

```
>>>{"jsonrpc":"2.0","id":427,"method":"time.get"}
<<<{"jsonrpc":"2.0","id":427,"result":1505914067}
```

int properties.get_timestamp()

Get server time (unix-timestamp). It's analog of int time.get().

- Input parameters: none.
- Return value: `number`.

Examples:

```
>>>{"jsonrpc":"2.0","id":427,"method":"properties.get_timestamp"}
<<<{"jsonrpc":"2.0","id":427,"result":1505914067}
```

String properties.get_network()

Get network identifier.

- Input parameters: none.
- Return value: `string`.

Examples:

```
>>>{"jsonrpc":"2.0","id":3,"method":"properties.get_network"}
<<<{"jsonrpc":"2.0","id":3,"result":"EON-B-2LVXG-F7DT2-SSWYS"}
```

ColoredCoinInfo colored.get_info(String id)

Get colored coin information.

- Input parameters:
  - `id:string` - identifier of a colored coin type.
- Return value: `object(ColoredCoinInfo)`
  - `state:object(State)` - colored coin state
    - `code:number` - status code (HTTP).
    - `name:string` - text description.
    - Possible values:
      - `{"code":200,"name":"OK"}` colored coin exists.
      - `{"code":401,"name":"Unauthorized"}` there is no information about this colored coin.
  - `supply:number` - the total number of colored coins of this type. Used in minimum units (e.g. in microEON), the maximum emission can be equal to: 2 ^ 63-1 = 9223372036854775807.

- `decimal:number` - number of decimals in this type of a colored coin (0 to 10).
- `timestamp:number` - time of creation of a colored coin.
- `auto:boolean` - status of auto-emission mode (true if enabled).

Examples:

>>>{"jsonrpc":"2.0","method":"colored.get_info","params":["EON-C-QB3PM-Y3MRM-28TFL"],"id":3}

<<< {"jsonrpc":"2.0","result":{"state":{"code":200,"name":"OK"},"supply":100000000000000,"auto":false,"decimal":8,"timestamp":1529420580},"id":3}

Peer-to-peer API

JSON-RPC 2.0 protocol is used for the interaction
Transport - HTTP (POST-requests).
The entry point (TestNet): https://peer.testnet.eontechnology.org:9443/peer/v1

Blocks synchronization functions

These functions are responsible for synchronization of blocks between peers.

Difficulty blocks.get_difficulty()

Get the blockchain difficulty on the peer.

- Input parameters: none.
- Return value: `object(Difficulty)`
  - `difficulty:string` - blockchain difficulty.
  - `last:string` - the last block identifier in the blockchain.

Examples:

>>>{"jsonrpc":"2.0","id":41782,"method":"blocks.get_difficulty"}

<<< {"jsonrpc":"2.0","id":41782,"result":{"difficulty":"56140","last":"EON-B-SDNBX-EKKCR-FRWPZ"}}

Block blocks.get_last_block()

Get the last block in the blockchain.

- Input parameters: none.
- Return value: `object(Block)`.

Examples:

>>>{"jsonrpc":"2.0","id":41782,"method":"blocks.get_last_block"}

<<< {"jsonrpc":"2.0","id":41782,"result":{"difficulty":"56140","generation":"6dc4880f41174d6fad32a525227d91d3c9d809c173584001405ef2820e5ad6e39ac1918699da21d381718afe7be8d5a6a4dbfbfd61b4f776eb54c876bedb9e0e","generator":"EON-NRVQC-892FF-475RR","height":6,"id":"EON-B-SDNBX-EKKCR-FRWPZ","prev":"EON-B-68NBX-E73SM-ML9LK","signature":"0f76bc8cffb441e62d3d3999213af9e23167203bd14304eb54d4317e1c590afe10dc57eabcb322c93c4d5821c09105af36fdeace44de512dff10b4b955a0fa0a","snapshot":"57d7dc24c0c6c3327f919afbd8fe08569ef6da5912e091f8abc13815685b396fd2f7a4d867970a64ac72d21e4079be50c6896a968ca55d48a5e034dc9b46219a","timestamp":1507119480,"version":1}}

Block[] blocks.get_block_history(String[] blockSequence)

Get continuation of the blockchain.
This function searches the common part of the blockchain and returns the part of continuation of the blockchain starting from the last common block.

- Input parameters:
  - `blockSequence:array(string)` - block IDs array in the chain.
- Return value: `array(Block)`.

Examples:

>>>{"jsonrpc":"2.0","id":41782,"method":"blocks.get_block_history","params":[["EON-B-22222-22222-2222J","EON-B-2UUHB-F79EY-TWFRY"]]}

<<<{"jsonrpc":"2.0","id":41782,"result":[{"difficulty":"2742","generation":"d5075ed6ecc6e35b802fb1765732be2b3e5e1abe22d8593c71d5cfb023233c6bc608ae37f9f1fa6112001a1e089d9bdb2a4450e09efd84be4bf415209d71de01","generator":"EON-D3GHJ-3YKU2-5XKTK","height":1,"id":"EON-B-NZUHB-FPWDD-Z9ZNQ","prev":"EON-B-2UUHB-F79EY-TWFRY","signature":"d8a6303c492be75e02fb79fb89d4e37cbc59c5760fbaab15b05bf3993811fc7f10c3487bb9b003d527b700cf5173a375b56f6a1e0b5b9539db5e118d7e2ba000","snapshot":"fdca2fb3391a5e5e035e1416ecfd0e0707c97ec06363759e1eeaa9a247006f886b6d49732b3fb4585d78e06e53edfc6297b9cdd4f562da476e8fd2729c7a3eb1","timestamp":1519905780,"version":1},{"difficulty":"4266","generation":"bbca0073a4dfe37c5c2abe0a302a47e76056f5e710a1a3b9ee940da048f8ea6ac0e8f2f1f3a10795fceeb9c3b5888b6da75e6187de64d0327174097f1e952309","generator":"EON-D3GHJ-3YKU2-5XKTK","height":2,"id":"EON-B-A7VHB-F3JJY-DSKJS","prev":"EON-B-NZUHB-FPWDD-Z9ZNQ","signature":"0f8035930e7a97bd588145c46a3c7c90a092943ece7d22532bc584799fbda9ada37bd57342d6b337f17710def6294ebfebb19abca04470123ceeddf2016ef607","snapshot":"fdca2fb3391a5e5e035e1416ecfd0e0707c97ec06363759e1eeaa9a247006f886b6d49732b3fb4585d78e06e53edfc6297b9cdd4f562da476e8fd2729c7a3eb1","timestamp":1519905960,"version":1},...]}

Peer list synchronization functions

These functions are responsible for synchronization of peer list and establish a connection between peers.

Attributes metadata.get_attributes()

Get server attributes.

- Input parameters: none.
- Return value: `object(Attributes)`.

- `announced_address:string` - public address of the peer.
- `application:string` - application name.
- `version:string` - application version.
- `fork:number` - fork number.
- `peer_id:number` - peer ID.
- `network_id:string` - network ID.
- `history_from_height:number` - peer contains full blockchain history from this block height.

Examples:

>>>{"jsonrpc":"2.0","id":30378,"method":"metadata.get_attributes"}

<<< {"jsonrpc":"2.0","id":30378,"result":{"announced_address":"127.0.0.1:9443","application":"EON","version":"0.10.0","network_id":"EON-B-2UUHB-F79EY-TWFRY","fork":2,"history_from_height":0,"peer_id":9105485843595180797}}

String[] metadata.get_well_known_nodes()

Get a list of peer addresses which the peer is connected to. The list contains only those connections which are currently active.

- Input parameters: none.
- Return value: `array(string).`

Examples:

>>>{"jsonrpc":"2.0","id":41782,"method":"metadata.get_well_known_nodes"}

<<< {"jsonrpc":"2.0","id":41782,"result":["193.124.176.13:9443","81.82.229.114:9443","185.82.218.107:9443","45.32.87.241:9443","195.123.212.70:9443","45.76.124.206:9443","217.12.221.5:9443","176.78.128.15:9443","45.76.131.32:9443","104.156.246.67:9443","193.70.41.106:9443"]}

boolean metadata.add_peer(long peerID, String address)

Add a peer in the peer list on the server.

- Input parameters:
  - `peerID:number` - peer ID which is being added to the list. It is randomly generated at the time of the launch of the peer.
  - `address:string` - public address of the peer that is added to the list.
- Return value: `boolean`
  - `true` - if the peer has successfully been added to the list of peers on the server.
  - `false` - otherwise.

If a local address was sent, the address of the sender peer will be used.

Server verifies peer connectivity. If all tests are passed successfully, the peer will be added to the list of peers on the server.

Examples:

```
>>>{"jsonrpc":"2.0","id":41782,"method":"metadata.add_peer","params":[123,"127.0.0.1:9443"]}
<<<{"jsonrpc":"2.0","id":41782,"result":true}
```

If there was impossible to connect to the peer with the mentioned address, an error is returned:

```
>>>{"jsonrpc":"2.0","id":41782,"method":"metadata.add_peer","params":[123,"127.0.0.1:9443"]}
<<<{"jsonrpc":"2.0","error":{"code":-32099,"message":"java.net.SocketTimeoutException: connect timed out"},"id":null}
```

Transactions synchronization functions

These functions synchronize awaiting confirmation transactions between peers.

Transaction[] transactions.get_transactions(String lastBlockId, String[] ignoreList)

Get a list of transactions awaiting confirmation.

- Input parameters:
  - `lastBlockId:string` - the last block identifier at the client peer.
  - `ignoreList:array(string)` - transactions IDs array on the client peer.
- Return value: `array(Transaction)`.

Only those transactions are returned that should be included into the next block and are not in ignoreList.
If lastBlockId does not coincide with the last block on the current server peer, then an empty list of transactions is returned (in case of different blockchains on the client peer and when the server peer transaction synchronization is not performed).

Examples:

Step 1 - if there is no transactions awaiting confirmation on the client peer:

```
>>>{"jsonrpc":"2.0","id":41782,"method":"transactions.get_transactions","params":["EON-B-J2NBX-ET962-59LLL",[]]}
<<<{"jsonrpc":"2.0","id":41782,"result":[{"attachment":{"amount":100,"recipient":"EON-WEUCY-TPM29-EK53X"},"deadline":3600,"fee":10,"id":"EON-T-3CMBX-E3N3T-ERSPU","sender":"EON-NRVQC-892FF-475RR","signature":"59c8c
```

7e11d5a922b04e78a3b6da19daf6c75d06d9612bcda4610bd1a93dbf682db85c21ae48e
af4e3a94505e1ce64648ea0e8e6fdeda1411c83a461dce171103","timestamp":15071
18401,"type":200,"version":1}]}

Step 2 - repeated request

>>>{"jsonrpc":"2.0","id":41782,"method":"transactions.get_transactions"
,"params":["EON-B-SYWQ5-725BB-VQ7SS",["EON-T-3CMBX-E3N3T-ERSPU"]]}

<<< {"jsonrpc":"2.0","id":41782,"result":[]}

Snapshot synchronization functions

These functions are responsible for initial synchronization of peer with partial blockchain.

Block snapshot.get_last_block()

Analogous to the call block synchronization: blocks.get_last_block().

Block snapshot.get_block_by_height(int height)

Get the block from the blockchain by block height.

- Input parameters:
  - `height:number` - number of the block in the blockchain (generation number).
- Return value: `object(Block)`.

Examples:

>>>{"jsonrpc":"2.0","method":"snapshot.get_block_by_height","params":[5
],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":{"difficulty":"27785","generation":
"d3f45404bc0cab724996544b5ceab56dd3d7f87739a17b6954b5996a64713a4916e0a6
2331a953dd6d6064b5e5c3c53cd8fe08ca9e8dad38121026ff5e770e05","generator"
:"EON-G9RKK-2JTDZ-GEYBY","height":5,"id":"EON-B-68NBX-6EAL5-8DJUP","pre
v":"EON-B-J2NBX-624LC-RU4ZZ","signature":"034c903796b6f7ed62e4d182025d6
6d89cb445e32dd8a9a9f9717f83e0004e5076f8ee53d346a18ae19f191dcc23b6dd7b48
815b1dda0a6ba1ec6f66f70e0701","timestamp":1507119300,"version":1}}

Block[] snapshot.get_blocks_head_from(int height)

Get the blocks heads (block without transactions) from the blockchain started by block height.

- Input parameters:
  - `height:number` - number of the block in the blockchain (generation number).

- Return value: `array(Block)`.

Examples:

>>>{"jsonrpc":"2.0","method":"snapshot.get_blocks_head_from","params":[1],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":[{"difficulty":"2742","generation":"d5075ed6ecc6e35b802fb1765732be2b3e5e1abe22d8593c71d5cfb023233c6bc608ae37f9f1fa6112001a1e089d9bdb2a4450e09efd84be4bf415209d71de01","generator":"EON-D3GHJ-3YKU2-5XKTK","height":1,"id":"EON-B-NZUHB-FPWDD-Z9ZNQ","prev":"EON-B-2UUHB-F79EY-TWFRY","signature":"d8a6303c492be75e02fb79fb89d4e37cbc59c5760fbaab15b05bf3993811fc7f10c3487bb9b003d527b700cf5173a375b56f6a1e0b5b9539db5e118d7e2ba000","snapshot":"fdca2fb3391a5e5e035e1416ecfd0e0707c97ec06363759e1eeaa9a247006f886b6d49732b3fb4585d78e06e53edfc6297b9cdd4f562da476e8fd2729c7a3eb1","timestamp":1519905780,"version":1},{"difficulty":"4266","generation":"bbca0073a4dfe37c5c2abe0a302a47e76056f5e710a1a3b9ee940da048f8ea6ac0e8f2f1f3a10795fceeb9c3b5888b6da75e6187de64d0327174097f1e952309","generator":"EON-D3GHJ-3YKU2-5XKTK","height":2,"id":"EON-B-A7VHB-F3JJY-DSKJS","prev":"EON-B-NZUHB-FPWDD-Z9ZNQ","signature":"0f8035930e7a97bd588145c46a3c7c90a092943ece7d22532bc584799fbda9ada37bd57342d6b337f17710def6294ebfebb19abca04470123ceeddf2016ef607","snapshot":"fdca2fb3391a5e5e035e1416ecfd0e0707c97ec06363759e1eeaa9a247006f886b6d49732b3fb4585d78e06e53edfc6297b9cdd4f562da476e8fd2729c7a3eb1","timestamp":1519905960,"version":1},...]}

Account[] snapshot.get_accounts(String blockID)

Get account states from a snapshot of a block. For initial iteration over all accounts in a snapshot.

- Input parameters:
  - `blockID:string` - ID of the block in the blockchain.
- Return value: `array(Account)`.

Account ID can be interpreted as 8-byte number. Accounts in a snapshot are sorted according to reversed bits in Account ID (unsigned number).

Examples:

>>>{"jsonrpc":"2.0","id":41782,"method":"snapshot.get_accounts","params":["EON-B-2UH5G-FXQ2U-KAF2S"]}

<<< {"jsonrpc":"2.0","id":41782,"result":[{"balance":{"amount":1500000000000},"public-key":{"pk":"a99cbd462712f7a3e90fce2fe2b4c35048e4e1eb70022ae78b1e71a923c858a4"},"id":"EON-2JUNV-24Q55-UKMEJ"},...]}

Account[] snapshot.get_next_accounts(String blockID, String accountID)

Get next account states from a snapshot of a block. For iteration of next accounts in a snapshot (see snapshot.get_accounts(...)).

- Input parameters:
  - `blockID:string` - ID of the block in the blockchain.
  - `accountID:string` - ID of the first account in a set.
- Return value: `array(Account)`.

Examples:

>>>{"jsonrpc":"2.0","id":41782,"method":"snapshot.get_next_accounts","params":["EON-B-2UH5G-FXQ2U-KAF2S","EON-6UAM2-X6NEG-N7EYP"]}

<<< {"jsonrpc":"2.0","id":41782,"result":[{"deposit":{"amount":500000000,"timestamp":0},"public-key":{"pk":"6291a859ff5296299431ca9eaf81da977f3733fe9ae8caab037c7efeb9e2f99a"},"id":"EON-6UAM2-X6NEG-N7EYP"},,...]}

API block explorer

JSON-RPC 2.0 protocol is used.
Transport - HTTP (POST-requests).
The entry point (TestNet): https://peer.testnet.eontechnology.org:9443/explorer/v1

By default requests are allowed only from the IP 127.0.0.1.
Access policy configuration is performed in the file
peer-eon-app/src/main/webapp/WEB-INF/spring/spring-security.xml
section
`<intercept-url pattern="/explorer/**" access="hasIpAddress('127.0.0.1')"/>`

In order to access with another IP, you must add a condition (for example, for local access at peer launch in docker):
`<intercept-url pattern="/explorer/**" access="hasIpAddress('127.0.0.1') or hasIpAddress('172.17.0.1')"/>`

Functions used for interaction with accounts

These functions are completely analogous to those described in the "Client-to-peer API" section . Only the entry point is changed: /explorer instead of /bot.

Functions used for interaction with transaction and block history

Transaction[] explorer.get_committed_page(String accountId, int page)

Analogous to the call transaction history: history.get_committed_page().

Transaction[] explorer.get_uncommitted(String id)

Analogous to the call transaction history: history.get_uncommitted().

BlockHeader[] explorer.get_last_blocks()

Get the last blocks heads from the blockchain.

- Input parameters: none.
- Return value: `array(BlockHeader)`.
  - `id:string` - block identifier. Calculated by the fields `signature` and `timestamp`, converted into a string according to certain rules.
  - `timestamp:number` - transaction creation time (in seconds, unix-time).
  - `generator:string` - identifier of an account which generated the block.
  - `signature:string` - block signature.
  - `height:number` - block height in the blockchain.
  - `transactions_count:number` - transactions count in the block.

- `transactions_fee:number` - cumulative fee of all transactions in the block.

Examples:

>>>{"jsonrpc":"2.0","method":"explorer.get_last_blocks","params":[],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":[{"id":"EON-B-WS3QG-FBKW9-78NMT","signature":"38a8fd71aa7428e3e996604f0a90c6e2562d1e3828babaaaf0c80ab33c39b2a5caff4e92891ddd91fe73d6143b03a41195283c781f5d1927d76c96487d9afb0e","generator":"EON-7TGC9-NJYF8-P7E6X","height":3379,"timestamp":1525352220,"transactions_fee":120,"transactions_count":12},...]

BlockHeader[] explorer.get_last_blocks_from(int height)

Get the last blocks heads from the blockchain up to defined block number.

- Input parameters:
  - `height:number` - the maximum number of the block in the blockchain (generation number).
- Return value: `array(BlockHeader)`.

Examples:

>>>{"jsonrpc":"2.0","method":"explorer.get_last_blocks_from","params":[3379],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":[{"id":"EON-B-WS3QG-FBKW9-78NMT","signature":"38a8fd71aa7428e3e996604f0a90c6e2562d1e3828babaaaf0c80ab33c39b2a5caff4e92891ddd91fe73d6143b03a41195283c781f5d1927d76c96487d9afb0e","generator":"EON-7TGC9-NJYF8-P7E6X","height":3379,"timestamp":1525352220,"transactions_fee":120,"transactions_count":12},...]}

Block explorer.get_block_by_height(int height)

Get the block from the blockchain by block height.

- Input parameters:
  - `height:number` - number of the block in the blockchain (generation number).
- Return value: `object(Block)`.

Examples:

>>>{"jsonrpc":"2.0","method":"explorer.get_block_by_height","params":[5],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":{"difficulty":"27785","generation":"d3f45404bc0cab724996544b5ceab56dd3d7f87739a17b6954b5996a64713a4916e0a6

2331a953dd6d6064b5e5c3c53cd8fe08ca9e8dad38121026ff5e770e05","generator"
:"EON-G9RKK-2JTDZ-GEYBY","height":5,"id":"EON-B-68NBX-6EAL5-8DJUP","pre
v":"EON-B-J2NBX-624LC-RU4ZZ","signature":"034c903796b6f7ed62e4d182025d6
6d89cb445e32dd8a9a9f9717f83e0004e5076f8ee53d346a18ae19f191dcc23b6dd7b48
815b1dda0a6ba1ec6f66f70e0701","timestamp":1507119300,"version":1}}

Block explorer.get_block_by_id(String blockId)

Get the block from the blockchain by block identifier.

- Input parameters:
  - `blockId:string` - the block ID.
- Return value: `object(Block)`.

Examples:

>>>{"jsonrpc":"2.0","method":"explorer.get_block_by_id","params":["EON-
B-68NBX-6EAL5-8DJUP"],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":{"difficulty":"27785","generation":
"d3f45404bc0cab724996544b5ceab56dd3d7f87739a17b6954b5996a64713a4916e0a6
2331a953dd6d6064b5e5c3c53cd8fe08ca9e8dad38121026ff5e770e05","generator"
:"EON-G9RKK-2JTDZ-GEYBY","height":5,"id":"EON-B-68NBX-6EAL5-8DJUP","pre
v":"EON-B-J2NBX-624LC-RU4ZZ","signature":"034c903796b6f7ed62e4d182025d6
6d89cb445e32dd8a9a9f9717f83e0004e5076f8ee53d346a18ae19f191dcc23b6dd7b48
815b1dda0a6ba1ec6f66f70e0701","timestamp":1507119300,"version":1}}

Transaction explorer.get_transaction_by_id(String id)

Get the transaction by the transaction identifier.

- Input parameters:
  - `id:string` - transaction ID.
- Return value: `object(Transaction)`.

Examples:

>>>{"jsonrpc":"2.0","method":"explorer.get_transaction_by_id","params":
["EON-T-XPX5E-FBVTR-ERGHJ"],"id":3}

<<<< {"jsonrpc":"2.0","id":3,"result":{"attachment":{"amount":1,"recipie
nt":"EON-MM5U6-3VTJR-LES4P"},"deadline":3600,"fee":10,"id":"EON-T-XPX5E
-FBVTR-ERGHJ","sender":"EON-FE8TS-C4ZVR-UMPCL","signature":"1c99982721f
14d4285fdb3b23fda8650eabe25a6ee7cb62c7825e9819961343b90848ae5355a52973e
531eaecc4421fff6daac18f88eedf047fc26bc06c2840f","timestamp":1522661053,
"type":200,"version":1}}

Transaction[] explorer.get_last_uncommitted_trs()

Get the latest uncommitted transactions.

- Input parameters: none.
- Return value: `array(Transaction).`

Examples:

>>>{"jsonrpc":"2.0","method":"explorer.get_last_uncommitted_trs","params":[],"id":3}

<<< {"jsonrpc":"2.0","id":3,"result":[{"attachment":{"amount":1,"recipient":"EON-MM5U6-3VTJR-LES4P"},"deadline":3600,"fee":10,"id":"EON-T-XPX5E-FBVTR-ERGHJ","sender":"EON-FE8TS-C4ZVR-UMPCL","signature":"1c99982721f14d4285fdb3b23fda8650eabe25a6ee7cb62c7825e9819961343b90848ae5355a52973e531eaecc4421fff6daac18f88eedf047fc26bc06c2840f","timestamp":1522661053,"type":200,"version":1},...]}

Peer monitoring

Prometheus API

Peer supports Prometheus monitoring system (https://prometheus.io/).
URL: https://peer.testnet.eontechnology.org:9443/metrics

Monitoring parameters:

- `eon_memory_used` - volume of memory used (in Java VM, in bytes).
- `eon_memory_total` - total amount of allocated memory (in Java VM, in bytes).
- `eon_memory_max` - the maximum possible amount of allocated memory (in Java VM, in bytes).
- `eon_last_block_id` - numeric identifier of the last block in the blockchain.
- `eon_last_block_height` - block height of the blockchain.
- `eon_last_block_generator` - numeric identifier of the author of the last block in the blockchain.
- `eon_last_block_transactions_count` - number of transactions in the last block in the blockchain.
- `eon_cumulative_difficulty` - overall difficulty of the blockchain.
- `eon_transactions_count` - the number of transactions on the peer awaiting confirmation.
- `eon_peer_count` - number of peers in the network.
- `eon_uptime` - uptime since the launch of the peer (in seconds).
- `eon_target_height` - expected height of the blockchain.
- `eon_node_cache_added` - total items added to node cache (cache size - 100000).
- `eon_node_cache_removed` - total items removed from node cache.
- `eon_history_from_height` - peer contains full blockchain history from this block height.

Example (command line):

```
user@pc:~$ curl --insecure -L
https://peer.testnet.eontechnology.org:9443/metrics

eon_memory_used 53784296
eon_memory_total 103284736
eon_memory_max 354811904
eon_last_block_id -4237599240209011444
eon_last_block_height 26399
eon_last_block_generator 4147593152785103498
eon_last_block_transactions_count 3
eon_cumulative_difficulty 8818470048
eon_transactions_count 1
eon_peer_count 16
eon_uptime 1293
```

```
eon_history_from_height 0
eon_node_cache_added 105499
eon_node_cache_removed 5499
eon_target_height 26399
```