EIYARO Core API Reference

Table of Contents

Wallet Endpoints

These endpoints are available when we set:
config.toml

```toml
[wallet]
disable = false
```

This is the default value and we can possibly omit it.

Create Key Endpoint

Creates a private key. The private key is encrypted in the file and not visible to the user.

Parameters

`Object`:

- `String` - **alias**, name of the key.
- `String` - **password**, password of the key.
- `String` - **language**, mnemonic language of the key.

Optional:

- `String` - **mnemonic**, mnemonic of the key, create key by specified mnemonic.

Returns

`Object`:

- `String` - **alias**, name of the key.
- `String` - **xpub**, root pubkey of the key.
- `String` - **file**, path to the file of key.

Optional:

- `String` - **mnemonic**, mnemonic of the key, exist when the request mnemonic is null.

Example

Create key by random pattern:

Request

```
curl -X POST http://localhost:9888/create-key -d '{"alias": "alice", "password":
"123456", "language": "en"}'
```

Response

```
{
  "alias": "alice",
  "xpub":
"a85e6eccb22f4c5fdade905f9a969003a17b6f35c237183a4313354b819a92689d52da3bcfe55f15a5508
77e8d789bd2bb9620f46e5049ea36470ab1b588a986",
  "file": "/home/yang/.eiyaro/keystore/UTC--2024-3-10T07-09-17.509894697Z--341695b9-
9223-470c-a26d-bea210f8e1bb",
  "mnemonic": "verb smoke glory dentist annual peanut oval dragon fiction current
orbit lab load total language female mushroom coyote regular toy slide welcome employ
three"
}
```

Create key by specified mnemonic:

Request

```
curl -X POST http://localhost:9888/create-key -d '{"alias":"jack",
"password":"123456", "mnemonic":"please observe raw beauty blue sea believe then boat
float beyond position", "language":"en"}'
```

Response

```
{
  "alias": "jack",
  "xpub":
"c7bcb65febd31c6d900bc84c386d95c3d5b047090628d9bf5c51a848945b6986e99ff70388018a7681fa3
7a240dbd8df39a994c86f9314a61e75feb33563ca72",
  "file": "/home/yang/.eiyaro/keystore/UTC--2024-3-10T07-08-51.815030323Z--46ee932e-
88d3-4680-a5c1-dd9e63918fcc"
}
```

List Keys Endpoint

Returns the list of all available keys.

Parameters

None.

Returns

- `Array of Object`, keys owned by the client.
  - `Object`:
    - `String` - **alias**, name of the key.
    - `String` - **xpub**, pubkey of the key.

Example

Request a list of the current keys on the node.

Request

```
curl -X POST http://localhost:9888/list-keys
```

Response

```
[
  {
    "alias": "alice",
    "xpub":
"a7dae957c2d35b42efe7e6871cf5a75ebd2a0d0e51caffe767db42d3e6d69dbe211d1ca492ecf05908fe6
fa625ad61b3253375ea744c9442dd5551613ba50aea",
    "file": "/Path/To/Library/Eiyaro/keystore/UTC--2024-03-21T02-35-15.035935116Z--
4f2b8bd7-0576-4b82-8941-6cc6da05efe3"
  },
  {
    "alias": "bob",
    "xpub":
"d30a810e88532f73816b7b5007d413cbd21e526ae9159023e5262511893adc1526b8eacd691b27c080201
d7d79336a4f3d2cb4c167d997821cad445765916254",
    "file": "/Path/To/Library/Eiyaro/keystore/UTC--2018-03-22T06-30-27.609315219Z--
0e34293c-8856-4f5f-b934-37456a3820fa"
  }
]
```

Update Key Alias Endpoint

Update the alias for an existing key.

Parameters

`Object`:

- `String` - **xpub**, pubkey of the key.
- `String` - **new_alias**, new alias of the key.

Returns

Nothing in case the key alias is updated successfully.

Example

Update an existing key's alias.

Request

```
curl -X POST http://localhost:9888/update-key-alias -d '{"xpub":
"a7dae957c2d35b42efe7e6871cf5a75ebd2a0d0e51caffe767db42d3e6d69dbe211d1ca492ecf05908fe6
fa625ad61b3253375ea744c9442dd5551613ba50aea", "new_alias": "new_key"}'
```

Response

Nothing if the operation was successful.

Delete Key Endpoint

Deletes an existing key.

Please make sure that there is no balance in the related accounts.

Parameters

`Object`:

- `String` - **xpub**, pubkey of the key.
- `String` - **password**, password of the key.

Returns

Nothing in case the key is deleted successfully.

Example

Delete an existing key.

Request

```
curl -X POST {bas-url}delete-key -d '{"xpub":
"a7dae957c2d35b42efe7e6871cf5a75ebd2a0d0e51caffe767db42d3e6d69dbe211d1ca492ecf05908fe6
fa625ad61b3253375ea744c9442dd5551613ba50aea", "password": "123456"}'
```

Response

Nothing if the operation was successful.

Check Key Password Endpoint

Check an existing key’s password.

Parameters

`Object`:

- `String` - **xpub**, pubkey of the key.
- `String` - **password**, password of the key.

Returns

`Object`:

- `Boolean` - **check_result**, if check is successful the value will be true, otherwise it will be false.

Example

Check the password for an existing key.

Request

```
curl -X POST http://localhost:9888/check-key-password -d '{"xpub":
"a7dae957c2d35b42efe7e6871cf5a75ebd2a0d0e51caffe767db42d3e6d69dbe211d1ca492ecf05908fe6
fa625ad61b3253375ea744c9442dd5551613ba50aea", "password": "123456"}'
```

Response

```
{
  "check_result": true
}
```

Reset Key Password Endpoint

Reset an existing key's password.

Parameters

`Object`:

- `String` - **xpub**, pubkey of the key.
- `String` - **old_password**, old password of the key.
- `String` - **new_password**, new password of the key.

Returns

`Object`:

- `Boolean` - **changed**, if reset is successful the value will be true, otherwise it will be false.

Example

Reset the password for an existing key.

Request

```
curl -X POST http://localhost:9888/reset-key-password -d '{"xpub":
"a7dae957c2d35b42efe7e6871cf5a75ebd2a0d0e51caffe767db42d3e6d69dbe211d1ca492ecf05908fe6
fa625ad61b3253375ea744c9442dd5551613ba50aea", "old_password": "123456",
"new_password": "654321"}'
```

Response

```
{
  "changed": true
}
```

Create Account Endpoint

Create an account to manage addresses.
Single sign account contains only one `root_xpubs` and quorum; however multi sign account can contain any number of `root_xpubs` and quorum.
Quorum is the number of verify signatures, the range is `[1, len(root_xpubs)]`.

Parameters

`Object`:

- `Array of String` - **root_xpubs**, pubkey array.
- `String` - **alias**, name of the account.
- `Integer` - **quorum**, the default value is `1`, threshold of keys that must sign a transaction to spend asset units controlled by the account.

Optional:

- `String` - **access_token**, if optional when creating account locally. However, if you want to create account remotely, it's indispensable.

Returns

`Object`:

- `String` - **id**, account id.
- `String` - **alias**, name of account.
- `Integer` - **key_index**, key index of account.
- `Integer` - **quorum**, threshold of keys that must sign a transaction to spend asset units controlled by the account.
- `Array of Object` - **xpubs**, pubkey array.

Example

Create an account with a given `root_xpubs` and `alias`.

Request

```
curl -X POST http://localhost:9888/create-account -d
'{"root_xpubs":["2d6c07cb1ff7800b0793e300cd62b6ec5c0943d308799427615be451ef09c0304bee5
dd492c6b13aaa854d303dc4f1dcb229f9578786e19c52d860803efa3b9a"],"quorum":1,"alias":"alic
e"}'
```

Response

```
{
  "alias": "alice",
  "id": "08FO663C00A02",
  "key_index": 1,
  "quorum": 1,
  "xpubs": [

"2d6c07cb1ff7800b0793e300cd62b6ec5c0943d308799427615be451ef09c0304bee5dd492c6b13aaa854
d303dc4f1dcb229f9578786e19c52d860803efa3b9a"
  ]
}
```

List Accounts Endpoint

Returns a list of the available accounts on the node.

Parameters

Optional:

- `String` - **id**, account id.
- `String` - **alias**, name of account.

Returns

- `Array of Object`, account array.
  - `Object`:
    - `String` - **id**, account id.
    - `String` - **alias**, name of account.
    - `Integer` - **key_index**, key index of account.
    - `Integer` - **quorum**, threshold of keys that must sign a transaction to spend asset units controlled by the account.

    - `Array of Object` - **xpubs**, pubkey array.

Example

Request a list of the accounts present on the node.

Request

```
curl -X POST http://localhost:9888/list-accounts -d '{"alias":"alice"}'
```

Response

```
[
  {
    "alias": "alice",
    "id": "086KQD75G0A02",
    "key_index": 1,
    "quorum": 1,
    "xpubs": [
"180aab8bf247932a7cf68da5cc9a873266279155097612f1e5fdda4add88d5e91e2e7ce5b736f3ac93382
4cdee9effcf1531b90dfcb388e5cc306d14e9a2c85e"
    ]
  }
]
```

Update Account Alias Endpoint

Updates an alias for the an existing account.

Parameters

`Object`: **account_alias** | **account_id**
* `String` - **new_alias**, new alias of account.

optional:

- `String` - **account_alias**, alias of account.
- `String` - **account_id**, id of account.

Returns

Nothing in case the account alias is updated successfully.

Example

Update the alias for a given account ID or an account alias.

Request

```
curl -X POST http://localhost:9888/update-account-alias -d '{"account_id":
"08F0663C00A02", "new_alias": "new_account"}'
# or
curl -X POST http://localhost:9888/update-account-alias -d '{"account_alias": "alice",
"new_alias": "new_account"}'
```

Response

Nothing if the operation was successful.

Delete Account Endpoint

Delete an existing account.

⚠ Please make sure that there is no balance in the related accounts.

Parameters

`Object`: **account_alias** | **account_id**

Optional:

- `String` - **account_alias**, alias of account.
- `String` - **account_id**, id of account.

Returns

Nothing if the account is deleted successfully.

Example

Delete an existing account by account ID or account alias.

Request

```
curl -X POST http://localhost:9888/delete-account -d '{"account_id": "08F0663C00A02"}'
# or
curl -X POST http://localhost:9888/delete-account -d '{"account_alias": "alice"}'
```

Response

Nothing if the operation was successful.

Create Account Receiver Endpoint

Creates an address and control program.
The address and control program are a one to one relationship.
In the `build-transaction` endpoint, the receiver is the address when the action is of type `control_address`, and the receiver is the control program when the action is of type `control_program`, both can be used to the same effect.

Parameters

`Object`: **account_alias** | **account_id**

Optional:

- `String` - **account_alias**, alias of account.
- `String` - **account_id**, id of account.

Returns

`Object`:

- `String` - **address**, address of account.
- `String` - **control_program**, control program of account.

Example

Create an account alias on the existing account ID.

Request

```
curl -X POST http://localhost:9888/create-account-receiver -d '{"account_alias":
"alice", "account_id": "0BDQARM800A02"}'
```

Response

```
{
    "address": "ey1q5u8u4eldhjf3lvnkmyl78jj8a75neuryzlknk0",
    "control_program": "0014a70fcae7edbc931fb276d93fe3ca47efa93cf064"
}
```

List Addresses Endpoint

Returns the sub list of all available addresses by account with a limit count.

Parameters

- `String` - **account_alias**, alias of account.
- `String` - **account_id**, id of account.
- `Integer` - **from**, the start position of first address
- `Integer` - **count**, the number of returned

Returns

- `Array of Object`, account address array.
  - `Object`:
    - `String` - **account_alias**, alias of account.
    - `String` - **account_id**, id of account.
    - `String` - **address**, address of account.
    - `Boolean` - **change**, whether the account address is change.

Example

List three addresses from first position by `account_id` or `account_alias`

Request

```
curl -X POST http://localhost:9888/list-addresses -d '{"account_alias": "alice",
"account_id": "086KQD75G0A02", "from": 0, "count": 3}'
```

Response

```
[
  {
    "account_alias": "alice",
    "account_id": "086KQD75G0A02",
    "address": "ey1qcn9lf7nxhswratvmg6d78nq7r7yupm36qgsv55",
    "change": false
  },
  {
    "account_alias": "alice",
    "account_id": "086KQD75G0A02",
    "address": "ey1qew4h5uvt5ssrtg2alms0j77r94c30m78ucrcxy",
    "change": false
  },
  {
    "account_alias": "alice",
    "account_id": "086KQD75G0A02",
    "address": "ey1qgnp4lte7wge0rsekevjlrdh39vkzz0c2alheue",
    "change": false
  }
]
```

Validate Address Endpoint

Validate that the address is valid and report if it is local or not.

Parameters

`Object`:

- `string` - **address**, address of account.

Returns

`Object`:

- `Boolean` - **valid**, whether the account address is valid.
- `Boolean` - **is_local**, whether the account address is local.

Example

Request the validity of an address.

Request

```
curl -X POST http://localhost:9888/validate-address -d '{"address":
"ey1qcn9lf7nxhswratvmg6d78nq7r7yupm36qgsv55"}'
```

Response

```
{
  "valid": true,
  "is_local": true,
}
```

Get Mining Address Endpoint

Query the current mining address.

Parameters

None.

Returns

`Object`:

- `String` - **mining_address**, the current mining address being used.

Example

Request the current mining address.

Request

```
curl -X POST http://localhost:9888/get-mining-address
```

Response

```
{
    "mining_address":"ey1qnhr65jq3q9gf8uymza8vp0ew8tfyh642wddxh6"
}
```

Set Mining Address Endpoint

Set the current mining address, no matter wether the address is a local one or not.
It returns an error message if the address format is incorrect.

Parameters

`Object`:

- `String` - **mining_address**, mining address to set.

Returns

`Object`:

- `String` - **mining_address**, the new mining address.

Example

Update the node's mining address.

Request

```
curl -X POST http://localhost:9888/set-mining-address -d
'{"mining_address":"ey1qnhr65jq3q9gf8uymza8vp0ew8tfyh642wddxh6"}'
```

Response

```
{
    "mining_address":"ey1qnhr65jq3q9gf8uymza8vp0ew8tfyh642wddxh6"
}
```

Get Coinbase Arbitrary Endpoint

Get coinbase arbitrary.

Parameters

None.

Returns

`Object`:

- `String` - **arbitrary**, the arbitrary data append to coinbase, in hexadecimal format. (The full coinbase data for a block will be `0x00&block_height&arbitrary`.)

Example

Query for the coinbase arbitrary.

Request

```
curl -X POST http://localhost:9888/get-coinbase-arbitrary
```

Response

```
{
    "arbitrary":"ff"
}
```

Set Coinbase Arbitrary Endpoint

Set coinbase arbitrary.

Parameters

`Object`:

- `String` - **arbitrary**, the arbitrary data to be appended to coinbase, in hexadecimal format.

Returns

`Object`:

- `String` - **arbitrary**, the arbitrary data being appended to coinbase, in hexadecimal format. (The full coinbase data for a block will be `0x00&block_height&arbitrary`.)

Example

Set the coinbase arbitrary.

Request

```
curl -X POST http://localhost:9888/set-coinbase-arbitrary -d '{"arbitrary":"ff"}'
```

Response

```
{
    "arbitrary":"ff"
}
```

List `pubkeys` Endpoint

Returns the list of all available `pubkeys` by account.

Parameters

`Object`: **account_alias** | **account_id** | **public_key**

Optional:

- `String` - **account_alias**, alias of account.
- `String` - **account_id**, id of account.
- `string` - **public_key**, public key.

Returns

`Object`:

- `String` - **root_xpub**, root xpub.
- `Array of Object` -**pubkey_infos**, public key array.
    - `String` - **pubkey**, public key.
    - `Object` - **derivation_path**, derivation path for root xpub.

Example

Query for the list of `pubkeys` by account ID or account alias.

Request

```
curl -X POST http://localhost:9888/list-pubkeys -d '{"account_id": "0GO0LLUV00A02"}'
```

Response

```
{
  "pubkey_infos": [
    {
      "derivation_path": [
        "010100000000000000",
        "0100000000000000"
      ],
      "pubkey": "b7730319feac582056379548360da5c08258e248e5c29de08a97a6614df1425d"
    },
    {
      "derivation_path": [
        "010100000000000000",
        "0200000000000000"
      ],
      "pubkey": "5044a0d6113faaf4cb2550f63a820ab579a2af6134e503b76378490d5fe75af4"
    },
    {
      "derivation_path": [
        "010100000000000000",
        "0300000000000000"
      ],
      "pubkey": "ff5c28ce257b25c2a6e172ded490a708a8e654253836d92eb0a68b81ce63bea3"
    }
  ],
  "root_xpub":
"94a909319eac179f7694b99b8367b9c02b4414b95961e2e3a5bd887e0616af05a7c5e4448df92cd6cdfd82e57cd7aefc1ee0a7fd0d6a2194b5e5faf82556bedc"
}
```

Create Asset Endpoint

Create an asset definition, it prepares for the issuance of an asset.

Parameters

`Object`:

- `String` - **alias**, name of the asset.

- `Object` - **definition**, definition of asset.

Optional:(please pick one from the following two ways)

- `Array of String` - **root_xpubs**, xpub array.
- `Integer` - **quorum**, the default value is `1`, threshold of keys that must sign a transaction to spend asset units controlled by the account.

or

- `String` - **issuance_program**, user-defined contract program.

Returns

`Object`:

- `String` - **id**, asset id.
- `String` - **alias**, name of the asset.
- `String` - **issuance_program**, control program of the issuance of asset.
- `Array of Object` - **keys**, information of asset pubkey.
- `String` - **definition**, definition of asset.
- `Integer` - **quorum**, threshold of keys that must sign a transaction to spend asset units controlled by the account.

Example

Create an asset by `xpubs`:

Request

```
curl -X POST http://localhost:9888/create-asset -d '{"alias": "GOLD", "root_xpubs":
["f6a16704f745a168642712060e6c5a69866147e21ec2447ae628f87d756bb68cc9b91405ad0a95f00409
0e864fde472f62ba97053ea109837bc89d63a64040d5"], "quorum":1}'
```

Response

```
{
  "id": "3c1cf4c9436e3f942cb2f1d70a584f1c61df3697698dacccdc89e46f46a003d0",
  "alias": "GOLD",
  "issuance_program":
"766baa209683b893483c0a5a317bf9868a8e2a09691f8aa8c1f3e2a7bb62b157e76712e05151ad696c00c
0",
  "keys": [
    {
      "root_xpub":
"f6a16704f745a168642712060e6c5a69866147e21ec2447ae628f87d756bb68cc9b91405ad0a95f004090
e864fde472f62ba97053ea109837bc89d63a64040d5",
      "asset_pubkey":
"9683b893483c0a5a317bf9868a8e2a09691f8aa8c1f3e2a7bb62b157e76712e012bd443fa7d56a0627df0
a29dffcdc52641672a0f5cba54d104ad76ebeb8dfc3",
      "asset_derivation_path": [
        "0002000000000000000"
      ]
    }
  ],
  "quorum": 1,
  "definition": {}
}
```

Create an asset by `issuance_program`:

Request

```
curl -X POST http://localhost:9888/create-asset -d '{"alias":
"TESTASSET","issuance_program":
"20e9108d3ca8049800727f6a3505b3a2710dc579405dde03c250f16d9a7e1e6e78160014c5a5b563c4623
018557fb299259542b8739f6bc20163201e074b22ed7ae8470c7ba5d8a7bc95e83431a753a17465e8673af
68a82500c22741a547a6413000000007b7b51547ac1631a000000547a547aae7cac00c0",
"definition":{"name":"TESTASSET","symbol":"TESTASSET","decimals":8,"description":{}}}'
```

Response

```
{
  "id": "59621aa82c047bd21f73711d4a7905b7a9fbb49bc1a3fdc309b13807cc8b9094",
  "alias": "TESTASSET",
  "issuance_program":
"20e9108d3ca8049800727f6a3505b3a2710dc579405dde03c250f16d9a7e1e6e78160014c5a5b563c4623
018557fb299259542b8739f6bc20163201e074b22ed7ae8470c7ba5d8a7bc95e83431a753a17465e8673af
68a82500c22741a547a6413000000007b7b51547ac1631a000000547a547aae7cac00c0",
  "keys": null,
  "quorum": 0,
  "definition": {
    "decimals": 8,
    "description": {},
    "name": "TESTASSET",
    "symbol": "TESTASSET"
  }
}
```

Get Asset Endpoint

Query asset details by asset ID.

Parameters

`Object`:

- `String` - **id**, id of asset.

Returns

`Object`:

- `String` - **id**, asset id.
- `String` - **alias**, name of the asset.
- `String` - **issuance_program**, control program of the issuance of asset.
- `Integer` - **key_index**, index of key for xpub.
- `Integer` - **quorum**, threshold of keys that must sign a transaction to spend asset units controlled by the account.
- `Array of Object` - **xpubs**, pubkey array.
- `String` - **type**, type of asset.
- `Integer` - **vm_version**, version of VM.
- `String` - **raw_definition_byte**, byte of asset definition.
- `Object` - **definition**, description of asset.

Example

Get asset details by asset ID.

Request

```
curl -X POST http://localhost:9888/get-asset -d '{"id":
"50ec80b6bc48073f6aa8fa045131a71213c33f3681203b15ddc2e4b81f1f4730"}'
```

Response

```
{
  "alias": "SILVER",
  "definition": null,
  "id": "50ec80b6bc48073f6aa8fa045131a71213c33f3681203b15ddc2e4b81f1f4730",
  "issue_program":
"ae2029cd61d9ef31d40af7541f9a50831d6317fdb0870249d0564fcfa9a8f843589c5151ad",
  "key_index": 1,
  "quorum": 1,
  "raw_definition_byte": "",
  "type": "asset",
  "vm_version": 1,
  "xpubs": [
"34b16ee500615cd325f8b84099f83c1ebecaca67977c5dc9b71ae32ceaf18207f996b0a9725b901d37926
89b2babcb60febe3b81a684d9b56b65f67f307d453d"
  ]
}
```

List Assets Endpoint

Returns the list of all available assets.

Parameters

None.

Returns

- `Array of Object`, asset array.
  - `Object`:
    - `String` - **id**, asset id.
    - `String` - **alias**, name of the asset.
    - `String` - **issuance_program**, control program of the issuance of asset.
    - `Integer` - **key_index**, index of key for xpub.
    - `Integer` - **quorum**, threshold of keys that must sign a transaction to spend asset units controlled by the account.
    - `Array of Object` - **xpubs**, pubkey array.
    - `String` - **type**, type of asset.
    - `Integer` - **vm_version**, version of VM.
    - `String` - **raw_definition_byte**, byte of asset definition.
    - `Object` - **definition**, description of asset.

Example

List all the available assets.

Request

```
curl -X POST http://localhost:9888/list-assets -d '{}'
```

Response

```
[
  {
    "alias": "EY",
    "definition": {
      "decimals": 8,
      "description": "Eiyaro Official Issue",
      "name": "EY",
      "symbol": "EY"
    },
    "id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
    "issue_program": "",
    "key_index": 0,
    "quorum": 0,
    "raw_definition_byte":
"7b0a202022646563696d616c73223a20382c0a2020226465736372697074696f6e223a20224279746f6d2
04f666696369616c204973737565222c0a2020226e616d65223a202262746d222c0a20202273796d626f6
c223a202262746d220a7d",
    "type": "internal",
    "vm_version": 1,
    "xpubs": null
  },
  {
    "alias": "SILVER",
    "definition": null,
    "id": "50ec80b6bc48073f6aa8fa045131a71213c33f3681203b15ddc2e4b81f1f4730",
    "issue_program":
"ae2029cd61d9ef31d40af7541f9a50831d6317fdb0870249d0564fcfa9a8f843589c5151ad",
    "key_index": 1,
    "quorum": 1,
    "raw_definition_byte": "",
    "type": "asset",
    "vm_version": 1,
    "xpubs": [

"34b16ee500615cd325f8b84099f83c1ebecaca67977c5dc9b71ae32ceaf18207f996b0a9725b901d37926
89b2babcb60febe3b81a684d9b56b65f67f307d453d"
    ]
  }
]
```

Update Asset Alias Endpoint

Update asset alias by assetID.

Parameters

`Object`:

- `String` - **id**, id of asset.
- `String` - **alias**, new alias of asset.

Returns

Nothing the asset alias is updated successfully.

Example

Update asset alias.

Request

```
curl -X POST http://localhost:9888/update-asset-alias -d
'{"id":"50ec80b6bc48073f6aa8fa045131a71213c33f3681203b15ddc2e4b81f1f4730",
"alias":"GOLD"}'
```

Response

Nothing if the operation was successful.

List Balances Endpoint

Returns the list of all available accounts' balances.

Parameters

Optional:

- `String` - **account_id**, account id.
- `String` - **account_alias**, name of account.

Returns

- `Array of Object`, balances owned by the account.
    - `Object`:
        - `String` - **account_id**, account id.
        - `String` - **account_alias**, name of account.
        - `String` - **asset_id**, asset id.
        - `String` - **asset_alias**, name of asset.
        - `Integer` - **amount**, specified asset balance of account.

Example

List all the available accounts' balances.

Request

```
curl -X POST http://localhost:9888/list-balances -d '{}'
```

Response

```
[
  {
    "account_alias": "default",
    "account_id": "0BDQ9AP100A02",
    "amount": 35508000000000,
    "asset_alias": "EY",
    "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"
  },
  {
    "account_alias": "alice",
    "account_id": "0BDQARM800A04",
    "amount": 60000000000,
    "asset_alias": "EY",
    "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"
  }
]
```

List available accounts' balances by a given `account_id`:

Request

```
curl -X POST http://localhost:9888/list-balances -d '{"account_id":"0BDQ9AP100A02"}'
```

Response

```
[
  {
    "account_alias": "default",
    "account_id": "0BDQ9AP100A02",
    "amount": 35508000000000,
    "asset_alias": "EY",
    "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff"
  }
]
```

List Unspent Outputs Endpoint

Returns the sub list of all available unspent outputs for all accounts in your wallet.

Parameters

`Object`:

Optional:

- `String` - **id**, id of unspent output.
- `Boolean` - **unconfirmed**, is include unconfirmed `utxo`
- `Boolean` - **smart_contract**, is contract `utxo`
- `Integer` - **from**, the start position of first `utxo`
- `Integer` - **count**, the number of returned
- `String` - **account_id**, account id.
- `String` - **account_alias**, name of account.

Returns

- `Array of Object`, unspent output array.
  - `Object`:
    - `String` - **account_id**, account id.
    - `String` - **account_alias**, name of account.
    - `String` - **asset_id**, asset id.
    - `String` - **asset_alias**, name of asset.
    - `Integer` - **amount**, specified asset balance of account.
    - `String` - **address**, address of account.
    - `Boolean` - **change**, whether the account address is change.
    - `String` - **id**, unspent output id.
    - `String` - **program**, program of account.
    - `String` - **control_program_index**, index of program.
    - `String` - **source_id**, source unspent output id.
    - `String` - **source_pos**, position of source unspent output id in block.
    - `String` - **valid_height**, valid height.

Example

List all the available unspent outputs:

Request

```
curl -X POST http://localhost:9888/list-unspent-outputs -d '{}'
```

Response

```
[
  {
    "account_alias": "alice",
    "account_id": "0BKBR6VR00A06",
    "address": "ey1qv3htuvug7qdv46ywcvvzytrwrsyg0swltfa0dm",
    "amount": 2000,
    "asset_alias": "GOLD",
    "asset_id": "1883cce6aab82cf9af8cd085a3115dd4a92cdb8e6a9152acd73d7ae4adb9030a",
    "change": false,
    "control_program_index": 2,
    "id": "58f29f0f85f7bd2a91088bcbe536dee41cd0642dfb1480d3a88589bdbfd642d9",
    "program": "0014646ebe3388f01acae88ec318222c6e1c0887c1df",
    "source_id": "5988c1630c1f325e69bb92cb4b19af14286aa107311bc64b8f1a54629a33e0f4",
    "source_pos": 2,
    "valid_height": 0
  },
  {
    "account_alias": "default",
    "account_id": "0BKBR2D2G0A02",
    "address": "ey1qx7ylnhszg24995d5e0nftu9e87kt9vnxcn633r",
    "amount": 624000000000,
    "asset_alias": "EY",
    "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
    "change": false,
    "control_program_index": 12,
    "id": "5af9d3c9b69470983377c1fc0c9125c4ac3bfd32c8d505f2a6042aade8503bc9",
    "program": "00143789f9de0242aa52d1b4cbe695f0b93facb2b266",
    "source_id": "233d1dd49e591980f98e11f333c6c28a867e78448e272011f045131df5aa260b",
    "source_pos": 0,
    "valid_height": 12
  }
]
```

List the unspent output matching the given id:

Request

```
curl -X POST http://localhost:9888/list-unspent-outputs -d '{"id":
"58f29f0f85f7bd2a91088bcbe536dee41cd0642dfb1480d3a88589bdbfd642d9"}'
```

Response

```
{
  "account_alias": "alice",
  "account_id": "0BKBR6VR00A06",
  "address": "ey1qv3htuvug7qdv46ywcvvzytrwrsyg0swltfa0dm",
  "amount": 2000,
  "asset_alias": "GOLD",
  "asset_id": "1883cce6aab82cf9af8cd085a3115dd4a92cdb8e6a9152acd73d7ae4adb9030a",
  "change": false,
  "control_program_index": 2,
  "id": "58f29f0f85f7bd2a91088bcbe536dee41cd0642dfb1480d3a88589bdbfd642d9",
  "program": "0014646ebe3388f01acae88ec318222c6e1c0887c1df",
  "source_id": "5988c1630c1f325e69bb92cb4b19af14286aa107311bc64b8f1a54629a33e0f4",
  "source_pos": 2,
  "valid_height": 0
}
```

Backup Wallet Endpoint

Backs up a wallet to an image file, it contains the accounts' image, the assets' image and the keys' image.

Parameters

None.

Returns

`Object`:

- `Object` - **account_image**, account image.
- `Object` - **asset_image**, asset image.
- `Object` - **key_images**, key image.

Example

Request a backup of the node's wallet information.

Request

```
curl -X http://localhost:9888/backup-wallet -d '{}'
```

Response

```
{
  "account_image": {
    "slices": [
      {
        "account": {
          "type": "account",
          "xpubs": [
"395d6e0ac25978c3f52f9c7bdfdf75ce6af02639fd7875b4b1f40778ab1120c6dcf461b7ab6fd310983af
b54a9a0fb3e09b6ec0d4364c4808c94383d50fb0681"
          ],
          "quorum": 1,
          "key_index": 1,
          "ID": "0CQTA3EOG0A02",
          "Alias": "def"
        },
        "contract_index": 2
      }
    ]
  },
  "asset_image": {
    "assets": []
  },
  "key_images": {
    "xkeys": [
      {
        "crypto": {
          "cipher": "aes-128-ctr",
          "ciphertext":
"bf44766fec149478af9500e25ce0a6bc50bb2fa04e40465781da6ff64e9b3a4c9af3d214cd92c5a41d849
8db5f4376526740f960ff429b16e52876aec6860e1d",
          "cipherparams": {
            "iv": "1b0fc61ae4dacb15f0f77d2b4ba67635"
          },
          "kdf": "scrypt",
          "kdfparams": {
            "dklen": 32,
            "n": 4096,
            "p": 6,
            "r": 8,
            "salt": "e133b1e7caae771ff1ab34b14824d6e27ef399f2b7ded4ad3500f080ede4a1dd"
          },
          "mac": "bc6bf411fb63e61a17bc15b94f29cf0d5a0f084c328955da1f7e2b26757cfc23"
        },
        "id": "1f40be59-7400-4fdc-b46b-15009f65363a",
        "type": "eiyaro_kd",
        "version": 1,
        "alias": "default",
        "xpub":
"c4ec9bfd5df19d175e17ff7fed89193c37a4a64e1c0928387da01387ca76c3bfd99390e3373ec4d438522
cc2d4644214cd2ec3b00965f7a1fa3546809583191c"
      },
      {
        "crypto": {
          "cipher": "aes-128-ctr",
          "ciphertext":
"f0887c8603cbbafc0a66d5b45f71488e089708c7dea4342625a67858a49d6d08c79cd3f1800627e3c8b46
68e8df34fcf0be9df5d9d4503acff05373976c312a9",
          "cipherparams": {
            "iv": "c111b46f9104f49f2c40aedb827e53b5"
          },
          "kdf": "scrypt",
          "kdfparams": {
            "dklen": 32,
            "n": 4096,
            "p": 6,
            "r": 8,
            "salt": "d9ef588b258b111dea1d99a4e4c5a4f968ab69072176bb95b111922e3bbea9e6"
          },
          "mac": "336f5fee643776e139f05ebe5e4f209d992ff97e16b906105fadac9e86133554"
        },
        "id": "611d407c-9e97-4297-a02a-13cd68e47983",
        "type": "eiyaro_kd",
        "version": 1,
        "alias": "def",
        "xpub":
"395d6e0ac25978c3f52f9c7bdfdf75ce6af02639fd7875b4b1f40778ab1120c6dcf461b7ab6fd310983af
b54a9a0fb3e09b6ec0d4364c4808c94383d50fb0681"
      }
    ]
  }
}
```

Restore Wallet Endpoint

Restores the wallet by image file.

Parameters

`Object`:

- `Object` - **account_image**, account image.
- `Object` - **asset_image**, asset image.
- `Object` - **key_images**, key image.

Returns

Nothing if the operation was successful.

Example

Restore a node’s wallet via the image file.

Request

```
curl -X POST http://localhost:9888/restore-wallet -d
```

```
'{"account_image":{"slices":[{"account":{"type":"account","xpubs":["395d6e0ac25978c3f5
2f9c7bdfdf75ce6af02639fd7875b4b1f40778ab1120c6dcf461b7ab6fd310983afb54a9a0fb3e09b6ec0d
4364c4808c94383d50fb0681"],"quorum":1,"key_index":1,"ID":"0CQTA3EOG0A02","Alias":"def"
},"contract_index":2}]},"asset_image":{"assets":[]},"key_images":{"xkeys":[{"crypto":{
"cipher":"aes-128-
ctr","ciphertext":"bf44766fec149478af9500e25ce0a6bc50bb2fa04e40465781da6ff64e9b3a4c9af
3d214cd92c5a41d8498db5f4376526740f960ff429b16e52876aec6860e1d","cipherparams":{"iv":"1
b0fc61ae4dacb15f0f77d2b4ba67635"},"kdf":"scrypt","kdfparams":{"dklen":32,"n":4096,"p":
6,"r":8,"salt":"e133b1e7caae771ff1ab34b14824d6e27ef399f2b7ded4ad3500f080ede4a1dd"},"ma
c":"bc6bf411fb63e61a17bc15b94f29cf0d5a0f084c328955da1f7e2b26757cfc23"},"id":"1f40be59-
7400-4fdc-b46b-
15009f65363a","type":"eiyaro_kd","version":1,"alias":"default","xpub":"c4ec9bfd5df19d1
75e17ff7fed89193c37a4a64e1c0928387da01387ca76c3bfd99390e3373ec4d438522cc2d4644214cd2ec
3b00965f7a1fa3546809583191c"},{"crypto":{"cipher":"aes-128-
ctr","ciphertext":"f0887c8603cbbafc0a66d5b45f71488e089708c7dea4342625a67858a49d6d08c79
cd3f1800627e3c8b4668e8df34fcf0be9df5d9d4503acff05373976c312a9","cipherparams":{"iv":"c
111b46f9104f49f2c40aedb827e53b5"},"kdf":"scrypt","kdfparams":{"dklen":32,"n":4096,"p":
6,"r":8,"salt":"d9ef588b258b111dea1d99a4e4c5a4f968ab69072176bb95b111922e3bbea9e6"},"ma
c":"336f5fee643776e139f05ebe5e4f209d992ff97e16b906105fadac9e86133554"},"id":"611d407c-
9e97-4297-a02a-
13cd68e47983","type":"eiyaro_kd","version":1,"alias":"def","xpub":"395d6e0ac25978c3f52
f9c7bdfdf75ce6af02639fd7875b4b1f40778ab1120c6dcf461b7ab6fd310983afb54a9a0fb3e09b6ec0d4
364c4808c94383d50fb0681"}]}}'
```

Response

Nothing if the operation was successful.

Rescan Wallet Endpoint

Trigger a rescan of the block information on the wallet.

Parameters

None.

Returns

Nothing if operation was a success.

Example

Request a rescan of the block information on the node.

Request

```
curl -X POST http://localhost:9888/rescan-wallet -d '{}'
```

Response

Nothing if the operation was successful.

Recovery Wallet Endpoint

Recovers a wallet and it’s accounts from root `xpubs`.
All accounts and balances of `bip44` multi-account hierarchy for deterministic wallets can be restored via root `xpubs`.

Parameters

`Object`:

- `Object` - **xpubs**, root XPubs.

Returns

Status of recovery wallet operation.

Example

Request a wallet’s recovery via `xpubs`.

Request

```
curl -X POST http://localhost:9888/recovery-wallet -d '{
"xpubs":["c536a2c11fafd8278e02e9393dcbf5aa420eb51a1761a7e5da7f2b9b37969b52a8f8e2b692e7
dcaf79dfa0d1e28c63eb9fda42942f20feaa8a71b383d9a4668c"]}'
```

Response

```
{
    "status": "success"
}
```

Wallet Info Endpoint

Returns the wallet’s information.

Parameters

None.

Returns

`Object`:

- `Integer` - **best_block_height**, current block height.
- `Integer` - **wallet_height**, current block height for wallet.

Example

Request the node's wallet information.

Request

```
curl -X POST http://localhost:9888/wallet-info -d '{}'
```

Response

```
{
  "best_block_height": 150,
  "wallet_height": 150
}
```

Sign Message Endpoint

Sign a message with the key password(decode encrypted private key) of an address.

Parameters

`Object`:

- `String` - **address**, address for account.
- `String` - **message**, message for signature by address xpub.
- `String` - **password**, password of account.

Returns

`Object`:

- `String` - **derived_xpub**, derived xpub.
- `String` - **signature**, signature of message.

Example

Request the signature of a message by an address' private key.

Request

```
curl -X POST http://localhost:9888/sign-message -d
'{"address":"ey1qx2qgvvjz734ur8x5lpfdtlau74aaa5djs0a5jn", "message":"this is a test
message", "password":"123456"}'
```

Response

```
{
  "signature":
"74da3d6572233736e3a439166719244dab57dd0047f8751b1efa2da26eeab251d915c1211dcad77e8b013
267b86d96e91ae67ff0be520ef4ec326e911410b609",
  "derived_xpub":
"6ff8c3d1321ce39a3c3550f57ba70b67dcbcef821e9b85f6150edb7f2f3f91009e67f3075e6e76ed5f657
ee4b1a5f4749b7a8c74c8e7e6a1b0e5918ebd5df4d0"
}
```

Decode Program Endpoint

Decode a program.

Parameters

`Object`:

- `String` - **program**, program for account.

Returns

`Object`:

- `String` - **instructions**, instructions and data for program.

Example

Request to have a program decoded into it’s instructions.

Request

```
curl -X POST http://localhost:9888/decode-program -d
'{"program":"0014a86c83ee12e6d790fb388345cc2e2b87056a0773"}'
```

Response

```
{
  "instructions": "DUP \nHASH160 \nDATA_20 a86c83ee12e6d790fb388345cc2e2b87056a0773
\nEQUALVERIFY \nTXSIGHASH \nSWAP \nCHECKSIG \n"
}
```

Get Transaction Endpoint

Query the account related transaction by transaction ID.

Parameters

`Object`:

- `String` - **tx_id**, transaction id, hash of transaction.

Returns

`Object`:

- `String` - **tx_id**, transaction id, hash of the transaction.
- `Integer` - **block_time**, the unix timestamp for when the requst was responsed.
- `String` - **block_hash**, hash of the block where this transaction was in.
- `Integer` - **block_height**, block height where this transaction was in.
- `Integer` - **block_index**, position of the transaction in the block.
- `Integer` - **block_transactions_count**, transactions count where this transaction was in the block.
- `Boolean` - **status_fail**, whether the state of the transaction request has failed.
- `Integer` - **size**, size of transaction.
- `Array of Object` - **inputs**, object of inputs for the transaction.
  - `String` - **type**, the type of input action, available option include: 'spend', 'issue', 'coinbase'.
  - `String` - **asset_id**, asset id.
  - `String` - **asset_alias**, name of asset.
  - `Object` - **asset_definition**, definition of asset(json object).
  - `Integer` - **amount**, amount of asset.
  - `Object` - **issuance_program**, issuance program, it only exist when type is 'issue'.
  - `Object` - **control_program**, control program of account, it only exist when type is 'spend'.
  - `String` - **address**, address of account, it only exist when type is 'spend'.
  - `String` - **spent_output_id**, the front of outputID to be spent in this input, it only exist when type is 'spend'.
  - `String` - **account_id**, account id.
  - `String` - **account_alias**, name of account.
  - `Object` - **arbitrary**, arbitrary infomation can be set by miner, it only exist when type is 'coinbase'.

    - `String` - **input_id**, hash of input action.
    - `Array of String` - **witness_arguments**, witness arguments.
- `Array of Object` - **outputs**, object of outputs for the transaction.
    - `String` - **type**, the type of output action, available option include: 'retire', 'control'.
    - `String` - **id**, outputid related to utxo.
    - `Integer` - **position**, position of outputs.
    - `String` - **asset_id**, asset id.
    - `String` - **asset_alias**, name of asset.
    - `Object` - **asset_definition**, definition of asset(json object).
    - `Integer` - **amount**, amount of asset.
    - `String` - **account_id**, account id.
    - `String` - **account_alias**, name of account.
    - `Object` - **control_program**, control program of account.
    - `String` - **address**, address of account.

Example

Retrieve a transaction by it ID.

Request

```
curl -X POST http://localhost:9888/get-transaction -d '{"tx_id":
"15b8d66e227feff47b3de0f278934ea16d6c828371ec6c13c8f84713dd11703b"}'
```

Response

```
{
  "block_hash": "1fa9bb389cf974a9b37b63ca38c0cf3453c30f394b9e8ae7f04f2d1b52c329b4",
  "block_height": 530,
  "block_index": 1,
  "block_time": 1528772056,
  "block_transactions_count": 2,
  "inputs": [
    {
      "account_alias": "default",
      "account_id": "0ER7MEFGG0A02",
      "address": "sy1q4pkg8msjumtep7ecsdzuct3tsuzk5pmnm3p8nr",
      "amount": 41250000000,
      "asset_alias": "EY",
      "asset_definition": {
        "decimals": 8,
        "description": "Eiyaro Official Issue",
        "name": "EY",
        "symbol": "EY"
      },
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "0014a86c83ee12e6d790fb388345cc2e2b87056a0773",
      "input_id": "02702fe116e052aaf4473b034ed40720bfb3aba77df64625311ca3947d367336",
      "spent_output_id":
"002025b727148d04197cc7b9cf7eafd9986041f07641ca82dc0a1d9e227b52f6",
      "type": "spend",
      "witness_arguments": [

"944a35f256a49712f95319743671152b12360df859deedbfa9f37f9fe6a81b5ff2dce36d9ee6fc19e8be8
b1dd5915719d4341f66f5569aad26283859d3c1bc05",
        "bedfd27f48007c59555da672b6207ac997add62241894ff181bb9d8cba3b7e25"
      ]
    }
  ],
  "outputs": [
    {
      "account_alias": "default",
      "account_id": "0ER7MEFGG0A02",
      "address": "sy1qmt6jxrr8etssufr8qp98emyaly3lknxyndh5cj",
      "amount": 29450000000,
      "asset_alias": "EY",
      "asset_definition": {
        "decimals": 8,
        "description": "Eiyaro Official Issue",
        "name": "EY",
        "symbol": "EY"
      },
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "0014daf5230c67cae10e2467004a7cec9df923fb4cc4",
      "id": "35a46dd36eb27b1ffdfdefbe5366175b6325e8f56e5bc3dd2aa1a47197e26e6c",
      "position": 0,
      "type": "control"
    },
    {
      "account_alias": "alice",
      "account_id": "0ER7OAK400A02",
      "address": "sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me",
      "amount": 11700000000,
      "asset_alias": "EY",
      "asset_definition": {
        "decimals": 8,
        "description": "Eiyaro Official Issue",
        "name": "EY",
        "symbol": "EY"
      },
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "0014366b275ed9b2266b645cf1b8be51009cc3b260e1",
      "id": "ae791bbde0cc5b370e28a505933b85082d67be8db81bdcc56b8202f200b883e7",
      "position": 1,
      "type": "control"
    }
  ],
  "size": 332,
  "status_fail": false,
  "tx_id": "15b8d66e227feff47b3de0f278934ea16d6c828371ec6c13c8f84713dd11703b"
}
```

List Transactions Endpoint

Returns the sub list of all the account related transactions.

Parameters

`Object`:

Optional:

- `String` - **id**, transaction id, hash of transaction.
- `String` - **account_id**, id of account.
- `Boolean` - **detail** , flag of detail transactions, default false (only return transaction summary)
- `Boolean` - **unconfirmed**, flag of unconfirmed transactions(query result include all confirmed and unconfirmed transactions), default false.
- `Integer` - **from**, the start position of first transaction
- `Integer` - **count**, the number of returned

Returns

`Array of Object`, transaction array.

Optional:

- `Object`:(summary transaction)
  - `String` - **tx_id**, transaction id, hash of the transaction.
  - `Integer` - **block_time**, the unix timestamp for when the requst was responsed.
  - `Array of Object` - **inputs**, object of summary inputs for the transaction.
    - `String` - **type**, the type of input action, available option include: 'spend', 'issue', 'coinbase'.
    - `String` - **asset_id**, asset id.
    - `String` - **asset_alias**, name of asset.
    - `Integer` - **amount**, amount of asset.
    - `String` - **account_id**, account id.
    - `String` - **account_alias**, name of account.
    - `Object` - **arbitrary**, arbitrary infomation can be set by miner, it only exist when type is

        'coinbase'.
    - `Array of Object` - **outputs**, object of summary outputs for the transaction.
        - `String` - **type**, the type of output action, available option include: 'retire', 'control'.
        - `String` - **asset_id**, asset id.
        - `String` - **asset_alias**, name of asset.
        - `Integer` - **amount**, amount of asset.
        - `String` - **account_id**, account id.
        - `String` - **account_alias**, name of account.
        - `Object` - **arbitrary**, arbitrary infomation can be set by miner, it only exist when type is input 'coinbase'(this place is empty).
- `Object`:(detail transaction)
    - `String` - **tx_id**, transaction id, hash of the transaction.
    - `Integer` - **block_time**, the unix timestamp for when the requst was responsed.
    - `String` - **block_hash**, hash of the block where this transaction was in.
    - `Integer` - **block_height**, block height where this transaction was in.
    - `Integer` - **block_index**, position of the transaction in the block.
    - `Integer` - **block_transactions_count**, transactions count where this transaction was in the block.
    - `Boolean` - **status_fail**, whether the state of the transaction request has failed.
    - `Integer` - **size**, size of transaction.
    - `Array of Object` - **inputs**, object of inputs for the transaction.
        - `String` - **type**, the type of input action, available option include: 'spend', 'issue', 'coinbase'.
        - `String` - **asset_id**, asset id.
        - `String` - **asset_alias**, name of asset.
        - `Object` - **asset_definition**, definition of asset(json object).
        - `Integer` - **amount**, amount of asset.
        - `Object` - **issuance_program**, issuance program, it only exist when type is 'issue'.
        - `Object` - **control_program**, control program of account, it only exist when type is 'spend'.
        - `String` - **address**, address of account, it only exist when type is 'spend'.
        - `String` - **spent_output_id**, the front of outputID to be spent in this input, it only exist when type is 'spend'.
        - `String` - **account_id**, account id.
        - `String` - **account_alias**, name of account.
        - `Object` - **arbitrary**, arbitrary infomation can be set by miner, it only exist when type is 'coinbase'.
        - `String` - **input_id**, hash of input action.

        - `Array of String` - **witness_arguments**, witness arguments.
    - `Array of Object` - **outputs**, object of outputs for the transaction.
        - `String` - **type**, the type of output action, available option include: 'retire', 'control'.
        - `String` - **id**, outputid related to utxo.
        - `Integer` - **position**, position of outputs.
        - `String` - **asset_id**, asset id.
        - `String` - **asset_alias**, name of asset.
        - `Object` - **asset_definition**, definition of asset(json object).
        - `Integer` - **amount**, amount of asset.
        - `String` - **account_id**, account id.
        - `String` - **account_alias**, name of account.
        - `Object` - **control_program**, control program of account.
        - `String` - **address**, address of account.

Example

List all the available transactions:

Request

```
curl -X POST http://localhost:9888/list-transactions -d '{}'
```

Response

```
[
  {
    "block_time": 1521771059,
    "inputs": [
      {
        "arbitrary": "06",
        "asset_id":
"0000000000000000000000000000000000000000000000000000000000000000",
        "type": "coinbase"
      }
    ],
    "outputs": [
      {
        "account_alias": "default",
        "account_id": "0BMHBOBVG0A02",
        "amount": 41250000000,
        "asset_alias": "EY",
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "type": "control"
      }
    ],
    "tx_id": "c631a8de401913a512c338bcf4a61cb2de6cede12a7385d9d11637eaa6578f33"
  },
  {
    "block_time": 1521770515,
    "inputs": [
      {
        "account_alias": "default",
        "account_id": "0BMHBOBVG0A02",
        "amount": 41250000000,
        "asset_alias": "EY",
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "type": "spend"
      }
    ],
    "outputs": [
      {
        "account_alias": "default",
        "account_id": "0BMHBOBVG0A02",
        "amount": 34649500000,
        "asset_alias": "EY",
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "type": "control"
      },
      {
        "account_alias": "alice",
        "account_id": "0BMHDI1P00A04",
        "amount": 6600000000,
        "asset_alias": "EY",
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "type": "control"
      }
    ],
    "tx_id": "1151ce5c7b32b8755b5e48109ec7ed956fb1783eaea9558bf5a2ad957825e4b7"
  }
]
```

List the transaction matching the given `tx_id` with detail:

Request

```
curl -X POST http://localhost:9888/list-transactions -d '{"id":
"7e9f9b999381da936e3cae48b5bac2b9bc28bb56c6c862be6c110448f7e2f6b3","detail": true}'
```

Response

```
[
  {
    "block_hash": "1b2d0efa025256603e9330273d37f5a900cd3dfb213e015ac53cf645e2315a6d",
    "block_height": 72,
    "block_index": 1,
    "block_time": 1528528584,
    "block_transactions_count": 2,
    "inputs": [
      {
        "account_alias": "default",
        "account_id": "0ER7MEFGG0A02",
        "address": "sy1q4pkg8msjumtep7ecsdzuct3tsuzk5pmnm3p8nr",
        "amount": 41250000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014a86c83ee12e6d790fb388345cc2e2b87056a0773",
        "input_id":
"adcef046c3f61fb6ba0d6a7107122f6e31cd4b49c7a3b05aa3391e5b0529d69a",
        "spent_output_id":
"0072a2c1cee30a7c7b7b006ca08a48c2b479bc81c0ec6463fe4865ef37626ab6",
        "type": "spend",
        "witness_arguments": [

"944a35f256a49712f95319743671152b12360df859deedbfa9f37f9fe6a81b5ff2dce36d9ee6fc19e8be8
b1dd5915719d4341f66f5569aad26283859d3c1bc05",
          "bedfd27f48007c59555da672b6207ac997add62241894ff181bb9d8cba3b7e25"
        ]
      }
    ],
    "outputs": [
      {
        "account_alias": "default",
        "account_id": "0ER7MEFGG0A02",
        "address": "sy1qskj096x5w7ejcmk746g3djmv84dpxts62dewvd",
        "amount": 34649500000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "001485a4f2e8d477b32c6edeae9116cb6c3d5a132e1a",
        "id": "b08c9bfc816064ca33da8b569998229774fc9552da7d4f16870b2c5a8f645b3b",
        "position": 0,
        "type": "control"
      },
      {
        "account_alias": "alice",
        "account_id": "0ER7OAK400A02",
        "address": "sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me",
        "amount": 6600000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014366b275ed9b2266b645cf1b8be51009cc3b260e1",
        "id": "0e8f8dc83a39b2b6d00a77759a797102d047f82f800fe21f5b1d80bb4d5e2e39",
        "position": 1,
        "type": "control"
      }
    ],
    "size": 333,
    "status_fail": false,
    "tx_id": "7e9f9b999381da936e3cae48b5bac2b9bc28bb56c6c862be6c110448f7e2f6b3"
  }
]
```

List the transaction matching the given `account_id` and unconfirmed flag(unconfirmed transaction's `block_hash`, `block_height` and `block_index` is default for zero):

Request

```
curl -X POST http://localhost:9888/list-transactions -d '{"account_id":
"0F1MQVI500A02", "unconfirmed": true, "detail": true}'
```

Response

```
[
  {
    "block_hash": "0000000000000000000000000000000000000000000000000000000000000000",
    "block_height": 0,
    "block_index": 0,
    "block_time": 1529032899,
    "inputs": [
      {
        "account_alias": "default",
        "account_id": "0F1L5Q3V00A02",
        "address": "sy1ql67n04pj8mfqzv3wjq8num3yrltdykemgrr45j",
        "amount": 41250000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014febd37d4323ed201322e900f3e6e241fd6d25b3b",
        "input_id":
"192ac93bad580cd53626b7f11c17e6eca64f66d1947add13a5620b78f666693e",
        "spent_output_id":
"00570443cbac4f68638ff565e8b04db2062800b9e23b7701913ddf6b190dbe65",
        "type": "spend",
        "witness_arguments": [

"512a2b60324433de96cd4274bd298b4b109a29c4d9d68582952065dfd0d7c00663cbc49e8e42fdef740a7
e1b78622ee31abf2e9b0d5609755f275afd6751590b",
          "bedfd27f48007c59555da672b6207ac997add62241894ff181bb9d8cba3b7e25"
        ]
      },
      {
        "account_alias": "default",
        "account_id": "0F1L5Q3V00A02",
        "address": "sy1ql67n04pj8mfqzv3wjq8num3yrltdykemgrr45j",
        "amount": 41250000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014febd37d4323ed201322e900f3e6e241fd6d25b3b",
        "input_id":
"83713c02b52eb18f782de67b322c43571d83793e082596b6410e2d3a8a41387d",
        "spent_output_id":
"01df9011ca0bed4bb9b95dc84da4c5103fed06ca28c03d92d34ee3d61b945288",
        "type": "spend",
        "witness_arguments": [
"512a2b60324433de96cd4274bd298b4b109a29c4d9d68582952065dfd0d7c00663cbc49e8e42fdef740a7
e1b78622ee31abf2e9b0d5609755f275afd6751590b",
          "bedfd27f48007c59555da672b6207ac997add62241894ff181bb9d8cba3b7e25"
        ]
      }
    ],
    "outputs": [
      {
        "account_alias": "default",
        "account_id": "0F1L5Q3V00A02",
        "address": "sy1qdcfprk7wjy6flavkzhcjh3dxyrwlm935trrs5m",
        "amount": 41249100000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "00146e1211dbce91349ff59615f12bc5a620ddfd9634",
        "id": "09fabb1a2bac44c45054175453e23e81a764557147523d8df70d8a190cf2eb17",
        "position": 0,
        "type": "control"
      },
      {
        "account_alias": "default",
        "account_id": "0F1L5Q3V00A02",
        "address": "sy1qt92xx2f4ys63dyhy58jle87nttcf37zftweklh",
        "amount": 39150000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014595463293524351692e4a1e5fc9fd35af098f849",
        "id": "6efae48663e872193e8a672eb85b8bbf29d8aee98e42816340fa0b2340cc355d",
        "position": 1,
        "type": "control"
      },
      {
        "account_alias": "alice",
        "account_id": "0F1MQVI500A02",
        "address": "sy1qum6ly8aq9u9k7xrkuck9pq64xg67gw40khnnxu",
        "amount": 2100000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014e6f5f21fa02f0b6f1876e62c5083553235e43aaf",
        "id": "aca1ecc59d8bcf548e4f5afb8a97e38f0eb56e1387b17400fd3c693c074a703d",
        "position": 2,
        "type": "control"
      }
    ],
    "size": 1194,
    "status_fail": false,
    "tx_id": "9c28a6a2a039ed5bdebe81eea44cdb22a951c472bc25cb1e8188ae423a98f251"
  },
  {
    "block_hash": "474b9c28b225fba02278ad3b097d561bf8f5c562ff2a548226fc10fc1d75b7ed",
    "block_height": 255,
    "block_index": 1,
    "block_time": 1528963126,
    "block_transactions_count": 2,
    "inputs": [
      {
        "account_alias": "alice",
        "account_id": "0F1MQVI500A02",
        "address": "sy1qum6ly8aq9u9k7xrkuck9pq64xg67gw40khnnxu",
        "amount": 10000000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014e6f5f21fa02f0b6f1876e62c5083553235e43aaf",
        "input_id":
"0705bb7f4aea4ef869f22ab5e4a26e051b066e41290c1b74734a82aee8c03dfc",
        "spent_output_id":
"767649aafdfe2c22d46d641a5b74d934e2590330f7280b0fc55b978812a99a58",
        "type": "spend",
        "witness_arguments": [

"a4d4f09a04371516d37e1d27f92c9cb41e4b1e7f62762cf23ed3904a9dfd2d794195862fffd00bf7ac373
e5891c8d2eb660dc5ff9c040ec4e01f973bbfd31c23",
          "2ecb3f55bfde18ec95a93c456dd3d44cb55da83148a68cbc059ea04e7b12d3bc"
        ]
      },
      {
        "account_alias": "alice",
        "account_id": "0F1MQVI500A02",
        "address": "sy1qum6ly8aq9u9k7xrkuck9pq64xg67gw40khnnxu",
        "amount": 1000000000000,
        "asset_alias": "GOLD",
        "asset_definition": {
          "decimals": 8,
          "description": {},
          "name": "",
          "symbol": ""
        },
        "asset_id":
"71deb74415f16a1f7bffb04c61d427bb1f93adfba257ffba2673f102d602e78f",
        "control_program": "0014e6f5f21fa02f0b6f1876e62c5083553235e43aaf",
        "input_id":
"35764d80217d0d2a3c1b000dc2dd47cf0c8bc152c842ce6e3a7783140087d3d6",
        "spent_output_id":
"5d7a88851f5696ded279cb9bc380e050024c555258ea7851dfdedc2797b0d820",
        "type": "spend",
        "witness_arguments": [

"a4d4f09a04371516d37e1d27f92c9cb41e4b1e7f62762cf23ed3904a9dfd2d794195862fffd00bf7ac373
e5891c8d2eb660dc5ff9c040ec4e01f973bbfd31c23",
          "2ecb3f55bfde18ec95a93c456dd3d44cb55da83148a68cbc059ea04e7b12d3bc"
        ]
      }
    ],
    "outputs": [
      {
        "account_alias": "alice",
        "account_id": "0F1MQVI500A02",
        "address": "sy1q39sztlh4jq5nknstn2udvvpm6v5ugussx2djc0",
        "amount": 9980000000,
        "asset_alias": "EY",
        "asset_definition": {
          "decimals": 8,
          "description": "Eiyaro Official Issue",
          "name": "EY",
          "symbol": "EY"
        },
        "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
        "control_program": "0014896025fef590293b4e0b9ab8d6303bd329c47210",
        "id": "2b44969d28d79544006e792411d6cd1d245f9af20419f6138494b4b5aac2a72e",
        "position": 0,
        "type": "control"
      },
      {
        "account_alias": "alice",
        "account_id": "0F1MQVI500A02",
        "address": "sy1q258yd0gvatje4pn0qc8z9w8cdv45j9tvhfpjh8",
        "amount": 999999999901,
        "asset_alias": "GOLD",
        "asset_definition": {
          "decimals": 8,
          "description": {},
          "name": "",
          "symbol": ""
        },
        "asset_id":
"71deb74415f16a1f7bffb04c61d427bb1f93adfba257ffba2673f102d602e78f",
        "control_program": "0014550e46bd0ceae59a866f060e22b8f86b2b49156c",
        "id": "54be1bc876d1deccb9845acec79eabf62d7eacd5935e337850233657914d0f9d",
        "position": 1,
        "type": "control"
      },
      {
        "amount": 99,
        "asset_alias": "GOLD",
        "asset_definition": {
          "decimals": 8,
          "description": {},
          "name": "",
          "symbol": ""
        },
        "asset_id":
"71deb74415f16a1f7bffb04c61d427bb1f93adfba257ffba2673f102d602e78f",
        "control_program":
"20e864761d8181103b6476435a805cba97361df9a05c40fae644c27f69ce045d3c16001464d928e181900
d382fa33def66534c7323c778c4015820684d6683d014abb4e019878b50fbbb547bcbf9c4739498d8eeef5
65d37f9a82f741a547a6413000000007b7b51547ac1631a000000547a547aae7cac00c0",
        "id": "347553923bb550c236a703e46600d53f25161e3eb74ee3183884d398e5d894b0",
        "position": 2,
        "type": "control"
      }
    ],
    "size": 691,
    "status_fail": false,
    "tx_id": "383f8636842301b2fe292c5b8b2f540c6ed7867ba5751680b2e77827c300e41e"
  }
]
```

Build Transaction Endpoint

Build transaction.

Parameters

`Object`:

- `String` - **base_transaction**, base data for the transaction, default is null.
- `Integer` - **ttl**, integer of the time to live in milliseconds, it means utxo will be reserved(locked) for builded transaction in this time range, if the transaction will not to be submitted into block, it will be auto unlocked for build transaction again after this ttl time. it will be set to 5 minutes(300 seconds) defaultly when ttl is 0.
- `Integer` - **time_range**, the block height at which this transaction will be allowed to be included in a block. If the block height of the main chain exceeds this value, the transaction will expire and no longer be valid.
- `Arrary of Object` - **actions**:
  - `Object`:
    - `String` - **account_id** | **account_alias**, (type is spend_account) alias or ID of account.
    - `String` - **asset_id** | **asset_alias**, (type is spend_account, issue, retire, control_program and control_address) alias or ID of asset.
    - `Integer` - **amount**, (type is spend_account, issue, retire, control_program and control_address) the specified asset of the amount sent with this transaction.
    - `String`- **type**, type of transaction, valid types: 'spend_account', 'issue', 'spend_account_unspent_output', 'control_address', 'control_program', 'retire'.
    - `String` - **address**, (type is control_address) address of receiver, the style of address is P2PKH or P2SH.
    - `String` - **control_program**, (type is control_program) control program of receiver.
    - `String` - **use_unconfirmed**, (type is spend_account and spend_account_unspent_output) flag of use unconfirmed UTXO, default is false.
    - `String` - **arbitrary**, (type is retire) arbitrary additional data by hexadecimal.
    - `Arrary of Object` - **arguments**, (type is issue and spend_account_unspent_output) arguments of contract, null when it's not contract.
      - `String`- **type**, type of argument, valid types: 'raw_tx_signature', 'data'.
      - `Object`- **raw_data**, json object of argument content.
        - `String`- **xpub**, (type is raw_tx_signature) root xpub.
        - `String`- **derivation_path**, (type is raw_tx_signature) derived path.
        - `String`- **value**, (type is data) string of binary value.

Returns

- `Object of build-transaction` - **transaction**, built transaction.

Example

Build transaction of type `spend`.

Request

```
curl -X POST http://localhost:9888/build-transaction -d
'{"base_transaction":null,"actions":[{"account_id":"0BF63M2U00A04","amount":20000000,"
asset_id":"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff","type":"s
pend_account"},{"account_id":"0BF63M2U00A04","amount":99,"asset_id":"3152a15da72be51b3
30e1c0f8e1c0db669269809da4f16443ff266e07cc43680","type":"spend_account"},{"amount":99,
"asset_id":"3152a15da72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc43680","address
":"bm1q50u3z8empm5ke0g3ngl2t3sqtr6sd7cepd3z68","type":"control_address"}],"ttl":0,"tim
e_range": 43432}'
```

Build transaction of type `issue`.

Request

```
curl -X POST http://localhost:9888/build-transaction -d
'{"base_transaction":null,"actions":[{"account_id":"0BF63M2U00A04","amount":20000000,"
asset_id":"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff","type":"s
pend_account"},{"amount":10000,"asset_id":"3152a15da72be51b330e1c0f8e1c0db669269809da4
f16443ff266e07cc43680","type":"issue"},{"amount":10000,"asset_id":"3152a15da72be51b330
e1c0f8e1c0db669269809da4f16443ff266e07cc43680","address":"ey1q50u3z8empm5ke0g3ngl2t3sq
tr6sd7cepd3z68","type":"control_address"}],"ttl":0,"time_range": 43432}'
```

Build transaction of type `address`.

Request

```
curl -X POST http://localhost:9888/build-transaction -d
'{"base_transaction":null,"actions":[{"account_id":"0BF63M2U00A04","amount":20000000,"
asset_id":"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff","type":"s
pend_account"},{"account_id":"0BF63M2U00A04","amount":99,"asset_id":"3152a15da72be51b3
30e1c0f8e1c0db669269809da4f16443ff266e07cc43680","type":"spend_account"},{"amount":99,
"asset_id":"3152a15da72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc43680","address
":"ey1q50u3z8empm5ke0g3ngl2t3sqtr6sd7cepd3z68","type":"control_address"}],"ttl":0,"tim
e_range": 43432}'
```

Build transaction of type `retire`.

Request

```
curl -X POST http://localhost:9888/build-transaction -d
'{"base_transaction":null,"actions":[{"account_id":"0BF63M2U00A04","amount":20000000,"
asset_id":"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff","type":"s
pend_account"},{"account_id":"0BF63M2U00A04","amount":99,"asset_id":"3152a15da72be51b3
30e1c0f8e1c0db669269809da4f16443ff266e07cc43680","type":"spend_account"},{"amount":99,
"asset_id":"3152a15da72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc43680","arbitra
ry":"77656c636f6d65efbc8ce6aca2e8bf8ee69da5e588b0e58e9fe5ad90e4b896e7958c","type":"ret
ire"}],"ttl":0,"time_range":43432}'
```

Build transaction of type `spend_account_unspent_output`(user can get `UTXO` information by calling the `list-unspent-outputs` endpoint).

- action field `output_id` correspond to `UTXO` result `id` field
- UTXO asset and amount will be spent in this transaction
- transaction fee is (utxo `asset_amount` - output `asset_amount`)

Request

```
curl -X POST http://localhost:9888/build-transaction -d
'{"base_transaction":null,"actions":[{"type":"spend_account_unspent_output","output_id
":"01c6ccc6f522228cd4518bba87e9c43fbf55fdf7eb17f5aa300a037db7dca0cb"},{"amount":412430
00000,"asset_id":"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff","a
ddress":"sy1qmw8c5s29zlexknfahrze3ghvlqrtn2huuntvpn","type":"control_address"}],"ttl":
0,"time_range":0}'
```

Response (this type is `spend`, the other types are similar)

```
{
  "allow_additional_actions": false,
  "local": true,
  "raw_transaction":
"07010000020161015fb6a63a3361170afca03c9d5ce1f09fe510187d69545e09f95548b939cd7fffa3fff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80fc93afdf01000116001426b
d1b851cf6eb8a701c20c184352ad8720eeee90100015d015bb6a63a3361170afca03c9d5ce1f09fe510187
d69545e09f95548b939cd7fffa33152a15da72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc
43680c03e0101160014489a678741ccc844f9e5c502f7fac0a665bedb25010003013efffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffff80a2cfa5df0101160014948fb4f500e66d20fba
cb903fe108ee81f9b6d9500013a3152a15da72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc
43680dd3d01160014cd5a822b34e3084413506076040d508bb12232c70001393152a15da72be51b330e1c0
f8e1c0db669269809da4f16443ff266e07cc436806301160014a3f9111f3b0ee96cbd119a3ea5c60058f50
6fb1900",
  "signing_instructions": [
    {
      "position": 0,
      "witness_components": [
        {
          "keys": [
            {
              "derivation_path": [
                "010100000000000000",
                "0500000000000000"
              ],
              "xpub":
"ee9dd8affdef7e0cacd0fbbf310217c7f588156c28e414db74c27afaedd8f876cf54547a672b431ff06ee
8a146207df9595638a041b55ada1a764a8b5b30bda0"
            }
          ],
          "quorum": 1,
          "signatures": null,
          "type": "raw_tx_signature"
        },
        {
          "type": "data",
          "value": "62a73b6b7ffe52b6ad782b0e0efdc8309bf2f057d88f9a17d125e41bb11dbb88"
        }
      ]
    },
    {
      "position": 1,
      "witness_components": [
        {
          "keys": [
            {
              "derivation_path": [
                "010100000000000000",
                "0600000000000000"
              ],
              "xpub":
"ee9dd8affdef7e0cacd0fbbf310217c7f588156c28e414db74c27afaedd8f876cf54547a672b431ff06ee
8a146207df9595638a041b55ada1a764a8b5b30bda0"
            }
          ],
          "quorum": 1,
          "signatures": null,
          "type": "raw_tx_signature"
        },
        {
          "type": "data",
          "value": "ba5a63e7416caeb945eefc2ce874f40bc4aaf6005a1fc792557e41046f7e502f"
        }
      ]
    }
  ]
}
```

Build Chain Transactions Endpoint

Build chain transactions. To solve the problem of excessive `utxo` causing the transaction to fail, the `utxo` merge will be performed automatically. Currently, only `EY` transactions are supported.

This feature requires the `core` software to be higher than `v1.0.1`.

Parameters

`Object`:

- `String` - **base_transaction**, base data for the transaction, default is null.
- `Integer` - **ttl**, integer of the time to live in milliseconds, it means utxo will be reserved(locked) for builded transaction in this time range, if the transaction will not to be submitted into block, it will be auto unlocked for build transaction again after this ttl time. it will be set to 5 minutes(300 seconds) defaultly when ttl is 0.
- `Integer` - **time_range**, time stamp(block height)is maximum survival time for the transaction, the transaction will be not submit into block after this time stamp.
- `Arrary of Object` - **actions**:
    - `Object`:
        - `String` - **account_id** | **account_alias**, (type is spend_account) alias or ID of account.
        - `String` - **asset_id** | **asset_alias**, (type is spend_account, issue, retire, control_program and control_address) alias or ID of asset.
        - `Integer` - **amount**, (type is spend_account, issue, retire, control_program and control_address) the specified asset of the amount sent with this transaction.
        - `String`- **type**, type of transaction, valid types: 'spend_account', 'issue', 'spend_account_unspent_output', 'control_address', 'control_program', 'retire'.
        - `String` - **address**, (type is control_address) address of receiver, the style of address is P2PKH or P2SH.
        - `String` - **control_program**, (type is control_program) control program of receiver.
        - `String` - **use_unconfirmed**, (type is spend_account and spend_account_unspent_output) flag of use unconfirmed UTXO, default is false.
        - `Arrary of Object` - **arguments**, (type is issue and spend_account_unspent_output) arguments of contract, null when it’s not contract.
            - `String`- **type**, type of argument, valid types: 'raw_tx_signature', 'data'.
            - `Object`- **raw_data**, json object of argument content.
                - `String`- **xpub**, (type is raw_tx_signature) root xpub.
                - `String`- **derivation_path**, (type is raw_tx_signature) derived path.
                - `String`- **value**, (type is data) string of binary value.

Returns

- `Object of raw_transaction` - **raw_transaction**, builded transactions.
- `Object of signing_instructions` - **signing_instructions**, Information used to sign a transactions.

Example

Build chain transaction of type `spend`.

Request

```
curl -X POST http://localhost:9888//build-chain-transactions -d '{"base_transaction":
null,"actions":[{"account_id":"0JCH28A600A02","amount":30000500000000,"asset_id":"ffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff","type":
"spend_account"}, {"amount": 30000490000000,"asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff","address":
"sm1qx93sge8jkgzclc7pled7uqr596hjm2xe558lkr","type": "control_address"}],"ttl":
1000000,"time_range": 0}'
```

Response

```
{
    "status": "success",
    "data": [{
        "raw_transaction":
"0701000201620160a0d36052ca3d1335120ae48e1ffb2fb6b25588628eff90fa88bef3117dfb4301fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffff80ddb2c490e9060101160014316
30464f2b2058fe3c1fe5bee00742eaf2da8d901000161015f72de2064ab999acf22c05b5cf9c7d53164f80
038b46b1ce426708514a30a3485fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffff80d4f4f69901000116001431630464f2b2058fe3c1fe5bee00742eaf2da8d9010001013fffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffff8084c5b6aaea0601160014316304 64f
2b2058fe3c1fe5bee00742eaf2da8d900",
        "signing_instructions": [{
            "position": 0,
            "witness_components": [{
                "type": "raw_tx_signature",
                "quorum": 1,
                "keys": [{
                    "xpub":
"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f3514b461
f6931ac2073f586a35cd628c90270063725e6e1e983",
                    "derivation_path": ["010100000000000000", "0100000000000000"]
                }],
                "signatures": null
            }, {
                "type": "data",
                "value":
"a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a8973919"
            }]
        }, {
            "position": 1,
            "witness_components": [{
                "type": "raw_tx_signature",
                "quorum": 1,
                "keys": [{
                    "xpub":
"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f3514b461
f6931ac2073f586a35cd628c90270063725e6e1e983",
                    "derivation_path": ["010100000000000000", "0100000000000000"]
                }],
                "signatures": null
            }, {
                "type": "data",
                "value":
"a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a8973919"
            }]
        }],
        "allow_additional_actions": false
    }, {
        "raw_transaction":
"0701000101620160571cc5d99a2994ff6b192bc9387838a3651245cb66dad4a6bc5f660310cebfa9fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8084c5b6aaea060001160014316
30464f2b2058fe3c1fe5bee00742eaf2da8d9010002013efffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffff80faafed99010116001431630464f2b2058fe3c1fe5bee00742eaf2da8d90
0013fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80ddb2c490e906011
6001431630464f2b2058fe3c1fe5bee00742eaf2da8d900",
        "signing_instructions": [{
            "position": 0,
            "witness_components": [{
                "type": "raw_tx_signature",
                "quorum": 1,
                "keys": [{
                    "xpub":
"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f3514b461
f6931ac2073f586a35cd628c90270063725e6e1e983",
                    "derivation_path": ["010100000000000000", "0100000000000000"]
                }],
                "signatures": null
            }, {
                "type": "data",
                "value":
"a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a8973919"
            }]
        }],
        "allow_additional_actions": false
    }]
}
```

Sign Transaction Endpoint

Sign a transaction.

Parameters

`Object`:

- `String` - **password**, signature of the password.
- `Object` - **transaction**, builded transaction.

Returns

`Object`:

- `Boolean` - **sign_complete**, returns true if sign succesfully and false otherwise.
- `Object of sign-transaction` - **transaction**, signed transaction.

Example

Perform the signature of a transaction.

Request

```
curl -X POST http://localhost:9888/sign-transaction -d
'{"password":"123456","transaction":{"allow_additional_actions":false,"local":true,"ra
w_transaction":"0701000020161015fb6a63a3361170afca03c9d5ce1f09fe510187d69545e09f95548
b939cd7fffa3ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80fc93afdf
01000116001426bd1b851cf6eb8a701c20c184352ad8720eeee90100015d015bb6a63a3361170afca03c9d
5ce1f09fe510187d69545e09f95548b939cd7fffa33152a15da72be51b330e1c0f8e1c0db669269809da4f
16443ff266e07cc43680c03e0101160014489a678741ccc844f9e5c502f7fac0a665bedb25010003013eff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80a2cfa5df0101160014948f
b4f500e66d20fbacb903fe108ee81f9b6d9500013a3152a15da72be51b330e1c0f8e1c0db669269809da4f
16443ff266e07cc43680dd3d01160014cd5a822b34e3084413506076040d508bb12232c70001393152a15d
a72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc436806301160014a3f9111f3b0ee96cbd11
9a3ea5c60058f506fb1900","signing_instructions":[{"position":0,"witness_components":[{"
keys":[{"derivation_path":["010100000000000000","0500000000000000"],"xpub":"ee9dd8affd
ef7e0cacd0fbbf310217c7f588156c28e414db74c27afaedd8f876cf54547a672b431ff06ee8a146207df9
595638a041b55ada1a764a8b5b30bda0"}],"quorum":1,"signatures":null,"type":"raw_tx_signat
ure"},{"type":"data","value":"62a73b6b7ffe52b6ad782b0e0efdc8309bf2f057d88f9a17d125e41b
b11dbb88"}]},{"position":1,"witness_components":[{"keys":[{"derivation_path":["0101000
00000000000","0600000000000000"],"xpub":"ee9dd8affdef7e0cacd0fbbf310217c7f588156c28e41
4db74c27afaedd8f876cf54547a672b431ff06ee8a146207df9595638a041b55ada1a764a8b5b30bda0"}]
,"quorum":1,"signatures":null,"type":"raw_tx_signature"},{"type":"data","value":"ba5a6
3e7416caeb945eefc2ce874f40bc4aaf6005a1fc792557e41046f7e502f"}]}]}}'
```

Response

```
{
  "sign_complete": true,
  "transaction": {
    "allow_additional_actions": false,
    "local": true,
    "raw_transaction":
"07010000020161015fb6a63a3361170afca03c9d5ce1f09fe510187d69545e09f95548b939cd7fffa3fff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80fc93afdf01000116001426b
d1b851cf6eb8a701c20c184352ad8720eeee96302400d432e6f0e22da3168d76552273e60d23d432d61b4d
ac53e6769d39a1097f1cd1bd8e54c7d39eda334803e5c904bc2de2f27ff29748166e0334dcfded20e980b2
062a73b6b7ffe52b6ad782b0e0efdc8309bf2f057d88f9a17d125e41bb11dbb88015d015bb6a63a3361170
afca03c9d5ce1f09fe510187d69545e09f95548b939cd7fffa33152a15da72be51b330e1c0f8e1c0db6692
69809da4f16443ff266e07cc43680c03e0101160014489a678741ccc844f9e5c502f7fac0a665bedb25630
2401eadd84ad07c3643f71a35cc5669a2c1def96ae98e790d287217e6a3543fe602dd90afffe853c729bd5
237a28f33538df631572847d9870829fb1fd1100ff20820ba5a63e7416caeb945eefc2ce874f40bc4aaf60
05a1fc792557e41046f7e502f03013effffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffff80a2cfa5df0101160014948fb4f500e66d20fbacb903fe108ee81f9b6d9500013a3152a15da72
be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc43680dd3d01160014cd5a822b34e3084413506
076040d508bb12232c70001393152a15da72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc43
6806301160014a3f9111f3b0ee96cbd119a3ea5c60058f506fb1900",
    "signing_instructions": [
      {
        "position": 0,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0500000000000000"
                ],
                "xpub":
"ee9dd8affdef7e0cacd0fbbf310217c7f588156c28e414db74c27afaedd8f876cf54547a672b431ff06ee
8a146207df9595638a041b55ada1a764a8b5b30bda0"
              }
            ],
            "quorum": 1,
            "signatures": [

"0d432e6f0e22da3168d76552273e60d23d432d61b4dac53e6769d39a1097f1cd1bd8e54c7d39eda334803
e5c904bc2de2f27ff29748166e0334dcfded20e980b"
            ],
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"62a73b6b7ffe52b6ad782b0e0efdc8309bf2f057d88f9a17d125e41bb11dbb88"
          }
        ]
      },
      {
        "position": 1,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0600000000000000"
                ],
                "xpub":
"ee9dd8affdef7e0cacd0fbbf310217c7f588156c28e414db74c27afaedd8f876cf54547a672b431ff06ee
8a146207df9595638a041b55ada1a764a8b5b30bda0"
              }
            ],
            "quorum": 1,
            "signatures": [

"1eadd84ad07c3643f71a35cc5669a2c1def96ae98e790d287217e6a3543fe602dd90afffe853c729bd523
7a28f33538df631572847d9870829fb1fd1100ff208"
            ],
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"ba5a63e7416caeb945eefc2ce874f40bc4aaf6005a1fc792557e41046f7e502f"
          }
        ]
      }
    ]
  }
}
```

Sign Transactions Endpoint

Sign transactions used for batch signing transactions.

Parameters

`Object`:

- `String` - **password**, signature of the password.
- `Object` - **transaction**, builded transactions.

Returns

`Object`:

- `Boolean` - **sign_complete**, returns true if sign succesfully and false otherwise.
- `Object of sign-transactions` - **transaction**, signed transactions.

Example

Perform the signature of a batch of transactions.

Request

```
curl -X POST http://localhost:9888/sign-transactions -d
'{"password":"123456","transactions":[{"raw_transaction":"0701000201620160a0d36052ca3d
1335120ae48e1ffb2fb6b25588628eff90fa88bef3117dfb4301ffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffff80ddb2c490e906010116001431630464f2b2058fe3c1fe5bee00742e
af2da8d901000161015f72de2064ab999acf22c05b5cf9c7d53164f80038b46b1ce426708514a30a3485ff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80d4f4f69901000116001431
630464f2b2058fe3c1fe5bee00742eaf2da8d9010001013fffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffff8084c5b6aaea060116001431630464f2b2058fe3c1fe5bee00742eaf2da8
d900","signing_instructions":[{"position":0,"witness_components":[{"type":"raw_tx_sign
ature","quorum":1,"keys":[{"xpub":"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc
6290dd14176f92523f229d9f1f3514b461f6931ac2073f586a35cd628c90270063725e6e1e983","deriva
tion_path":["010100000000000000","0100000000000000"]}],"signatures":null},{"type":"dat
a","value":"a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a8973919"}]},{"pos
ition":1,"witness_components":[{"type":"raw_tx_signature","quorum":1,"keys":[{"xpub":"
b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f3514b461f
6931ac2073f586a35cd628c90270063725e6e1e983","derivation_path":["010100000000000000","0
100000000000000"]}],"signatures":null},{"type":"data","value":"a86ab33efa9d71994270898
ad99f198d60889ef617d5eaf25e776929a8973919"}]}],"allow_additional_actions":false},{"raw
_transaction":"0701000101620160571cc5d99a2994ff6b192bc9387838a3651245cb66dad4a6bc5f660
310cebfa9ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8084c5b6aaea0
6000116001431630464f2b2058fe3c1fe5bee00742eaf2da8d9010002013effffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffff80faafed990101160014316304 64f2b2058fe3c1fe5bee0
0742eaf2da8d900013fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80d
db2c490e906011600143163 0464f2b2058fe3c1fe5bee00742eaf2da8d900","signing_instructions":
[{"position":0,"witness_components":[{"type":"raw_tx_signature","quorum":1,"keys":[{"x
pub":"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f351
4b461f6931ac2073f586a35cd628c90270063725e6e1e983","derivation_path":["0101000000000000
00","0100000000000000"]}],"signatures":null},{"type":"data","value":"a86ab33efa9d71994
270898ad99f198d60889ef617d5eaf25e776929a8973919"}]}],"allow_additional_actions":false}
]}'
```

Response

```
{
    "status": "success",
    "data": {
        "transaction": [{
                "raw_transaction":
"0701000201620160a0d36052ca3d1335120ae48e1ffb2fb6b25588628eff90fa88bef3117dfb4301fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80ddb2c490e9060101160014316
30464f2b2058fe3c1fe5bee00742eaf2da8d9630240acb57bc06f7e5de99ef3e630ce34fc74c33d4694301
202968092ca50ae7842e3331bfeb0cf7b65f383e27670c4d58aeeeb0b77e5355957ca729298d2b4e2470c2
0a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a89739190161015f72de2064ab999
```

```
acf22c05b5cf9c7d53164f80038b46b1ce426708514a30a3485fffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffff80d4f4f69901000116001431630464f2b2058fe3c1fe5bee00742eaf2
da8d96302404298424e89e5528f1d0cdd9028489b9d9e3f031ec34a74440cacc7900dc1eac9359c408a434
2fc6cef935d2978919df8b23f3912ac4419800d375fac06ddb50620a86ab33efa9d71994270898ad99f198
d60889ef617d5eaf25e776929a897391901013ffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffff8084c5b6aaea060116001431630464f2b2058fe3c1fe5bee00742eaf2da8d900",
                "signing_instructions": [{
                        "position": 0,
                        "witness_components": [{
                                "type": "raw_tx_signature",
                                "quorum": 1,
                                "keys": [{
                                    "xpub":
"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f3514b461
f6931ac2073f586a35cd628c90270063725e6e1e983",
                                    "derivation_path": [
                                        "010100000000000000",
                                        "0100000000000000"
                                    ]
                                }],
                                "signatures": [

"acb57bc06f7e5de99ef3e630ce34fc74c33d46943012029680 92ca50ae7842e3331bfeb0cf7b65f383e27
670c4d58aeeeb0b77e5355957ca729298d2b4e2470c"
                                ]
                            },
                            {
                                "type": "data",
                                "value":
"a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a8973919"
                            }
                        ]
                    },
                    {
                        "position": 1,
                        "witness_components": [{
                                "type": "raw_tx_signature",
                                "quorum": 1,
                                "keys": [{
                                    "xpub":
"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f3514b461
f6931ac2073f586a35cd628c90270063725e6e1e983",
                                    "derivation_path": [
                                        "010100000000000000",
                                        "0100000000000000"
                                    ]
                                }],
                                "signatures": [

"4298424e89e5528f1d0cdd9028489b9d9e3f031ec34a74440cacc7900dc1eac9359c408a4342fc6cef935
d2978919df8b23f3912ac4419800d375fac06ddb506"
```

```
                                ]
                            },
                            {
                                "type": "data",
                                "value":
"a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a8973919"
                            }
                        ]
                    }
                ],
                "allow_additional_actions": false
            },
            {
                "raw_transaction":
"0701000101620160571cc5d99a2994ff6b192bc9387838a3651245cb66dad4a6bc5f660310cebfa9fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8084c5b6aaea060001160014316
30464f2b2058fe3c1fe5bee00742eaf2da8d96302408c742d77eba6c56a8db8c114e60be6c6263df6120ae
fd7538376129d04ec71b78b718c2085bba85254b44bf4600ba31d4c5a7869d0be0c46d88bd5eb27490e082
0a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a897391902013efffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffff80faafed9901011600143163 0464f2b2058fe
3c1fe5bee00742eaf2da8d900013ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffff80ddb2c490e9060116001431630464f2b2058fe3c1fe5bee00742eaf2da8d900",
                "signing_instructions": [{
                    "position": 0,
                    "witness_components": [{
                            "type": "raw_tx_signature",
                            "quorum": 1,
                            "keys": [{
                                "xpub":
"b4d084e77bcda7fd8a37e31135200b2a6af98d19018674125dc6290dd14176f92523f229d9f1f3514b461
f6931ac2073f586a35cd628c90270063725e6e1e983",
                                "derivation_path": [
                                    "010100000000000000",
                                    "0100000000000000"
                                ]
                            }],
                            "signatures": [

"8c742d77eba6c56a8db8c114e60be6c6263df6120aefd7538376129d04ec71b78b718c2085bba85254b44
bf4600ba31d4c5a7869d0be0c46d88bd5eb27490e08"
                            ]
                        },
                        {
                            "type": "data",
                            "value":
"a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a8973919"
                        }
                    ]
                }],
                "allow_additional_actions": false
            }
```

```
        ],
        "sign_complete": true
    }
}
```

Network Endpoints

These endpoints are available regardless of the wallet being disabled or not.

Submit Transaction Endpoint

Submit transaction.

Parameters

`Object`:

- `Object` - **raw_transaction**, `raw_transaction` of signed transaction.

Returns

`Object`:

- `String` - **tx_id**, transaction id, hash of transaction.

Example

Submit a raw transaction.

Request

```
curl -X POST http://localhost:9888/submit-transaction -d
'{"raw_transaction":"07010000010161015ffe8bdb49bbd08f711a54f0fbed4141b74c276de44c83199
9aac43bdd56f98309ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80c0b
1ca94120001160014d61f9b3354771283461e6cb269bec98e7ee975f6630240403d7b35ec72a80816b2182
2d645adc98af2c1f07d5b379dda4527d8585ec12ef213ada312e3807ae0ccf7206575f313ffe2b405a2863
09d3172feabe07e7e0620d013706a314a57e84c2b262c2a291e8e2b063785aea9338476a66d41be4de4f20
2013effffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80bc82eb931201160
014bdf260424fd2f322dbec9ce4ddbaed8618d8e959000139fffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffff6301160014a3f9111f3b0ee96cbd119a3ea5c60058f506fb1900"}'
```

Response

```
{
  "tx_id": "2c0624a7d251c29d4d1ad14297c69919214e78d995affd57e73fbf84ece316cb"
}
```

Submit Transactions Endpoint

Submit transactions used for batch submit transactions.

Parameters

`Object`:

- `Object` - **raw_transactions**, `raw_transactions` of signed transactions.

Returns

`Object`:

- `String` - **tx_id**, transactions id, hash of transactions.

Example

Submit a collection of raw transactions.

Request

```
curl -X POST http://localhost:9888/submit-transactions -d
'{"raw_transactions":["0701000201620160a0d36052ca3d1335120ae48e1ffb2fb6b25588628eff90f
a88bef3117dfb4301ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80ddb
2c490e906010116001431630464f2b2058fe3c1fe5bee00742eaf2da8d9630240acb57bc06f7e5de99ef3e
630ce34fc74c33d4694301202968092ca50ae7842e3331bfeb0cf7b65f383e27670c4d58aeeeb0b77e5355
957ca729298d2b4e2470c20a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a897391
90161015f72de2064ab999acf22c05b5cf9c7d53164f80038b46b1ce426708514a30a3485fffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffff80d4f4f69901000116001431630464f2b20
58fe3c1fe5bee00742eaf2da8d96302404298424e89e5528f1d0cdd9028489b9d9e3f031ec34a74440cacc
7900dc1eac9359c408a4342fc6cef935d2978919df8b23f3912ac4419800d375fac06ddb50620a86ab33ef
a9d71994270898ad99f198d60889ef617d5eaf25e776929a897391901013ffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffff8084c5b6aaea060116001431630464f2b2058fe3c1fe5be
e00742eaf2da8d900","070100010162016057 1cc5d99a2994ff6b192bc9387838a3651245cb66dad4a6bc
5f660310cebfa9ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8084c5b6
aaea0600011600143163046 4f2b2058fe3c1fe5bee00742eaf2da8d96302408c742d77eba6c56a8db8c114
e60be6c6263df6120aefd7538376129d04ec71b78b718c2085bba85254b44bf4600ba31d4c5a7869d0be0c
46d88bd5eb27490e0820a86ab33efa9d71994270898ad99f198d60889ef617d5eaf25e776929a897391902
013effffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff80faafed9901011600
1431630464f2b2058fe3c1fe5bee00742eaf2da8d900013fffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffff80ddb2c490e906011600143163046 4f2b2058fe3c1fe5bee00742eaf2da8
d900"]}'
```

Response

```
{
    "status": "success",
    "data": {
        "tx_id": ["8524bf38701c17c57e2ad7368c0d6c815eb30e92713ff5dd86c1a931cddf2e95",
"a0bbfb75c9a00bb2d4801aa95ed0479993a67acfd1cec5b77a8ff86966f52dac"]
    }
}
```

Estimate Transaction Gas Endpoint

Estimate consumed neu($1EY = 10^8 NEU$) for the transaction.

Parameters

`Object`:

- `Object` - **transaction_template**, builded transaction response.

Returns

`Object`:

- `Integer` - **total_neu**, total consumed neu($1EY = 10^8 NEU$) for execute transaction, `total_neu` is rounded up `storage_neu` + `vm_neu`.
- `Integer` - **storage_neu**, consumed neu for storage transaction .
- `Integer` - **vm_neu**, consumed neu for execute VM.

Example

Query about a transaction's gas.

Request

```
curl -X POST http://localhost:9888/estimate-transaction-gas -d
'{"transaction_template":{"allow_additional_actions":false,"raw_transaction":"07010001
0161015ffe8a1209937a6a8b22e8c01f056fd5f1730734ba8964d6b79de4a639032cecddffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffff8099c4d59901000116001485eb6eee802333
2da85df60157dc9b16cc553fb2010002013dffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffff80afa08b4f011600142b4fd033bc76b4ddf5cb00f625362c4bc7b10efa00013dffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffff8090dfc04a011600146eea1ce6cfa5
b718ae8094376be9bc1a87c9c82700","signing_instructions":[{"position":0,"witness_compone
nts":[{"keys":[{"derivation_path":["010100000000000000","0100000000000000"],"xpub":"cb
4e5932d808ee060df9552963d87f60edac42360b11d4ad89558ef2acea4d4aaf4818f2ebf5a599382b8dfc
e0a0c798c7e44ec2667b3a1d34c61ba57609de55"}],"quorum":1,"signatures":null,"type":"raw_t
x_signature"},{"type":"data","value":"1c9b5c1db7f4afe31fd1b7e0495a8bb042a271d8d7924d4f
c1ff7cf1bff15813"}]}]}}'
```

Response

```
{
  "storage_neu": 3840000,
  "total_neu": 5259000,
  "vm_neu": 1419000
}
```

Create Access Token Endpoint

Create a named access token that provides basic authentication for the HTTP protocol.
This endpoint teturns a token containing a username and password, separated by a colon.

Parameters

`Object`:

- `String` - **id**, token ID.

Optional:

- `String` - **type**, type of token.

Returns

`Object`:

- `String` - **token**, access token, authentication username and password are separated by a colon.
- `String` - **id**, token ID.
- `String` - **type**, type of token.
- `Object` - **created_at**, time to create token.

Example

Create an named access token.

Request

```
curl -X POST http://localhost:9888/create-access-token -d '{"id":"token1"}'
```

Response

```
{
  "token": "token1:1fee70f537128a201338bd5f25a3adbf33dad02eae4f4c9ac43f336a069df8f3",
  "id": "token1",
  "created_at": "2024-03-20T18:56:01.043919771+08:00"
}
```

List Access Tokens Endpoint

Returns the list of all available access tokens.

Parameters

None.

Returns

- `Array of Object`, access token array.
  - `Object`:
    - `String` - **token**, access token.
    - `String` - **id**, token ID.
    - `String` - **type**, type of token.
    - `Object` - **created_at**, time to create token.

Example

List all the available access tokens.

Request

```
curl -X POST http://localhost:9888/list-access-tokens -d '{}'
```

Response

```
[
  {
    "token":
"token1:1fee70f537128a201338bd5f25a3adbf33dad02eae4f4c9ac43f336a069df8f3",
    "id": "token1",
    "created_at": "2024-03-20T18:56:01.043919771+08:00"
  },
  {
    "token": "alice:78598c6d9fb9e3258d01f78005d4e5725ad0d45e20af90a30b577b407d4a2edd",
    "id": "alice",
    "created_at": "2024-03-20T18:56:01.043919771+08:00"
  }
]
```

Delete Access Token Endpoint

Delete existing access token.

Parameters

`Object`:

- `String` - **id**, token ID.

Returns

Nothing if the access token is deleted successfully.

Example

Delete an existing access token.

Request

```
curl -X POST http://localhost:9888/delete-access-token -d '{"id": "token1"}'
```

Response

Nothing if the operation was successful.

Check Access Token Endpoint

Check if an access token is valid.

Parameters

`Object`:

- `String` - **id**, token ID.
- `String` - **secret**, secret of token, the second part of the colon division for token.

Returns

None if the access token's validity checks out.

Example

Check whether the access token is vaild or not.

Request

```
curl -X POST http://localhost:9888/check-access-token -d '{"id": "token1", "secret":
"1fee70f537128a201338bd5f25a3adbf33dad02eae4f4c9ac43f336a069df8f3"}'
```

Response

Nothing if the operation was successful.

Create Transaction Feed Endpoint

Create a transaction feed.

Parameters

`Object`:

- `String` - **alias**, name of the transaction feed.
- `String` - **filter**, filter of the transaction feed.

Returns

None if the transaction feed is created successfully.

Example

Create a transaction feed by alias.

Request

```
curl -X POST http://localhost:9888/create-transaction-feed -d '{"alias": "test1",
"filter": "asset_id='84778a666fe453cf73b2e8c783dbc9e04bc4fd7cbb4f50caeaee99cf9967ebed'
AND amount_lower_limit = 50 AND amount_upper_limit = 100"}'
```

Response

Nothing if the operation was successful.

Get Transaction Feed Endpoint

Query details of a transaction feed by alias.

Parameters

`Object`:

- `String` - **alias**, alias of the transaction feed.

Returns

`Object`:

- `String` - **id**, id of the transaction feed.
- `String` - **alias**, alias of the transaction feed.
- `String` - **filter**, filter of the transaction feed.
- `Object` - **param**, param of the transaction feed.
    - `String` - **assetid**, asset id.
    - `Integer` - **lowerlimit**, the lower limit of asset amount.
    - `Integer` - **upperlimit**, the upper limit of asset amount.
    - `String` - **transtype**, type of transaction.

Example

List the available transaction feed by alias.

Request

```
curl -X POST http://localhost:9888/get-transaction-feed -d '{"alias": "test1"}'
```

Response

```
{
  "alias": "test1",
  "filter":
"asset_id='84778a666fe453cf73b2e8c783dbc9e04bc4fd7cbb4f50caeaee99cf9967ebed' AND
amount_lower_limit = 50 AND amount_upper_limit = 100",
  "param": {}
}
```

List Transaction Feeds Endpoint

Returns the list of all available transaction feeds.

Parameters

None.

Returns

- `Array of Object`, the transaction feeds.
    - `Object`:

- `String` - **id**, id of the transaction feed.
- `String` - **alias**, name of the transaction feed.
- `String` - **filter**, filter of the transaction feed.
- `Object` - **param**, param of the transaction feed.
    - `String` - **assetid**, asset id.
    - `Integer` - **lowerlimit**, the lower limit of asset amount.
    - `Integer` - **upperlimit**, the upper limit of asset amount.
    - `String` - **transtype**, type of transaction.

Example

List all the available `txfeed`.

Request

```
curl -X POST http://localhost:9888/list-transaction-feeds -d '{}'
```

Response

```
[
  {
    "alias": "test1",
    "filter":
"asset_id='84778a666fe453cf73b2e8c783dbc9e04bc4fd7cbb4f50caeaee99cf9967ebed' AND
amount_lower_limit = 50 AND amount_upper_limit = 100",
    "param": {}
  },
  {
    "alias": "test2",
    "filter":
"asset_id='cee6a588cc3fc280749021ef42d5209952a1e6feceada4e69dd8a424ad22b199' AND
amount_lower_limit = 30 AND amount_upper_limit = 100",
    "param": {}
  }
]
```

Delete Transaction Feed Endpoint

Delete a transaction feed by alias.

Parameters

`Object`:

- `String` - **alias**, alias of the transaction feed.

Returns

Nothing if the transaction feed is deleted successfully.

Example

Delete a transaction feed by it's alias.

Request

```
curl -X POST http://localhost:9888/delete-transaction-feed -d '{"alias": "test1"}'
```

Response

Nothing if the operation was successful.

Update Transaction Feed Endpoint

Update transaction feed.

Parameters

`Object`:

- `String` - **alias**, name of the transaction feed.
- `String` - **filter**, filter of the transaction feed.

Returns

Nothing if the transaction feed is updated success.

Example

Deleted when the `txfeed` exists, and recreate it with alias and filter:

Request

```
curl -X POST http://localhost:9888/update-transaction-feed -d '{"alias": "test1",
"filter": "asset_id='84778a666fe453cf73b2e8c783dbc9e04bc4fd7cbb4f50caeaee99cf9967ebed'
AND amount_lower_limit = 60 AND amount_upper_limit = 80"}'
```

Response

Nothing if the operation was successful.

Get Unconfirmed Transaction Endpoint

Query `mempool` transaction by transaction ID.

Parameters

`Object`:

- `String` - **tx_id**, transaction id, hash of transaction.

Returns

`Object`:

- `String` - **id**, transaction id, hash of the transaction.
- `Integer` - **version**, version of transaction.
- `Integer` - **size**, size of transaction.
- `Integer` - **time_range**, the time range of transaction.
- `Boolean` - **status_fail**, whether the state of the request has failed.
- `String` - **mux_id**, the previous transaction mux id(wallet enable can be acquired, this place is empty).
- `Array of Object` - **inputs**, object of inputs for the transaction(input struct please refer to get-transction API description).
- `Array of Object` - **outputs**, object of outputs for the transaction(output struct please refer to get-transction API description).

Example

Retrieve a transaction from the `mempool` by `tx_id`.

Request

```
curl -X POST http://localhost:9888/get-unconfirmed-transaction -d '{"tx_id":
"382090f24fbfc2f737fa7372b9d161a43f00d1c597a7130a56589d1f469d04b5"}'
```

Response

```
{
  "id": "382090f24fbfc2f737fa7372b9d161a43f00d1c597a7130a56589d1f469d04b5",
  "inputs": [
    {
      "address": "ey1qqrm7ruecx7yrg9smtwmnmgj3umg9vcukgy5sdj",
      "amount": 41250000000,
      "asset_definition": {},
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "001400f7e1f338378834161b5bb73da251e6d0566396",
      "input_id": "a0c2fa0719bfe1446681537dcf1f8d0f03add093e29d12481eb807e07778d7b3",
      "spent_output_id":
"161b44e547a6cc68d732eb64fa38031da98211a99319e088cfe632223f9ac6d8",
      "type": "spend",
      "witness_arguments": [

"cf0e1b217ab92ade8e81fab10f9f307bb5cc1ad947b5629e3f7a760aba722f5044f2ab59ec92fa4264ff5
811de4361abb6eabd7e75ffd28a813a98ceff434c01",
        "6890a19b21c326059eef211cd8414282a79d3b9203f2592064221fd360e778a7"
      ]
    }
  ],
  "mux_id": "842cd07eed050b547377b5b123f14a5ec0d76933d564f030cf4d5d5c15769645",
  "outputs": [
    {
      "address": "ey1qehxd5cdnepckh5jc72ggn30havd78lsgcqmt7k",
      "amount": 21230000000,
      "asset_definition": {},
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "0014cdccda61b3c8716bd258f29089c5f7eb1be3fe08",
      "id": "a8f21ad24689c290634db85278f56d152efe6fe08bc194e5dee5127ed6d3ebee",
      "position": 0,
      "type": "control"
    },
    {
      "address": "ey1q2me9gwccnm3ehpnrcr99gcnj730js2zfucms3r",
      "amount": 20000000000,
      "asset_definition": {},
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "001456f2543b189ee39b8663c0ca546272f45f282849",
      "id": "78219e422ea3257aeb32f6d952b5ce5560dab1d6440c9f3aebcdaad2a852d2a8",
      "position": 1,
      "type": "control"
    }
  ],
  "size": 664,
  "status_fail": false,
  "time_range": 0,
  "version": 1
}
```

List Unconfirmed Transactions Endpoint

Returns the total number of `mempool` transactions and the list of transaction IDs.

Parameters

None.

Returns

`Object`:

- `Integer` - **total**, version of transaction.
- `Array of Object` - **tx_ids**, list of transaction id.

Example

Retrieve the total of `mempool` transactions and their respective IDs.

Request

```
curl -X POST http://localhost:9888/list-unconfirmed-transactions -d '{}'
```

Response

```
{
  "total": 2,
  "tx_ids": [
    "382090f24fbfc2f737fa7372b9d161a43f00d1c597a7130a56589d1f469d04b5",
    "fc2da5dfa094c2170144f149fa07a312983157aec0ec95063a1319eedcb2d23b"
  ]
}
```

Decode Raw Transaction Endpoint

Decode a serialized transaction hex string into a JSON object describing the transaction.

Parameters

`Object`:

- `String` - **raw_transaction**, hexstring of raw transaction.

Returns

`Object`:

- `String` - **tx_id**, transaction ID.
- `Integer` - **version**, version of transaction.
- `String` - **size**, size of transaction.
- `String` - **time_range**, time range of transaction.
- `String` - **fee**, fee for sending transaction.
- `Array of Object` - **inputs**, object of inputs for the transaction(input struct please refer to get-

transction API description).

- `Array of Object` - **outputs**, object of outputs for the transaction

Example

Ask for a decoded transaction from the contents of a raw transaction.

Request

```
curl -X POST http://localhost:9888/decode-raw-transaction -d '{"raw_transaction":
"070100010161015fc8215913a270d3d953ef431626b19a89adf38e2486bb235da732f0afed515299fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8099c4d59901000116001456ac1
70c7965eeac1cc34928c9f464e3f88c17d8630240b1e99a3590d7db80126b273088937a87ba1e8d2f91021
a2fd2c36579f7713926e8c7b46c047a43933b008ff16ecc2eb8ee888b4ca1fe3fdf082824e0b3899b02202
fb851c6ed665fcd9ebc259da1461a1e284ac3b27f5e86c84164aa518648222602013efffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffff80bbd0ec980101160014c3d320e1dc4fe787e9f
13c1464e3ea5aae96a58f00013cfffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffff8084af5f01160014bb93cdb4eca74b068321eeb84ac5d33686281b6500"}'
```

Response

```
{
  "fee": 20000000,
  "inputs": [
    {
      "address": "sy1q26kpwrrevhh2c8xrfy5vnaryu0ugc97csrdy69",
      "amount": 41250000000,
      "asset_definition": {},
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "001456ac170c7965eeac1cc34928c9f464e3f88c17d8",
      "input_id": "9963265eb601df48501cc240e1480780e9ed6e0c8f18fd7dd57954068c5dfd02",
      "spent_output_id":
"01bb3309666618a1507cb5be845b17dee5eb8028ee7e71b17d74b4dc97085bc8",
      "type": "spend",
      "witness_arguments": [

"b1e99a3590d7db80126b273088937a87ba1e8d2f91021a2fd2c36579f7713926e8c7b46c047a43933b008
ff16ecc2eb8ee888b4ca1fe3fdf082824e0b3899b02",
        "2fb851c6ed665fcd9ebc259da1461a1e284ac3b27f5e86c84164aa5186482226"
      ]
    }
  ],
  "outputs": [
    {
      "address": "sy1qc0fjpcwuflnc06038s2xfcl2t2hfdfv0lxzg7s",
      "amount": 41030000000,
      "asset_definition": {},
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "0014c3d320e1dc4fe787e9f13c1464e3ea5aae96a58f",
      "id": "567b34857614d16292220beaca16ce34b939c75023a49cc43fa432fff51ca0dd",
      "position": 0,
      "type": "control"
    },
    {
      "address": "sy1qhwfumd8v5a9sdqepa6uy43wnx6rzsxm9essn4l",
      "amount": 200000000,
      "asset_definition": {},
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "0014bb93cdb4eca74b068321eeb84ac5d33686281b65",
      "id": "a8069d412e48c2b2994d2816758078cff46b215421706b4bad41f72a32928d92",
      "position": 1,
      "type": "control"
    }
  ],
  "size": 332,
  "time_range": 0,
  "tx_id": "4c97d7412b04d49acc33762fc748cd0780d8b44086c229c1a6d0f2adfaaac2db",
  "version": 1
}
```

Get Block Count Endpoint

Returns the current block height for the blockchain.

Parameters

None.

Returns

`Object`:

- `Integer` - **block_count**, recent block height of the blockchain.

Example

Retrieve the current block height.

Request

```
curl -X POST http://localhost:9888/get-block-count
```

Response

```
{
    "block_count": 519
}
```

Get Block Hash Endpoint

Returns the current block hash for the block height of the blockchain.

Parameters

None.

Returns

`Object`:

- `String` - **block_hash**, recent block hash of the blockchain.

Example

Retrieves the block’s hash for the current block height.

Request

```
curl -X http://localhost:9888/POST get-block-hash
```

Response

```
{
  "block_hash": "997bf5cecb4df097991a7a121a7fd3cb5a404fa856e3d6032c791ac07bc7c74c"
}
```

Get Block Endpoint

Returns the details of a block by block height or block hash.

Parameters

`Object`: **block_height** | **block_hash**

Optional:

- `String` - **block_hash**, hash of block.
- `Integer` - **block_height**, height of block.

Returns

`Object`:

- `String` - **hash**, hash of block.
- `Integer` - **size**, size of block.
- `Integer` - **version**, version of block.
- `Integer` - **height**, height of block.
- `String` - **previous_block_hash**, previous block hash.
- `Integer` - **timestamp**, timestamp of block.
- `Integer` - **nonce**, nonce value.
- `Integer` - **bits**, bits of difficulty.
- `String` - **difficulty**, difficulty value(String type).
- `String` - **transaction_merkle_root**, merkle root of transaction.
- `String` - **transaction_status_hash**, merkle root of transaction status.
- `Array of Object` - **transactions**, transaction object:
  - `String` - **id**, transaction id, hash of the transaction.
  - `Integer` - **version**, version of transaction.
  - `Integer` - **size**, size of transaction.
  - `Integer` - **time_range**, the unix timestamp for when the requst was responsed.
  - `Boolean` - **status_fail**, whether the state of the request has failed.
  - `String` - **mux_id**, the previous transaction mux id(source id of utxo).
  - `Array of Object` - **inputs**, object of inputs for the transaction.
    - `String` - **type**, the type of input action, available option include: 'spend', 'issue', 'coinbase'.
    - `String` - **asset_id**, asset id.
    - `String` - **asset_alias**, name of asset.
    - `Object` - **asset_definition**, definition of asset(json object).
    - `Integer` - **amount**, amount of asset.
    - `Object` - **issuance_program**, issuance program, it only exist when type is 'issue'.
    - `Object` - **control_program**, control program of account, it only exist when type is 'spend'.
    - `String` - **address**, address of account, it only exist when type is 'spend'.
    - `String` - **spent_output_id**, the front of outputID to be spent in this input, it only exist when type is 'spend'.
    - `String` - **account_id**, account id.
    - `String` - **account_alias**, name of account.
    - `Object` - **arbitrary**, arbitrary infomation can be set by miner, it only exist when type is 'coinbase'.

        - `String` - **input_id**, hash of input action.
        - `Array of String` - **witness_arguments**, witness arguments.
    - `Array of Object` - **outputs**, object of outputs for the transaction.
        - `String` - **type**, the type of output action, available option include: 'retire', 'control'.
        - `String` - **id**, outputid related to utxo.
        - `Integer` - **position**, position of outputs.
        - `String` - **asset_id**, asset id.
        - `String` - **asset_alias**, name of asset.
        - `Object` - **asset_definition**, definition of asset(json object).
        - `Integer` - **amount**, amount of asset.
        - `String` - **account_id**, account id.
        - `String` - **account_alias**, name of account.
        - `Object` - **control_program**, control program of account.
        - `String` - **address**, address of account.

Example

Get specified block information by `block_hash` or `block_height`, if both exists, the block result is queried by hash.

Request

```
curl -X POST http://localhost:9888/get-block -d '{"block_height": 43, "block_hash":
"886a8e85b275e7d65b569ba510875c0e63dece1a94569914d7624c0dac8002f9"}'
```

Response

```
{
  "bits": 2305843009222082600,
  "difficulty": "5549086336",
  "hash": "886a8e85b275e7d65b569ba510875c0e63dece1a94569914d7624c0dac8002f9",
  "height": 43,
  "nonce": 3,
  "previous_block_hash":
"2c72ccbd53b92a4f9423c5b610b4e106bbe8fbf3ecf2e16a1266b17ee323ff9d",
  "size": 386,
  "timestamp": 1521614189,
  "transaction_merkle_root":
"77d40262cfeca3a16d4587132974552ef00951e43ce567a26801ebc3dbdb4d96",
  "transaction_status_hash":
"53c0ab896cb7a3778cc1d35a271264d991792b7c44f5c334116bb0786dbc5635",
  "transactions": [
    {
      "id": "4576b1b1ec251da3263dbdd5486bcbf9a1cd1f712172dbe7a7a5fe46ab194629",
      "inputs": [
        {
          "amount": 0,
          "arbitrary": "09",
          "asset_definition": "7b7d",
          "asset_id":
"0000000000000000000000000000000000000000000000000000000000000000",
          "input_id":
"6cb8491e4b1cbdc052c2fdb5f2849194d59118b954d5ea5244bbd20e3cff3b80",
          "type": "coinbase",
          "witness_arguments": null
        }
      ],
      "mux_id": "2383cefe8a34ea5810cc0706f2cf8cf08a106f90fc3eb3441f723cecdbc61331",
      "outputs": [
        {
          "address": "sy1q4pkg8msjumtep7ecsdzuct3tsuzk5pmnm3p8nr",
          "amount": 624000000000,
          "asset_definition": "7b7d",
          "asset_id":
"ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
          "control_program": "0014f3403bcd8b443d03882a280b50f6f98986e1a255",
          "id": "da87b40854a9b93be72ecdc24fe7bb03986ea3530e344b0f918f0788c3d83717",
          "position": 0,
          "type": "control"
        }
      ],
      "size": 77,
      "status_fail": false,
      "time_range": 0,
      "version": 1
    }
  ],
  "version": 1
}
```

Get Block Header Endpoint

Returns the details of a block header by block height or block hash.

Parameters

`Object`: **block_height** | **block_hash**

Optional:

- `String` - **block_hash**, hash of block.
- `Integer` - **block_height**, height of block.

Returns

`Object`:

- `Object` - **block_header**, header of block.
- `Integer` - **reward**, reward.

Example

Retrieve the block header details.

Request

```
curl -X POST http://localhost:9888/get-block-header -d '{"block_height": 43,
"block_hash": "886a8e85b275e7d65b569ba510875c0e63dece1a94569914d7624c0dac8002f9"}'
```

Response

```
{
  "block_header":
"01019601e87da37e7d73f31d54304c719c9058ec7bc7de7819deda89a7c8834a99bc05b8fbdbe6d60540e
ba9e5d5cb79fd87b3c0fad32f6772c1e4483f2a070e093a6176d85226d986a8c9c377e5192668bc0a367e4
a4764f11e7c725ecced1d7b6a492974fab1b6d5bc00ad918480808080801e",
  "reward": 41250000000
}
```

Get Difficulty Endpoint

Returns the block difficulty by block height or block hash.

Parameters

`Object`:

Optional:

- `String` - **block_hash**, hash of block.
- `Integer` - **block_height**, height of block.

Returns

`Object`:

- `Integer` - **bits**, bits of block.
- `String` - **difficulty**, difficulty of block.
- `String` - **hash**, block hash.

- `Integer` - **height**, block height.

Example

Get difficulty for specified block hash / height.

Request

```
curl -X POST http://localhost:9888/get-difficulty -d '{"block_height": 506,
"block_hash": "d1fce60caea5466eae2b812e4586b55120c52aca27b6c92bd7c51e9cda82dcdf"}'
```

Response

```
{
  "bits": 2161727821137910500,
  "difficulty": "15154807",
  "hash": "d1fce60caea5466eae2b812e4586b55120c52aca27b6c92bd7c51e9cda82dcdf",
  "height": 506
}
```

Get Hash Rate Endpoint

Returns the block hash rate by block height or block hash, it returns the current block hash rate when request is empty.

Parameters

`Object`:

Optional:

- `String` - **block_hash**, hash of block.
- `Integer` - **block_height**, height of block.

Returns

`Object`:

- `Integer` - **hash_rate**, difficulty of block.
- `String` - **hash**, block hash.
- `Integer` - **height**, block height.

Example

Get hash rate for specified block hash / height.

Request

```
curl -X POST http://localhost:9888/get-hash-rate -d '{"block_height": 506,
"block_hash": "d1fce60caea5466eae2b812e4586b55120c52aca27b6c92bd7c51e9cda82dcdf"}'
```

Response

```
{
  "hash": "d1fce60caea5466eae2b812e4586b55120c52aca27b6c92bd7c51e9cda82dcdf",
  "hash_rate": 7577403,
  "height": 506
}
```

Net Info Endpoint

Returns the information for the current network state.

Parameters

None.

Returns

`Object`:

- `Boolean` - **listening**, whether the node is listening.
- `Boolean` - **syncing**, whether the node is syncing.
- `Boolean` - **mining**, whether the node is mining.
- `Integer` - **peer_count**, current count of connected peers.
- `Integer` - **current_block**, current block height in the node's blockchain.
- `Integer` - **highest_block**, current highest block of the connected peers.
- `String` - **network_id**, network id.
- `Object` - **version_info**, node's version information:
  - `String` - **version**, current version of the running node.
  - `uint16` - **update**, whether there exists an update.
    - 0: no update;
    - 1: small update;
    - 2: significant update.
  - `String` - **new_version**, the newest version of the `core` if there is one.

Example

Query the node for it’s network state.

Request

```
curl -X POST http://localhost:9888/net-info
```

Response

```
{
  "listening": true,
  "syncing": false,
  "mining": false,
  "peer_count": 6,
  "current_block": 33409,
  "highest_block": 33409,
  "network_id": "mainnet",
  "version_info": {
    "version": "1.0.3",
    "update": 0,
    "new_version": "1.0.3"
  }
}
```

Is Mining Endpoint

Returns the mining status.

Parameters

None.

Returns

`Object`:

- `Boolean` - **is_mining**, whether the node is mining.

Example

Retrieve the mining status.

Request

```
curl -X POST http://localhost:9888/is-mining
```

Response

```
{
  "is_mining": true
}
```

Set Mining Endpoint

Start up node mining.

Parameters

`Object`:

- `Boolean` - **is_mining**, sets the value for the node's mining.

Returns

Nothing in case the mining has been turned on or off.

Example

Turn node mining on of off.

Request

```
curl -X POST http://localhost:9888/set-mining -d '{"is_mining": true}'
```

Response

Nothing if the operation was successful.

Gas Rate Endpoint

Query gas rate.

Parameters

None.

Returns

`Object`:

- `Integer` - **gas_rate**, gas rate.

Example

Retrieve the current gas rate.

Request

```
curl -X POST http://localhost:9888/gas-rate -d '{}'
```

Response

```
{
  "gas_rate": 1000
}
```

Verify Message Endpoint

Verify a signed message with derived `pubkey` of the address.

Parameters

`Object`:

- `String` - **address**, address for account.
- `String` - **derived_xpub**, derived xpub.
- `String` - **message**, message for signature by derived_xpub.
- `String` - **signature**, signature for message.

Returns

`Object`:

- `Boolean` - **result**, verify result.

Example

Given a signed message, verify it’s authenticity.

Request

```
curl -X POST http://localhost:9888/verify-message -d
'{"address":"ey1qx2qgvvjz734ur8x5lpfdtlau74aaa5djs0a5jn",
"derived_xpub":"6ff8c3d1321ce39a3c3550f57ba70b67dcbcef821e9b85f6150edb7f2f3f91009e67f3
075e6e76ed5f657ee4b1a5f4749b7a8c74c8e7e6a1b0e5918ebd5df4d0", "message":"this is a test
message",
"signature":"74da3d6572233736e3a439166719244dab57dd0047f8751b1efa2da26eeab251d915c1211
dcad77e8b013267b86d96e91ae67ff0be520ef4ec326e911410b609"}'
```

Response

```
{
  "result": true
}
```

Compile Endpoint

Compile an equity contract.

Parameters

`Object`:

- `String` - **contract**, content of equity contract.

Optional:

- `Array of Object` - **args**, parameters of contract.
  - `Boolean` - **boolean**, boolean parameter.
  - `Integer` - **integer**, integer parameter.
  - `String` - **string**, string parameter.

Returns

`Object`:

- `String` - **name**, contract name.
- `String` - **source**, source content of contract.
- `String` - **program**, generated program by compiling contract.
- `Array of Object` - **params**, parameters of contract.
- `String` - **value**, locked value name of contract.
- `Array of Object` - **clause_info**, clauses of contract.
- `String` - **opcodes**, opcodes of contract.
- `String` - **error**, returned error information for compiling contract.

Example

Given a contract source code, compile it into an equity contract.

Request

```
curl -X POST http://localhost:9888/compile -d '{"contract": "contract
LockWithPublicKey(publicKey: PublicKey) locks locked { clause unlockWithSig(sig:
Signature) { verify checkTxSi (publicKey, sig) unlock locked }}","args": ["string":
"e9108d3ca8049800727f6a3505b3a2710dc579405dde03c250f16d9a7e1e6e78"}]}'
```

Response

```
{
  "name": "LockWithPublicKey",
  "source": "contract LockWithPublicKey(publicKey: PublicKey) locks locked { clause
unlockWithSig(sig: Signature) { verify checkTxSig(publicKey, sig) unlock locked }}",
  "program":
"20e9108d3ca8049800727f6a3505b3a2710dc579405dde03c250f16d9a7e1e6e787403ae7cac00c0",
  "params": [
    {
      "name": "publicKey",
      "type": "PublicKey"
    }
  ],
  "value": "locked",
  "clause_info": [
    {
      "name": "unlockWithSig",
      "args": [
        {
          "name": "sig",
          "type": "Signature"
        }
      ],
      "value_info": [
        {
          "name": "locked"
        }
      ],
      "block_heights": [],
      "hash_calls": null
    }
  ],
  "opcodes": "0xe9108d3ca8049800727f6a3505b3a2710dc579405dde03c250f16d9a7e1e6e78 DEPTH
0xae7cac FALSE CHECKPREDICATE",
  "error": ""
}
```

List Peers Endpoint

Returns the list of connected peers.

Parameters

None.

Returns

- `Array of Object`, connected peers.
    - `Object`:
        - `String` - **peer_id**, peer id.
        - `String` - **remote_addr**, the address(IP and port) of connected peer.
        - `Integer` - **height**, the current height of connected peer.
        - `String` - **ping**, the delay time of connected peer.
        - `String` - **duration**, the connected time.
        - `Integer` - **total_sent**, total data sent in byte.
        - `Integer` - **total_received**, total data received in byte.
        - `Integer` - **average_sent_rate**, average data sent rate in byte.
        - `Integer` - **average_received_rate**, average data received rate in byte.
        - `Integer` - **current_sent_rate**, current data sent rate in byte.
        - `Integer` - **current_received_rate**, current data received rate in byte.

Example

Retrieve a list of the connected peers.

Request

```
curl -X POST http://localhost:9888/list-peers -d '{}'
```

Response

```
[
  {
    "peer_id":"3B58D7139B53066F2031FC1F027D2B3423FA4CE01F1FB1CC2DC4003C78413C24",
    "remote_addr":"52.83.251.197:46656",
    "height":84222,
    "ping":"40ms",
    "duration":"17.26s",
    "total_sent":17565,
    "total_received":3642187,
    "average_sent_rate":1018,
    "average_received_rate":211019,
    "current_sent_rate":2420,
    "current_received_rate":157087
    }
]
```

Disconnect Peer Endpoint

Disconnect from the specified peer.

Parameters

`Object`:

- `String` - **peer_id**, peer id.

Returns

Nothing if peer disconnected successfully.

Example

Disconnect from a specific peer.

Request

```
curl -X POST http://localhost:9888/disconnect-peer -d
'{"peer_id":"29661E8BB9A8149F01C6594E49EA80C6B18BF247946A7E2E01D8235BBBFC3594"}'
```

Response

Nothing if the operation was successful.

Connect Peer Endpoint

Connect to specified peer.

Parameters

`Object`:

- `String` - **ip**, peer IP address.
- `Integer` - **port**, peer port.

Returns

`Object`:

- `String` - **peer_id**, peer id.

- `String` - **remote_addr**, the address(IP and port) of connected peer.
- `Integer` - **height**, the current height of connected peer.
- `Integer` - **delay**, the delay time of connected peer.

Example

Request the node to connect to a specified peer.

Request

```
curl -X POST http://localhost:9888/connect-peer -d '{"ip":"139.198.177.164",
"port":46657}'
```

Response

```
{
  "peer_id": "29661E8BB9A8149F01C6594E49EA80C6B18BF247946A7E2E01D8235BBBFC3594",
  "remote_addr": "139.198.177.164:46657",
  "height": 65941,
  "delay": 0
}
```

Mining Endpoints

These endpoints are for the `CPU` miner.

Get Work Endpoint

Get the proof of work.

Parameters

None.

Returns

`Object`:

- `Object` - **block_header**, raw block header.
- `String` - **seed**, seed.

Example

Request work from the node.

Request

```
curl -X POST http://localhost:9888/get-work -d '{}'
```

Response

```
{
  "block_header":
"0101870103f2c7495164c8f3af43697e81faa21dcb2d60aa5e10ce4f233491e62420742fbeadfcd50540b
ef2670a5fade2e58ad4955e2375a04ad1e4cb9c104faddab43f4a79e35be253c9c377e5192668bc0a367e4
a4764f11e7c725ecced1d7b6a492974fab1b6d5bc00ffffff838080808020",
  "seed": "702bef3f1707577fd0d75b6359a2919fa216487fe306771e27710acbaa9164ce"
}
```

Submit Work Endpoint

Submit the proof of work.

Parameters

`Object`:

- `Object` - **block_header**, raw block header.

Returns

True if share is correct.

Example

Submit a share.

Request

```
curl -X POST http://localhost:9888/submit-work -d '{"block_header":
"0101870103f2c7495164c8f3af43697e81faa21dcb2d60aa5e10ce4f233491e62420742fbeadfcd50540b
ef2670a5fade2e58ad4955e2375a04ad1e4cb9c104faddab43f4a79e35be253c9c377e5192668bc0a367e4
a4764f11e7c725ecced1d7b6a492974fab1b6d5bc00ffffff838080808020"}'
```

Response

```
true / error
```

Get Work `JSON` Endpoint

Get the proof of work by `JSON`.

Parameters

None.

Returns

`Object`:

- `Object` - **block_header**, Object of block header.
  - `Integer` - **version**, version of block.
  - `Integer` - **height**, height of block.
  - `String` - **previous_block_hash**, previous block hash.
  - `Integer` - **timestamp**, timestamp of block.
  - `Integer` - **nonce**, nonce value.
  - `Integer` - **bits**, bits of difficulty.
  - `Object` - **block_commitment**, Object of block commitment.
    - `String` - **transaction_merkle_root**, merkle root of transaction.
    - `String` - **transaction_status_hash**, merkle root of transaction status.
- `String` - **seed**, seed.

Example

Request work from the node.

Request

```
curl -X POST http://localhost:9888/get-work-json -d '{}'
```

Response

```
{
  "block_header": {
    "version": 1,
    "height": 62960,
    "previous_block_hash":
"dabdb926f8635791ac43f5d5fc62a4597e10e140f00aced3af621a77ead4e9fd",
    "timestamp": 1533006396,
    "nonce": 0,
    "bits": 2017612633069711400,
    "block_commitment": {
      "transaction_merkle_root":
"a13fc86af3852ab73e30c3ae30e8cedbe990560a3c0f20dc37c4c14562b94802",
      "transaction_status_hash":
"c9c377e5192668bc0a367e4a4764f11e7c725ecced1d7b6a492974fab1b6d5bc"
    }
  },
  "seed": "2e2010e11289d6b273b7b1ea947f98cc5ad60d206df184b1501f8ac903fa01a9"
}
```

Submit Work `JSON` Endpoint

Submit the proof of work by `JSON`.

Parameters

`Object`:

- `Object` - **block_header**, Object of block header.
    - `Integer` - **version**, version of block.
    - `Integer` - **height**, height of block.
    - `String` - **previous_block_hash**, previous block hash.
    - `Integer` - **timestamp**, timestamp of block.
    - `Integer` - **nonce**, nonce value.
    - `Integer` - **bits**, bits of difficulty.
    - `Object` - **block_commitment**, Object of block commitment.
        - `String` - **transaction_merkle_root**, merkle root of transaction.
        - `String` - **transaction_status_hash**, merkle root of transaction status.

Returns

True if success

Example

Submit a share.

Request

```
curl -X POST http://localhost:9888/submit-work-json -d
'{"block_header":{"version":1,"height":62960,"previous_block_hash":"dabdb926f8635791ac
43f5d5fc62a4597e10e140f00aced3af621a77ead4e9fd","timestamp":1533006396,"nonce":0,"bits
":2017612633069711400,"block_commitment":{"transaction_merkle_root":"a13fc86af3852ab73
e30c3ae30e8cedbe990560a3c0f20dc37c4c14562b94802","transaction_status_hash":"c9c377e519
2668bc0a367e4a4764f11e7c725ecced1d7b6a492974fab1b6d5bc"}}}'
```

Response

```
true / error
```

Error Codes

`EY0XX`: `API` errors

These errors are for the `API` in general.

Table 1. `EY0XX`: `API` Errors

Code	Message	Description
EY000	`EIYARO` `API` Error	Non-EY standard error
EY001	Request timed out	`API` request timeout
EY002	Invalid request body	Illegal `API` request body

`EY1XX`: Network errors

These errors pertain to the network.

Table 2. `EY1XX`: Network Errors

Code	Message	Description
EY103	A peer core is operating on a different blockchain network	Blockchain network type mismatch

`EY2XX`: Signature errors

These are errors pertaining to the signature of transactions/messages.

Table 3. `EY2XX`: Signature Errors

Code	Message	Description
EY200	Quorum must be greater than 1 and less than or equal to the length of `xpubs`	The number of signatures required exceeded the number of signatures actually needed
EY201	Invalid `xpub` format	Incorrectly formatted signature
EY202	At least one `xpub` is required	Lack of master key
EY204	Root `xpubs` cannot contain the same key more than once	Master Key Repeat

`EY7XX`: Transaction errors

These are errors pertaining to transactions.

Transaction Construction

Table 4. `EY70X`,`EY710`: Transaction Construction Errors

Code	Message	Description
EY700	Funds of account are insufficient	Insufficient balance of assets
EY701	Available funds of account are immature	Coinbase transactions are immature and coins are not spendable
EY702	Available UTXOs of account have been reserved	Assets are locked out for five minutes and cannot be spent (typically generated by incorrect password entry)
EY703	Not found UTXO with given hash	UTXO is not part of the current wallet
EY704	Invalid action type	Action type does not exist
EY705	Invalid action object	Action input content error
EY706	Invalid action construction	Action structure error (input only or output only)
EY707	One or more fields are missing	Action input content is missing
EY708	Invalid asset amount	Asset quantity incorrectly formatted (exceeded maximum quantity)
EY709	Not found account	Account does not exist
EY710	Not found asset	Assets do not exist

Transaction Verification

Table 5. `EY73X`,`EY74X`: Transaction Verification Errors

Code	Message	Description
EY730	Invalid transaction version	Incorrect transaction version
EY731	Invalid transaction size	Transaction size cannot be 0
EY732	Invalid transaction time range	Out of Transaction Timeframe for voiding unconfirmed transactions that have stayed too long
EY733	Not standard transaction	Not a standard transaction, using the contract address to accept `EY` Times error
EY734	Invalid coinbase transaction	Illegal coinbase trading
EY735	Invalid coinbase `assetID`	Illegal coinbase asset IDs
EY736	Invalid coinbase arbitrary size	Coinbase size is too large and additional data exceeds a certain limit

Code	Message	Description
EY737	No results in the transaction	Transaction action hash missing
EY738	Mismatched `assetID`	Mismatched asset IDs, wrong asset ID when posting assets
EY739	Mismatched value source/dest position	Mismatched action positions
EY740	Mismatched reference	Mismatched references
EY741	Mismatched value	Mismatched values, action’s asset value
EY742	Missing required field	Mismatched fields, action entered with mismatched asset value types
EY743	No source for value	Input source does not exist
EY744	Arithmetic overflow/underflow	Calculation overflow, asset calculation value exceeds limits
EY745	Invalid source or destination position	Action position mismatch
EY746	Unbalanced asset amount between input and output	Imbalance in total imported and exported non-`EY` assets
EY747	Gas credit has been spent	Number of `UTXO`s exceeds upper limit (currently 21)
EY748	Gas usage calculate got a math error	Gas arithmetic error

Virtual Machine

Table 6. `EY76X`,`EY77X`: Virtual Machine Errors

Code	Message	Description
EY760	Alt stack underflow	Sub-VM stack overflow
EY761	Bad value	Illegal stack data
EY762	Wrong context	Wrong context value, context is the virtual machine execution context
EY763	Data stack underflow	Virtual Machine Data Overflow
EY764	Disallowed opcode	Virtual machine command does not exist
EY765	Division by zero	Division error
EY766	False result for executing VM	Virtual machine execution results in False
EY767	Program size exceeds max `int32`	Contract byte size exceeds `int32` limit
EY768	Arithmetic range error	There was an error in the calculation
EY769	RETURN executed	Results returned by executing the `opfail` command
EY770	Run limit exceeded because the EY Fee is insufficient	Insufficient Gas costs, resulting in contract termination

Code	Message	Description
EY771	Unexpected end of program	Error in entering contract program parameters
EY772	Unrecognized token	Unrecognised virtual machine command data
EY773	Unexpected error	Abnormal error
EY774	Unsupported `VM` because the version of the `VM` is mismatched	Virtual machine version mismatch
EY775	VERIFY failed	Failure to execute verify command

`EY8XX`: `HMS` errors

These are errors pertaining to `HMS`.

Table 7. `EY8XX`: `HMS` Errors

Code	Message	Description
EY800	Key Alias already exists	Duplicate key aliases
EY801	Invalid after in query	This bug is deprecated
EY802	Key not found or wrong password	The key does not exist or the password is wrong
EY803	Requested key aliases exceeds limit	This bug is deprecated
EY804	Could not decrypt key with given passphrase	Decryption process failed
EY860	Request could not be authenticated	Access token error

Transactions

This addendum tries to explain how to build transactions in more detail.

Acccount management mode

This section focuses on users sending transactions using Eiyaro's own account mode.

Step 1: Build Transaction

API interface `build-transaction`, code core/api/transact.go#L117

As an example, a standard non-EY asset transfer transaction with an asset ID of full F denotes an EY asset, in this example the EY asset is only used as a handling fee, and the transaction denotes spending 99 specific assets into the specified address. Where the input request json format for building the transaction is as follows:

```
{
  "base_transaction": null,
  "actions": [
    {
      "account_id": "0ER7MEFGG0A02",
      "amount": 20000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "type": "spend_account"
    },
    {
      "account_id": "0ER7MEFGG0A02",
      "amount": 99,
      "asset_id": "42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f",
      "type": "spend_account"
    },
    {
      "amount": 99,
      "asset_id": "42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f",
      "address": "sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me",
      "type": "control_address"
    }
  ],
  "ttl": 0,
  "time_range": 0
}
```

The request objects corresponding to the source code are as follows:

```
// BuildRequest is main struct when building transactions
type BuildRequest struct {
    Tx        *types.TxData            `json:"base_transaction"`
    Actions   []map[string]interface{} `json:"actions"`
    TTL       json.Duration            `json:"ttl"`
    TimeRange uint64                   `json:"time_range"`
}
```

The structure fields are described below:

- `Tx` The `TxData` portion of the transaction, this field is reserved and empty is sufficient
- `TTL` The time to live (in milliseconds) to build the transaction, meaning that within this timeframe, utxo that has been cached cannot be used to build the transaction again, unless there is enough utxo left to build a new transaction, otherwise an error will be reported. When `ttl` is 0 it is set to 600s by default, i.e. 5 minutes.
- `TimeRange` timestamp, which means that the transaction will not be submitted to the chain after that timestamp (block height), to prevent the transaction from waiting too long due to transmission delays in the network, if the transaction isn't packaged within a specific time range, the transaction will automatically expire

- `Actions` The `actions` structure of a transaction, all transactions are composed of actions, and the `interface{}` of the `map` type ensures the extensibility of the action type. Which action must contain type field, used to distinguish between different action types, `action` mainly contains `input` and `output` two types, its detailed description is as follows:
  - `input action` type:
    - `issue` issue asset
    - `spend_account` spend utxo in account mode
    - `spend_account_unspent_output` Spend the specified utxo directly.
  - `output action` Type:
    - `control_address` Receive in address mode.
    - `control_program` Receive in (program) contract mode.
    - `retire` Destroy the asset

Caveat:

- A transaction must contain at least one input and output (except for coinbase transactions, which are generated by the system and are not described here), otherwise the transaction will report an error.
- Except for EY assets (all transactions use EY assets as fees), when constructing inputs and outputs, the sum of all inputs and outputs must be equal, otherwise the trade will report an input/output imbalance error message.
- Fee for the transaction: EY asset amount for all inputs - EY asset amount for all outputs
- The asset count in the transaction is in neu, the unit conversion of EY is as follows: 1 EY = 1000 mEY = 100000000 neu

Introduction to Action

The following is a detailed description of the various `action` types used when building transactions.

Issue: `issue`

The source code for the `issueAction` structure is shown below:

```
type issueAction struct {
    assets *Registry
    bc.AssetAmount
}

type AssetAmount struct {
    AssetId *AssetID `protobuf:"bytes,1,opt,name=asset_id,json=assetId"
json:"asset_id,omitempty"`
    Amount  uint64   `protobuf:"varint,2,opt,name=amount" json:"amount,omitempty"`
}
```

The structure fields are described below:

- `assets` is mainly used for asset management, no need to set parameter by user.
- `AssetAmount` indicates the asset ID and the corresponding number of assets that the user needs to issue, here the `AssetID` needs to be created by `create-asset`, and the asset ID of `EY` cannot be used here.

The `JSON` format of `issueAction` is:

```
{
  "amount": 100000000,
  "asset_id": "3152a15da72be51b330e1c0f8e1c0db669269809da4f16443ff266e07cc43680",
  "type": "issue"
}
```

For example a sample transaction for issuing an asset is shown below:
(This transaction represents the issuance of an asset of `42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f` in the quantity of `90,000,000` `assetIDs` to the receiving address `sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me`, where the handling fee is `20000000` neu's EY assets)

```
{
  "base_transaction": null,
  "actions": [
    {
      "account_id": "0ER7MEFGG0A02",
      "amount": 20000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "type": "spend_account"
    },
    {
      "amount": 900000000,
      "asset_id": "42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f",
      "type": "issue"
    },
    {
      "amount": 900000000,
      "asset_id": "42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f",
      "address": "sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me",
      "type": "control_address"
    }
  ],
  "ttl": 0,
  "time_range": 0
}
```

Spend Account: `spend_account`

The source code for the `spendAction` structure is as follows:

```
type spendAction struct {
    accounts *Manager
    bc.AssetAmount
    AccountID   string  `json:"account_id"`
    ClientToken *string `json:"client_token"`
}

type AssetAmount struct {
    AssetId *AssetID `protobuf:"bytes,1,opt,name=asset_id,json=assetId"
json:"asset_id,omitempty"`
    Amount  uint64   `protobuf:"varint,2,opt,name=amount" json:"amount,omitempty"`
}
```

The structure fields are described below:

- `Accounts` is mainly used for the management of accounts, no need to set parameters by the user
- `AccountID` Indicates the account ID on which the asset is to be spent.
- `AssetAmount` Indicates the ID of the asset to be spent and the corresponding asset amount.
- `ClientToken` indicates the restriction of UTXO for the Reserve user, currently it can be left empty or not filled.

The `JSON` format of `spendAction` is:

```
{
  "account_id": "0BF63M2U00A04",
  "amount": 2000000000,
  "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
  "type": "spend_account"
}
```

For example a sample transaction to transfer an asset is as follows:
(This transaction means transferring EY assets in the amount of `100000000`neu to the address `sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me` by means of an account, where the fee of `20000000`neu = the number of input EY assets - the number of output EY assets.)

```
{
  "base_transaction": null,
  "actions": [
    {
      "account_id": "0ER7MEFGG0A02",
      "amount": 120000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "type": "spend_account"
    },
    {
      "amount": 100000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "address": "sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me",
      "type": "control_address"
    }
  ],
  "ttl": 0,
  "time_range": 0
}
```

Spend Account Unspent Output: `spend_account_unspent_output`

The source code for the `spendUTXOAction` structure is shown below:

```
type spendUTXOAction struct {
    accounts *Manager
    OutputID *bc.Hash `json:"output_id"`
    ClientToken *string `json:"client_token"`
}
```

The structure fields are described below:

- `accounts` is mainly used for account management, no need for user to set parameters
- `OutputID` indicates the ID of the UTXO that needs to be spent, you can query the available UTXOs according to `list-unspent-outputs`, where `OutputID` corresponds to the `id` field of the result returned by this API.
- `ClientToken` indicates the restriction of Reserve user UTXO, it can be left empty or not filled.

The `JSON` format of `spendUTXOAction` is:

```
{
  "type": "spend_account_unspent_output",
  "output_id": "58f29f0f85f7bd2a91088bcbe536dee41cd0642dfb1480d3a88589bdbfd642d9"
}
```

For example, a sample transaction to transfer an asset by spending UTXO is as follows:
(This transaction represents the transfer of EY assets of `100000000`neu directly by spending UTXO to the address `sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me`, where the fee = the UTXO value of the input EY assets - the number of the output EY assets)

```
{
  "base_transaction": null,
  "actions": [
    {
      "output_id": "58f29f0f85f7bd2a91088bcbe536dee41cd0642dfb1480d3a88589bdbfd642d9",
      "type": "spend_account_unspent_output"
    },
    {
      "amount": 100000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "address": "sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me",
      "type": "control_address"
    }
  ],
  "ttl": 0,
  "time_range": 0
}
```

Control Address: `control_address`

The source code for the `controlAddressAction` structure is shown below:

```
type controlAddressAction struct {
    bc.AssetAmount
    Address string `json:"address"`
}

type AssetAmount struct {
    AssetId *AssetID `protobuf:"bytes,1,opt,name=asset_id,json=assetId"
json:"asset_id,omitempty"`
    Amount  uint64   `protobuf:"varint,2,opt,name=amount" json:"amount,omitempty"`
}
```

The structure fields are described below:

- `Address` Indicates the address of the receiving asset, which can be created according to the `create-account-receiver` API interface
- `AssetAmount` Indicates the received asset ID and the corresponding number of assets.

The `JSON` format of `controlAddressAction` is:

```
{
  "amount": 100000000,
  "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
  "address": "ey1q50u3z8empm5ke0g3ngl2t3sqtr6sd7cepd3z68",
  "type": "control_address"
}
```

For example a sample transaction to transfer an asset is as follows:
(This transaction represents the transfer of EY assets of `100000000`neu by means of an account to the address `sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me`, where the type `control_address` indicates that the address is used as the means of receipt.)

```
{
  "base_transaction": null,
  "actions": [
    {
      "account_id": "0ER7MEFGG0A02",
      "amount": 120000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "type": "spend_account"
    },
    {
      "amount": 100000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "address": "sy1qxe4jwhkekgnxkezu7xutu5gqnnpmyc8ppq98me",
      "type": "control_address"
    }
  ],
  "ttl": 0,
  "time_range": 0
}
```

Control Program: `control_program`

The source code for the `controlProgramAction` structure is shown below:

```
type controlProgramAction struct {
    bc.AssetAmount
    Program json.HexBytes `json:"control_program"`
}

type AssetAmount struct {
    AssetId *AssetID `protobuf:"bytes,1,opt,name=asset_id,json=assetId"
json:"asset_id,omitempty"`
    Amount  uint64   `protobuf:"varint,2,opt,name=amount" json:"amount,omitempty"`
}
```

The structure fields are described below:

- `Program` Indicates the contract script for receiving assets, which can be created according to the `create-account-receiver` API interface to receive a `program` (the `program` and `address` of the returned result are one-to-one correspondence)
- `AssetAmount` indicates the received asset ID and the corresponding asset count.

The `JSON` format of `controlProgramAction` is:

```
{
  "amount": 100000000,
  "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
  "control_program":"0014a3f9111f3b0ee96cbd119a3ea5c60058f506fb19",
  "type": "control_program"
}
```

For example a sample transaction to transfer an asset is as follows:
(This transaction represents the transfer of an EY asset of `100000000`neu by way of an account to the receiving `program` (which is a one-to-one correspondence with `address`) `0014a3f9111f3b0ee96cbd119a3ea5c60058f506fb19`, where the `control_program ` type indicates that `program` is used as the receiving method).

```
{
  "base_transaction": null,
  "actions": [
    {
      "account_id": "0ER7MEFGG0A02",
      "amount": 120000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "type": "spend_account"
    },
    {
      "amount": 100000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "control_program": "0014a3f9111f3b0ee96cbd119a3ea5c60058f506fb19",
      "type": "control_program"
    }
  ],
  "ttl": 0,
  "time_range": 0
}
```

Retire: `retire`

The source code for the `retireAction` structure is shown below:

```
type retireAction struct {
    bc.AssetAmount
    Arbitrary json.HexBytes `json:"arbitrary"`
}

type AssetAmount struct {
    AssetId *AssetID `protobuf:"bytes,1,opt,name=asset_id,json=assetId"
json:"asset_id,omitempty"`
    Amount  uint64   `protobuf:"varint,2,opt,name=amount" json:"amount,omitempty"`
}
```

The structure fields are described below:

- `AssetAmount` Indicates the ID of the asset destroyed and the corresponding asset count

- `Arbitrary` Indicates arbitrary additional information (hexadecimal string data), can be empty

The `JSON` format for `retireAction` is:

```
{
  "amount": 900000000,
  "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
  "arbitrary": "77656c636f6d65efbc8ce6aca2e8bf8ee69da5e588b0e58e9fe5ad90e4b896e7958c",
  "type": "retire"
}
```

For example, a sample transaction for the destruction of an asset is shown below:
(This transaction represents the destruction of the EY assets of `1`neu by means of an account and adds additional information, where `retire' represents the destruction of a specified quantity of assets).

```
{
  "base_transaction": null,
  "actions": [
    {
      "account_id": "0ER7MEFGG0A02",
      "amount": 900000000,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "type": "spend_account"
    },
    {
      "amount": 1,
      "asset_id": "ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff",
      "arbitrary":
"77656c636f6d65efbc8ce6aca2e8bf8ee69da5e588b0e58e9fe5ad90e4b896e7958c",
      "type": "retire"
    }
  ],
  "ttl": 0,
  "time_range": 0
}
```

Once the `build-transaction` input has been constructed, the transaction can be sent via an http call, and the json result returned after a successful build-transaction request is as follows:

```
{
  "allow_additional_actions": false,
  "fee": 20000000,
  "raw_transaction":
"070100020161015f1190c60818b4aff485c865113c802942f29ce09088cae1b117fc4c8db2292212fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffff8099c4d599010001160014a86c8
3ee12e6d790fb388345cc2e2b87056a077301000161015fb018097c4040c8dd86d95611a13c24f90d4c9d9
d06b25f5c9ed0556ac8abd73442275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c1
```

```
23f80a094a58d1d0101160014068840e56af74038571f223b1c99f1b60caaf456010003013effffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffff80bfffcb9901011600140b946646626c5
5a52a325c8bb48de792284d9b7200013e42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab
71a568c123f9d9f94a58d1d01160014c8b4391bab4923a83b955170d24ee4ca5b6ec3fb00013942275aacb
eda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f6301160014366b275ed9b2266b645cf
1b8be51009cc3b260e100",
  "signing_instructions": [
    {
      "position": 0,
      "witness_components": [
        {
          "keys": [
            {
              "derivation_path": [
                "010100000000000000",
                "0100000000000000"
              ],
              "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
            }
          ],
          "quorum": 1,
          "signatures": null,
          "type": "raw_tx_signature"
        },
        {
          "type": "data",
          "value": "d174db6506e35f2decb5be148c2984bfd0f6c67f043365bf642d1af387c04fd5"
        }
      ]
    },
    {
      "position": 1,
      "witness_components": [
        {
          "keys": [
            {
              "derivation_path": [
                "010100000000000000",
                "0800000000000000"
              ],
              "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
            }
          ],
          "quorum": 1,
          "signatures": null,
          "type": "raw_tx_signature"
        },
        {
          "type": "data",
          "value": "05cdbcc705f07ad87521835bbba226ad7b430cc24e5e3f008edbe61540535419"
        }
      ]
    }
  ]
}
```

The source code of the corresponding response object is as follows:

```go
// Template represents a partially- or fully-signed transaction.
type Template struct {
    Transaction         *types.Tx             `json:"raw_transaction"`
    SigningInstructions []*SigningInstruction `json:"signing_instructions"`

    // AllowAdditional affects whether Sign commits to the tx sighash or
    // to individual details of the tx so far. When true, signatures
    // commit to tx details, and new details may be added but existing
    // ones cannot be changed. When false, signatures commit to the tx
    // as a whole, and any change to the tx invalidates the signature.
    AllowAdditional bool `json:"allow_additional_actions"`
}
```

The structure fields are described below:

- `Transaction` Transaction related information, this field contains `TxData` and `bc.Tx` parts:
    - `TxData` The part of the transaction data that is displayed to the user and is visible to the user.
        - `Version` The transaction version
        - `SerialisedSize` The size of the transaction after serialisation.
        - `TimeRange` The maximum timestamp (block height) at which the transaction will be committed to the chain (after the main chain block height reaches this timestamp (block height), if the transaction is not committed to the chain, the transaction will be invalidated)
        - `Inputs` Transaction inputs
        - `Outputs` Transaction outputs
    - `bc.Tx` indicates the conversion structure used to process transactions in the system, this part is not visible to the user and is not described in detail.
- `SigningInstructions` Signing information for the transaction.
    - `Position` The position of the signature on the `input action`.
    - `WitnessComponents` The data information needed to sign the `input action`, where `signatures` of the build transaction is `null`, which means there is no signature; if the transaction is signed successfully, the signature information will exist in this field. This field is an interface interface and contains 3 different types:

    - `SignatureWitness` hashes the contract program at the transaction `input action` position in the transaction template `Template`, and then signs the hash value.
        - `signatures` (array type) signatures for the transaction, the value exists only after the `sign-transaction` has been executed
        - `keys` (array type) contains the master public key `xpub` and the derivation path `derivation_path`, through which the corresponding derivation private key `child_xprv` can be found during the signing phase, and then the derivation private key can be used to sign the transaction.
        - The number of `quorum` account `keys` must be equal to the length of `keys` above. If `quorum` is equal to 1, it is a single-signature account, otherwise it is a multi-signature account.
        - `program` The data part of the signature, the hash value of `program` is used as the signature data. If `program` is empty, a hash is generated based on the current transaction ID and the InputID of the corresponding action location, and then a `program` is automatically constructed from them as the command data.
    - `RawTxSigWitness` hashes the transaction ID of the transaction template `Template` and the InputID of the corresponding `input action` location (which is located in bc.Tx), and then signs the hash value.
        - `signatures` (array type) signatures for the transaction, the value exists only after the `sign-transaction` has completed execution
        - `keys` (array type) contains the master public key `xpub` and the derivation path `derivation_path`, through which the corresponding derivation private key `child_xprv` can be found during the signing phase, and then the derivation private key can be used to sign the transaction.
        - The number of `quorum` account `keys` must be equal to the length of `keys` above. If `quorum` is equal to 1, it is a single-signature account, otherwise it is a multi-signature account.
    - `DataWitness` This type does not require a signature, but verifies additional data on the contract program.
- `AllowAdditional` If `true`, additional data will be added to the transaction, but will not affect the executed `program` script of the transaction, and will have no effect on the signature result; if `false`, the whole transaction is signed as a whole, and any data changes will affect the signature of the whole transaction. Signature of the whole transaction

Introduction to Estimate Transaction Gas

The `estimate-transaction-gas` interface is an estimate of the handling fee for the result of a `build-transaction`. The estimated total handling fee `total_neu` needs to be added to the request structure of the `build-transaction`, and then the transaction is signed and submitted. The main flow is as follows:

```
build - estimate - build - sign - submit
```

The input request json format for estimating the handling fee is as follows:

```
{
  "transaction_template": {
    "allow_additional_actions": false,
    "fee": 20000000,
    "raw_transaction":
"0701000201 61015f1190c60818b4aff485c865113c802942f29ce09088cae1b117fc4c8db2292212fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8099c4d599010001160014a86c8
3ee12e6d790fb388345cc2e2b87056a07730100016101 5fb018097c4040c8dd86d95611a13c24f90d4c9d9
d06b25f5c9ed0556ac8abd73442275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c1
23f80a094a58d1d0101160014068840e56af74038571f223b1c99f1b60caaf456010003013effffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffff80bfffcb9901011600140b946646626c5
5a52a325c8bb48de792284d9b7200013e42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab
71a568c123f9d9f94a58d1d01160014c8b4391bab4923a83b955170d24ee4ca5b6ec3fb00013942275aacb
eda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f6301160014366b275ed9b2266b645cf
1b8be51009cc3b260e100",
    "signing_instructions": [
      {
        "position": 0,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0100000000000000"
                ],
                "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
              }
            ],
            "quorum": 1,
            "signatures": null,
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"d174db6506e35f2decb5be148c2984bfd0f6c67f043365bf642d1af387c04fd5"
          }
        ]
      },
      {
        "position": 1,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0800000000000000"
                ],
                "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
              }
            ],
            "quorum": 1,
            "signatures": null,
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"05cdbcc705f07ad87521835bbba226ad7b430cc24e5e3f008edbe61540535419"
          }
        ]
      }
    ]
  }
}
```

The source code of the corresponding response object is as follows:

```
type request struct{
    TxTemplate txbuilder.Template `json:"transaction_template"`
}

// Template represents a partially- or fully-signed transaction.
type Template struct {
    Transaction         *types.Tx             `json:"raw_transaction"`
    SigningInstructions []*SigningInstruction `json:"signing_instructions"`

    // AllowAdditional affects whether Sign commits to the tx sighash or
    // to individual details of the tx so far. When true, signatures
    // commit to tx details, and new details may be added but existing
    // ones cannot be changed. When false, signatures commit to the tx
    // as a whole, and any change to the tx invalidates the signature.
    AllowAdditional bool `json:"allow_additional_actions"`
}
```

The `TxTemplate` fields are described in the build-transaction result description.

The json result returned after a successful call to the `estimate-transaction-gas` interface is as follows:

```
{
  "total_neu": 5000000,
  "storage_neu": 3840000,
  "vm_neu": 1419000
}
```

The source code of the corresponding response object is as follows:

```
// EstimateTxGasResp estimate transaction consumed gas
type EstimateTxGasResp struct {
    TotalNeu   int64 `json:"total_neu"`
    StorageNeu int64 `json:"storage_neu"`
    VMNeu      int64 `json:"vm_neu"`
}
```

The structure fields are described below:

- `TotalNeu` The estimated total handling fee (in neu), this value is added directly to the EY asset input action of build-transaction
- `StorageNeu` The handling fee for the storage transaction.
- `VMNeu` The fee for running the virtual machine.

Step 2: Sign Transaction

API interface `sign-transaction`, code core/api/hsm.go#L53

The input request json format of the signature transaction is as follows:

```
{
  "password": "123456",
  "transaction": {
    "allow_additional_actions": false,
    "fee": 20000000,
    "raw_transaction":
"070100020161015f1190c60818b4aff485c865113c802942f29ce09088cae1b117fc4c8db2292212fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8099c4d59901000116001 4a86c8
3ee12e6d790fb388345cc2e2b87056a077301000161015fb018097c4040c8dd86d95611a13c24f90d4c9d9
d06b25f5c9ed0556ac8abd73442275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c1
23f80a094a58d1d0101160014068840e56af74038571f223b1c99f1b60caaf456010003013effffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffff80bfffcb9901011600140b946646626c5
5a52a325c8bb48de792284d9b7200013e42275aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab
71a568c123f9d9f94a58d1d01160014c8b4391bab4923a83b955170d24ee4ca5b6ec3fb00013942275aacb
eda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f6301160014366b275ed9b2266b645cf
1b8be51009cc3b260e100",
    "signing_instructions": [
      {
        "position": 0,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0100000000000000"
                ],
                "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
              }
            ],
            "quorum": 1,
            "signatures": null,
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"d174db6506e35f2decb5be148c2984bfd0f6c67f043365bf642d1af387c04fd5"
          }
        ]
      },
      {
        "position": 1,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0800000000000000"
                ],
                "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
              }
            ],
            "quorum": 1,
            "signatures": null,
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"05cdbcc705f07ad87521835bbba226ad7b430cc24e5e3f008edbe61540535419"
          }
        ]
      }
    ]
  }
```

```
}
```

The source code of the corresponding request object is as follows:

```
type SignRequest struct {    //function pseudohsmSignTemplates request
    Password string            `json:"password"`
    Txs      txbuilder.Template `json:"transaction"`
}
```

The structure fields are described below:

- Password The password for signing, based on which the user's private key can be parsed from the node server, and then the transaction can be signed with the private key
- Txs Transaction template, the result of build-transaction, structure type is txbuilder.Template, the related fields are described in the result of build-transaction.

The json result returned after a successful sign-transaction request is as follows:

```
{
  "sign_complete": true,
  "transaction": {
    "allow_additional_actions": false,
    "fee": 20000000,
    "raw_transaction":
"0701000201610 15f1190c60818b4aff485c865113c802942f29ce09088cae1b117fc4c8db2292212fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8099c4d599010001160014a86c8
3ee12e6d790fb388345cc2e2b87056a0773630240273d5fc4fb06909fbc2968ea91c411fd20f690c88e742
84ce27320524001299485385 62fe432afd6cf17e590e8645b80edf80b9d9581d0a980d5f9f859e3880620d
174db6506e35f2decb5be148c2984bfd0f6c67f043365bf642d1af387c04fd50161015fb018097c4040c8d
d86d95611a13c24f90d4c9d9d06b25f5c9ed0556ac8abd73442275aacbeda1522cd41580f875c3c452daf5
174b17ba062bf0ab71a568c123f80a094a58d1d0101160014068840e56af74038571f223b1c99f1b60caaf
4566302400cf0beefceaf9fbf1efadedeff7aee5b38ee7a25a20d78b630b01613bc2f8c9230555a6e09aaa
11a82ba68c0fc9e98a47c852dfe3de851d93f9b2b7ce256f90d2005cdbcc705f07ad87521835bbba226ad7
b430cc24e5e3f008edbe6154053541903013effffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffff80bfffcb9901011600140b946646626c55a52a325c8bb48de792284d9b7200013e42275
aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f9d9f94a58d1d01160014c8b4391
bab4923a83b955170d24ee4ca5b6ec3fb00013942275aacbeda1522cd41580f875c3c452daf5174b17ba06
2bf0ab71a568c123f6301160014366b275ed9b2266b645cf1b8be51009cc3b260e100",
    "signing_instructions": [
      {
        "position": 0,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0100000000000000"
                ],
                "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
              }
            ],
            "quorum": 1,
            "signatures": [

"273d5fc4fb06909fbc2968ea91c411fd20f690c88e74284ce2732052400129948538562fe432afd6cf17e
590e8645b80edf80b9d9581d0a980d5f9f859e38806"
            ],
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"d174db6506e35f2decb5be148c2984bfd0f6c67f043365bf642d1af387c04fd5"
          }
        ]
      },
      {
        "position": 1,
        "witness_components": [
          {
            "keys": [
              {
                "derivation_path": [
                  "010100000000000000",
                  "0800000000000000"
                ],
                "xpub":
"de0db655c091b2838ccb6cddb675779b0a9a4204b122e61699b339867dd10eb0dbdc926882ff6dd75c099
c181c60d63eab0033a4b0a4d0a8c78079e39d7ad1d8"
              }
            ],
            "quorum": 1,
            "signatures": [

"0cf0beefceaf9fbf1efadedeff7aee5b38ee7a25a20d78b630b01613bc2f8c9230555a6e09aaa11a82ba6
8c0fc9e98a47c852dfe3de851d93f9b2b7ce256f90d"
            ],
            "type": "raw_tx_signature"
          },
          {
            "type": "data",
            "value":
"05cdbcc705f07ad87521835bbba226ad7b430cc24e5e3f008edbe61540535419"
          }
        ]
      }
```

```
    ]
  }
}
```

The source code of the corresponding response object is as follows:

```go
type signResp struct {
    Tx           *txbuilder.Template `json:"transaction"`
    SignComplete bool                `json:"sign_complete"`
}
```

The structure fields are described below:

- `Tx` Transaction template after signing `txbuilder.Template`, if the signature is successful then `signatures` will be changed from null to the value of the signature, and the length of `raw_transaction` will be longer, because the `bc.Tx` part adds the parameter information for verifying the signature.
- `SignComplete` Signature completion flag, if it is `true` means the signature is complete, otherwise it is `false` means the signature is not complete, if it is a single signature, it may be a wrong password for the signature; if it is multiple signatures, it may be a need for other signatures. In case of signature failure, you just need to re-sign the signed transaction data with the correct password, no need to build the transaction again with `build-transaction`.

Step 3: Submit Transaction

API interface `submit-transaction`, code core/api/transact.go#L132

The input request json format for submit-transaction is as follows:

```
{
  "raw_transaction":
"070100020161015f1190c60818b4aff485c865113c802942f29ce09088cae1b117fc4c8db2292212fffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffff8099c4d599010001160014a86c8
3ee12e6d790fb388345cc2e2b87056a0773630240273d5fc4fb06909fbc2968ea91c411fd20f690c88e742
84ce27320524001299485385 62fe432afd6cf17e590e8645b80edf80b9d9581d0a980d5f9f859e3880620d
174db6506e35f2decb5be148c2984bfd0f6c67f043365bf642d1af387c04fd50161015fb018097c4040c8d
d86d95611a13c24f90d4c9d9d06b25f5c9ed0556ac8abd73442275aacbeda1522cd41580f875c3c452daf5
174b17ba062bf0ab71a568c123f80a094a58d1d0101160014068840e56af74038571f223b1c99f1b60caaf
4566302400cf0beefceaf9fbf1efadedeff7aee5b38ee7a25a20d78b630b01613bc2f8c9230555a6e09aaa
11a82ba68c0fc9e98a47c852dfe3de851d93f9b2b7ce256f90d2005cdbcc705f07ad87521835bbba226ad7
b430cc24e5e3f008edbe6154053541903013efffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffff80bfffcb9901011600140b946646626c55a52a325c8bb48de792284d9b7200013e42275
aacbeda1522cd41580f875c3c452daf5174b17ba062bf0ab71a568c123f9d9f94a58d1d01160014c8b4391
bab4923a83b955170d24ee4ca5b6ec3fb00013942275aacbeda1522cd41580f875c3c452daf5174b17ba06
2bf0ab71a568c123f6301160014366b275ed9b2266b645cf1b8be51009cc3b260e100"
}
```

The request object corresponding to the source code is as follows:

```
type SubmitRequest struct { //function submit request
    Tx types.Tx `json: "raw_transaction"`
}
```

The struct fields are described below:

- `Tx` Information about the transaction after the signature is complete. Note that the `raw_transaction` in this field is not the full return result of the signature transaction `sign-transaction`, but rather the `raw_transaction` field in the `transaction` in the return result of the signature transaction.

The json result returned after a successful `sign-transaction` request is as follows:

```
{
  "tx_id": "2c0624a7d251c29d4d1ad14297c69919214e78d995affd57e73fbf84ece361cd"
}
```

The response object corresponding to the source code is as follows:

```
type submitTxResp struct {
    TxID *bc.Hash `json: "tx_id"`
}
```

The structure fields are described below:

- `TxID` The transaction ID, which is displayed when the transaction has been submitted to the pool, otherwise the transaction fails.

`UTXO` User Own Management Model

This section is for users to manage their own private keys and addresses, and to build and send transactions via `utxo`.

The following steps as well as functional transformation is for reference only, the specific code implementation needs to be debugged by the user according to the actual situation, you can refer to the unit test case code core/blockchain/txbuilder/txbuilder_test.go#L252

Step 1: Create Private and Public Keys

This part of the function can refer to the code crypto/ed25519/chainkd/util.go#L11, you can create master private key and master public key by `NewXKeys(nil)`.

```go
func NewXKeys(r io.Reader) (xprv XPrv, xpub XPub, err error) {
    xprv, err = NewXPrv(r)
    if err != nil {
        return
    }
    return xprv, xprv.XPub(), nil
}
```

Step 2: Create a Receive Object Based on the Public Key.

Receiving object contains two forms: `address` form and `program` form, both are one-to-one correspondence, either one can be. Which create a single signature address reference code account/accounts.go#L253 for the corresponding transformation for:

```go
func (m *Manager) createP2PKH(xpub chainkd.XPub) (*CtrlProgram, error) {
    pubKey := xpub.PublicKey()
    pubHash := crypto.Ripemd160(pubKey)

    // TODO: pass different params due to config
    address, err := common.NewAddressWitnessPubKeyHash(pubHash, &consensus
.ActiveNetParams)
    if err != nil {
        return nil, err
    }

    control, err := vmutil.P2WPKHProgram([]byte(pubHash))
    if err != nil {
        return nil, err
    }

    return &CtrlProgram{
        Address:        address.EncodeAddress(),
        ControlProgram: control,
    }, nil
}
```

Create multi-signature address reference code accounts/accounts.go#L276 to be transformed accordingly as follows: (quorum means the is the number of verifications required for multi-signature address, for example, 3-2 multi-signature address, means 3 master public keys, two signatures are required to verify the pass)

```go
func (m *Manager) createP2SH(xpubs []chainkd.XPub, quorum int) (*CtrlProgram, error) {
    derivedPKs := chainkd.XPubKeys(xpubs)
    signScript, err := vmutil.P2SPMultiSigProgram(derivedPKs, quorum)
    if err != nil {
        return nil, err
    }
    scriptHash := crypto.Sha256(signScript)

    // TODO: pass different params due to config
    address, err := common.NewAddressWitnessScriptHash(scriptHash, &consensus
.ActiveNetParams)
    if err != nil {
        return nil, err
    }

    control, err := vmutil.P2WSHProgram(scriptHash)
    if err != nil {
        return nil, err
    }

    return &CtrlProgram{
        Address:        address.EncodeAddress(),
        ControlProgram: control,
    }, nil
}
```

Step 3: Finding Spendable `UTXO`

Finding the spendable utxo is really about finding the receiving address or receiving `program` that is your own `unspend_output`. Where the structure of utxo is:

```
// UTXO describes an individual account utxo.
type UTXO struct {
    OutputID bc.Hash
    SourceID bc.Hash

    // Avoiding AssetAmount here so that new(utxo) doesn't produce an
    // AssetAmount with a nil AssetId.
    AssetID bc.AssetID
    Amount  uint64

    SourcePos      uint64
    ControlProgram []byte

    AccountID           string
    Address             string
    ControlProgramIndex uint64
    ValidHeight         uint64
    Change              bool
}
```

The related fields involving utxo constructed transactions are described below:

- `SourceID` The mux_id of the previous associated transaction, based on which the output of the previous transaction can be located

- AssetID The asset ID of the utxo.
- Amount The number of assets in the utxo.
- SourcePos The position of the utxo in the output of the previous transaction.
- ControlProgram The receiving program of the utxo.
- Address The receiving address of the utxo.

Information about these fields of the utxo above can be found in the transaction that the get-block interface returns the result of, and its related structure is as follows: (refer to the code core/api/block_retrieve.go#L29)

```
// BlockTx is the tx struct for getBlock func
type BlockTx struct {
    ID         bc.Hash                    `json:"id"`
    Version    uint64                     `json:"version"`
    Size       uint64                     `json:"size"`
    TimeRange  uint64                     `json:"time_range"`
    Inputs     []*query.AnnotatedInput    `json:"inputs"`
    Outputs    []*query.AnnotatedOutput   `json:"outputs"`
    StatusFail bool                       `json:"status_fail"`
    MuxID      bc.Hash                    `json:"mux_id"`
}

//AnnotatedOutput means an annotated transaction output.
type AnnotatedOutput struct {
    Type            string             `json:"type"`
    OutputID        bc.Hash            `json:"id"`
    TransactionID   *bc.Hash           `json:"transaction_id,omitempty"`
    Position        int                `json:"position"`
    AssetID         bc.AssetID         `json:"asset_id"`
    AssetAlias      string             `json:"asset_alias,omitempty"`
    AssetDefinition *json.RawMessage   `json:"asset_definition,omitempty"`
    Amount          uint64             `json:"amount"`
    AccountID       string             `json:"account_id,omitempty"`
    AccountAlias    string             `json:"account_alias,omitempty"`
    ControlProgram  chainjson.HexBytes `json:"control_program"`
    Address         string             `json:"address,omitempty"`
}
```

The correspondence between UTXO and get-block return result fields is as follows:

```
SourceID       `json:"mux_id"`
AssetID        `json:"asset_id"`
Amount         `json:"amount"`
SourcePos      `json:"position"`
ControlProgram `json:"control_program"`
Address        `json:"address,omitempty"`
```

Step4: Constructing Transactions Via UTXO

Constructing a transaction via utxo means spending the specified utxo using send_account_unspent_output.

The first step is to construct the transaction input TxInput and the data information needed for signing SigningInstruction through utxo, this part of the function can refer to the code core/account/builder.go#L305 can be modified accordingly as follows.

```
// UtxoToInputs convert an utxo to the txinput
func UtxoToInputs(xpubs []chainkd.XPub, quorum int, u *UTXO) (*types.TxInput,
*txbuilder.SigningInstruction, error) {
    txInput := types.NewSpendInput(nil, u.SourceID, u.AssetID, u.Amount, u.SourcePos,
u.ControlProgram)
    sigInst := &txbuilder.SigningInstruction{}

    if u.Address == "" {
        return txInput, sigInst, nil
    }

    address, err := common.DecodeAddress(u.Address, &consensus.ActiveNetParams)
    if err != nil {
        return nil, nil, err
    }

  sigInst.AddRawWitnessKeys(xpubs, nil, quorum)
    switch address.(type) {
    case *common.AddressWitnessPubKeyHash:
        derivedPK := xpubs[0].PublicKey()
        sigInst.WitnessComponents = append(sigInst.WitnessComponents, txbuilder
.DataWitness([]byte(derivedPK)))

    case *common.AddressWitnessScriptHash:
        derivedPKs := chainkd.XPubKeys(xpubs)
        script, err := vmutil.P2SPMultiSigProgram(derivedPKs, quorum)
        if err != nil {
            return nil, nil, err
        }
        sigInst.WitnessComponents = append(sigInst.WitnessComponents, txbuilder
.DataWitness(script))

    default:
        return nil, nil, errors.New("unsupport address type")
    }

    return txInput, sigInst, nil
}
```

The second step is to construct the transaction output TxOutput by UTXO
This part of the function can be found in the code core/protocol/bc/types/txoutput.go#L20.

```go
// NewTxOutput create a new output struct
func NewTxOutput(assetID bc.AssetID, amount uint64, controlProgram []byte) *TxOutput {
    return &TxOutput{
        AssetVersion: 1,
        OutputCommitment: OutputCommitment{
            AssetAmount: bc.AssetAmount{
                AssetId: &assetID,
                Amount:  amount,
            },
            VMVersion:      1,
            ControlProgram: controlProgram,
        },
    }
}
```

Step 5: Combine the Input and Output of a Transaction to Form a Transaction Template

Construct a transaction `txbuilder.Template` from the transaction information already generated above, the function of this part can refer to core/blockchain/txbuilder/builder.go#L96 to transform it to.

```go
type InputAndSigInst struct {
    input *types.TxInput
    sigInst *SigningInstruction
}

// Build build transactions with template
func BuildTx(inputs []InputAndSigInst, outputs []*types.TxOutput) (*Template, *types
.TxData, error) {
    tpl := &Template{}
    tx := &types.TxData{}
    // Add all the built outputs.
    tx.Outputs = append(tx.Outputs, outputs...)

    // Add all the built inputs and their corresponding signing instructions.
    for p, in := range inputs {
        // Empty signature arrays should be serialized as empty arrays, not null.
        in.sigInst.Position = uint32(p)
        if in.sigInst.WitnessComponents == nil {
            in.sigInst.WitnessComponents = []witnessComponent{}
        }
        tpl.SigningInstructions = append(tpl.SigningInstructions, in.sigInst)
        tx.Inputs = append(tx.Inputs, in.input)
    }

    tpl.Transaction = types.NewTx(*tx)
    return tpl, tx, nil
}
```

Step 6: Signing the Constructed Transaction

The account model is based on the password to find the corresponding private key to sign the transaction, here the user can directly use the private key to sign the transaction, you can refer to the signature code core/blockchain/txbuilder/txbuilder.go#L88 is transformed into: (the following transformation only supports single-signature transactions, multi-signature transactions users can refer to the example for transformation)

```go
// Sign will try to sign all the witness
func Sign(tpl *Template, xprv chainkd.XPrv) error {
    for i, sigInst := range tpl.SigningInstructions {
        for _, wc := range sigInst.WitnessComponents {
            switch sw := wc.(type) {
            case *RawTxSigWitness:
                h := tpl.Hash(uint32(i)).Byte32()
                sig := xprv.Sign(h[:])
                sw.Sigs = append(sw.Sigs, sig)
            }
        }
    }
    return materializeWitnesses(tpl)
}
```

The multi-signature approach can be seen in the following modification: (xprvs needs to be the same as the number of signatures Quorum, also note the order of the multi-signatures)

```go
func Sign(tpl *Template, xprvs []chainkd.XPrv) error {
    for i, sigInst := range tpl.SigningInstructions {
        for _, wc := range sigInst.WitnessComponents {
            switch sw := wc.(type) {
            case *RawTxSigWitness:
                h := tpl.Hash(uint32(i)).Byte32()
                for k, xprv := range xprvs {
                    if len(sw.Sigs[k]) > 0 {
                        // Already have a signature for this key
                        continue
                    }
                    sig := xprv.Sign(h[:])
                    sw.Sigs[k] = sig
                    break  // the one private sign this tx only once
                }
            }
        }
    }
    return materializeWitnesses(tpl)
}
```

Step 7: Submit Transaction for Uploading

There is no need to change anything in this step, just refer to the `API` submit-transaction.