Qora API V13

Resources

Qora

Provides general information about the state of the application and the ability to close the application.

GET qora/stop

Will stop the application. This command might not be able to return a http OK message.

GET qora/status

Returns the status of the application

Status	Value
0	No connections
1	Synchronizing
2	Oke

GET qora/isuptodate

Returns a boolean that shows if the application is synchronized with the network.

Seed

To generate a random base58 encoded seed. These seeds can be used to create a wallet or to import an account.

GET seed/{length}

Returns a base58 encoded random seed of 32 bytes. Use the optional parameter length to request a seed of {length} bytes.

Peers

GET peers

Returns an array of all the IP's of the peers to which the application is currently connected.

Transactions

GET transactions

Returns an array of your accounts each with their 50 last transactions.

Format

```
[
  {
        "transactions":[],
    "account":"Qdxn4qW8kiPUiBnBSy9mbqMGBrBHRhK2JM"
  },
  {
        "transactions": [],
    "account":"QQNFGuE8iZZ3sHjCnvvdbhfQHXokU8SgCX"
  }
]
```

Errors

Error	Description
201	Wallet does not exist.

GET transactions/limit/{limit}

Returns an array of your accounts each with their {limit} last transactions.

Format

```
[
  {
        "transactions":[],
    "account":"Qdxn4qW8kiPUiBnBSy9mbqMGBrBHRhK2JM"
  },
  {
        "transactions": [],
    "account":"QQNFGuE8iZZ3sHjCnvvdbhfQHXokU8SgCX"
  }
]
```

Errors

Error	Description
201	Wallet does not exist.

GET transactions/{address} (DEPRECATED)

Returns an array of the last 50 transactions of a specific address in your wallet.
Will be removed in future versions, use GET transactions/address/{address} instead.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	address does not exist in wallet

GET transactions/address/{address}

Returns an array of the last 50 transactions of a specific address in your wallet.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	address does not exist in wallet

GET transactions/address/{address}/limit/{limit}

Returns an array of the last {limit} transactions of a specific address in your wallet.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	address does not exist in wallet

GET transactions/signature/{signature}

Returns the transaction that matches the given signature.
Will be moved to GET transactions/{signature} once GET transactions/{address} is removed.

Errors

Error	Description
101	Invalid signature.
311	Transaction does not exist.

GET transactions/network

Returns an array of all the unconfirmed transactions known to the client.

POST transactions/scan (NEW)

Returns all the transactions that match the filters. All filters are optional but please limit that amount of transactions or blocks to scan to avoid running into issues.
Return the last block it scanned, the amount of blocks it scanned and the scanned transactions.

Filter	Description
start	The signature of the starting block.
blocklimit	The maximum amount of blocks to scan.
transactionlimit	The maximum amount of transactions to return.
type	Only return transactions with the given type.
service	Only return Arbitrary Transactions with the given service.
address	Only return transactions where the given address is involved.

Format


```
{
  "blocklimit": 1000,
  "transactionlimit": 100,
  "type": 2
}
```


Response


```
{
  "amount":303,
"lastscanned":"HQzVY265bpf2pSuLK8P5PRWQN47Ui5PiqyFsgyz2WpRiu6xUaLyyj99yrkUt2xSMbexRoF5fqJijJwG9
DGJjPVYDwrhgSa33vg2KxvehAXRdhXvegzjnW55Fn1NXy51Ei3D8A9CyW7N4ohzFdHwmGjNryM26RPnqaTmmrZA
32HeX7uc"
  "transactions":[
    {
      "fee":"1.00000000",
      "timestamp":1400254833419,
      "sender":"Qec5ueWc4rcBrty47GZfFSqvLymxvcycFm",
      "amount":"1.00000000",
      "confirmations":1959,
      "type":2,

"reference":"38sGerf4a24fRCTeknzQJgGBJe6hKribjKZGpQmrqajJwDM216FoUm9VCve7tst4Dypn1qgHKVgb6qN4v
K9QFv3p",

"signature":"2kioUSPPZdGvqWYZuT61J5M9nQ3udSzS2AZGU3MhdtNdbS2naWfuD6cpR2T9ZpKjRs2GFsyEYHDnt
QbBmTdsCyZL",
      "recipient":"QWNNYAh4dD7gktCfN9hb454qEaitjEnfy5"
    },
    {
      "fee":"1.00000000",
      "timestamp":1400258674297,
      "sender":"QVjcFWE6TnGePGJEtbNc1thwD2sgHBLvUV",
      "amount":"42940527.25000000",
      "confirmations":1949,
      "type":2,

"reference":"kRMLekdCQY7Prq2nyL8XLuL77oAJKP8WNarq1GaU6CTLQZ8VHgZJQFAeqJrNeKpt52QgsYrawcscRnc
Y1XEsePB",

"signature":"4YEbzGrvKntTJbfX3v4AZSY1JUaUqw7LFRm1s4ZDZKJqxfMWtKUw2Ho1jXXE16FSqwU4GqP8dxHaCV8
huA6xDSg5",
      "recipient":"Qd9jQKZSXoYgFypTQySJUSbXcZvjgdiemn"
    }]
}
```

Errors

Error	Description
1	Json error.
102	Invalid address.
101	Invalid signature.
301	Block does not exist.

Blocks

GET blocks

Returns an array of the 50 last blocks generated by your accounts.

Errors

Error	Description
201	Wallet does not exist.

GET blocks/address/{address}

Returns an array of the 50 last blocks generated by a specific address in your wallet.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	Address does not exist in wallet.

GET blocks/{signature}

Returns the block that matches the given signature.

Errors

Error	Description
101	Invalid signature.
301	Block does not exist.

GET blocks/first

Returns the genesis block.

GET blocks/last

Returns the last valid block.

GET blocks/child/{signature}

Returns the child block of the block that matches the given signature.

Errors

Error	Description
101	Invalid signature.
301	Block does not exist.

GET blocks/generatingbalance

Calculates the generating balance of the block that will follow the last block.

GET blocks/generatingbalance/{signature}

Calculates the generating balance of the block that will follow the block that matches the signature.

Errors

Error	Description
101	Invalid signature.
301	Block does not exist.

GET blocks/time

Calculates the time it should take for the network to generate the next block.

GET blocks/time/{generatingbalance}

Calculates the time it should take for the network to generate blocks when the current generating balance in the network is the specified generating balance.

GET blocks/height

Returns the block height of the last block.

GET blocks/height/{signature}

Returns the block height of the block that matches the given signature.

Errors

Error	Description
101	Invalid signature.
301	Block does not exist.

Addresses

GET addresses

Returns an array of all the addresses in your wallet.

Errors

Error	Description
201	Wallet does not exist.

GET addresses/validate/{address}

Validates the given address.
Returns true/false.

GET addresses/seed/{address}

Returns the 32-byte long base58-encoded account seed of the given address.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	Address does not exist in wallet.
203	Wallet is locked.

GET addresses/new

Generates a new account and returns the newly generated address.

Errors

Error	Description
201	Wallet does not exist.
203	Wallet is locked.

POST addresses

Imports the given 32-byte long base58-encoded account seed.
Returns the address when successfully imported.

Errors

Error	Description
103	Invalid seed.
201	Wallet does not exist.
203	Wallet is locked.

DELETE addresses/{address}

Deletes the given address.
Returns true/false.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
203	Wallet is locked.

GET addresses/generatingbalance/{address}

Return the generating balance of the given address.

GET addresses/balance/{address}

Returns the confirmed balance of the given address.

Errors

Error	Description
102	Invalid address.

GET addresses/balance/{address}/{confirmation}

Calculates the balance of the given address after the given confirmations.
0 confirmations can only be used on addresses that exist in your wallet.

Errors

Error	Description
102	Invalid address.

POST addresses/sign/{address}

Signs the given message using the given address.

Format


```
{
  "message":"test",
  "publickey":"6cWtyccawscvHhE5woPaLbDUc6qFaH7b7YuDJFrBvgJ3",
"signature":"2XuAEoUG2GmWJ8s5ZMZMK7csQ1nfHcqL5JYm3JBqetUAZKeT9mu7mSKYYMjLQoLBr5DqLCfaKXLQJ
nbzCLYCfC21"
}
```


Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	Address does not exist in wallet.
203	Wallet is locked.

POST addresses/verify/{address}

Verifies if the given message was signed by the given address. Returns true/false.

Format


```
{
  "message":"test",
  "publickey":"6cWtyccawscvHhE5woPaLbDUc6qFaH7b7YuDJFrBvgJ3",
"signature":"2XuAEoUG2GmWJ8s5ZMZMK7csQ1nfHcqL5JYm3JBqetUAZKeT9mu7mSKYYMjLQoLBr5DqLCfaKXLQJ
nbzCLYCfC21"
}
```


Errors

Error	Description
101	Invalid signature
102	Invalid address.
112	Invalid public key.

Wallet

GET wallet

Returns general information about the wallet.

Format

```
{
  "exists":true,
  "isunlocked":false
}
```

GET wallet/seed

Return the 32-byte long base58-encoded wallet seed.

Errors

Error	Description
201	Wallet does not exist.
203	Wallet is locked.

GET wallet/synchronize

Rescans the blockchain for data.

Errors

Error	Description
201	Wallet does not exist.

GET wallet/lock

Locks the wallet.
Returns true/false depending on the fact if the wallet was already locked and if the password was correct.

Errors

Error	Description
201	Wallet does not exist.

POST wallet

Creates a wallet using the given 32-byte long base58-encoded seed, password,recover flag and amount.

Format

```
{
  "seed":"FQgbSAm6swGbtqA3NE8PttijPhT4N3Ufh4bHFAkyVnQz",
  "password":"cookies",
  "recover":false,
  "amount":10
}
```

Errors

Error	Description
1	Json error.
103	Invalid seed.
104	Invalid amount.
204	Wallet already exists.

POST wallet/unlock

Unlocks the wallet using the given password.
Returns true/false depending on the fact if the password is correct.

Errors

Error	Description
201	Wallet does not exist.

Payment

POST payment

Send a new payment using the given data.
Returns the transaction in JSON when successful.

Format

```
{
  "amount":"10.05",
  "fee":"1.000001",
  "sender":"Qdxn4qW8kiPUiBnBSy9mbqMGBrBHRhK2JM",
  "recipient":"QhMaXFowsVqdAhvU2xkcLzuVaH5VDyEWsS"
}
```

Errors

Error	Description
1	Json error.
104	Invalid amount.
105	Invalid fee.
106	Invalid sender.
107	Invalid recipient.
201	Wallet does not exist.
203	Wallet is locked.

Names

GET names

Returns an array of all the names owned by your accounts.

GET names/address/{address}

Returns an array of all the names owned by a specific address in your wallet.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	Address does not exist in wallet.

GET names/{name}

Returns details about the given name

Format


```
{
  "name":"qora",
  "value":"http://qora.org",
  "owner":"QVeHoptRAeLj5DqGq2TKHVL4w51KFGS5R5"
}
```


Errors

Error	Description
401	Name does not exist.

POST names

Register a new name.
Returns the transaction in JSON when successful.

Format

```
{
  "name":"qora",
  "value":"http://qora.org",
  "registrant":"QVeHoptRAeLj5DqGq2TKHVL4w51KFGS5R5",
  "fee":"1.00001"
}
```

Errors

Error	Description
1	Json error.
2	Not enough balance.
102	Invalid address.
105	Invalid fee.
108	Invalid name length.
109	Invalid value length.
201	Wallet does not exist.
203	Wallet is locked.
402	Name already exists.
404	Name must be lower case.

POST names/{name}

Updates an existing name.
Returns the transaction in JSON when successful.

Format

```
{
  "newvalue":"http://qora.net",
  "newowner":"QVeHoptRAeLj5DqGq2TKHVL4w51KFGS5R5",
  "fee":"1.00001"
}
```

Errors

Error	Description
1	Json error.
2	Not enough balance.
102	Invalid address.
105	Invalid fee.
108	Invalid name length.
109	Invalid value length.
201	Wallet does not exist.
203	Wallet is locked.
401	Name does not exist.
403	Name already for sale.

NameSales

GET namesales

Returns an array of all the namesales owned by your accounts.

Errors

Error	Description
201	Wallet does not exist.

GET namesales/address/{address}

Returns an array of all the namesales owned by a specific address in your wallet.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	Address does not exist in wallet.

GET namesales/{name}

Return details about the given name that is for sale.

Errors

Error	Description
410	Name is not for sale.

GET namesales/network

Returns an array of all the names that are for sale.
For performance this array only contains the keys of the names that are for sale and not the details.

Format

```
[
  "qora",
  "test"
]
```

POST namesales/{name}

Used to sell the given name.
Returns the transaction in JSON when successful.

Format

```
{
  "amount":"100”,
  "fee":"1.00001"
}
```

Errors

Error	Description
1	Json error.
2	Not enough balance.
102	Invalid address.
104	Invalid amount.
105	Invalid fee.
108	Invalid name length.
109	Invalid name owner.
201	Wallet does not exist.
203	Wallet is locked.
401	Name does not exist.
403	Name already for sale.

DELETE namesales/{name}/{fee}

Used to cancel the sale of the given name.
Returns the transaction in JSON when successful.

Errors

Error	Description
1	Json error.
2	Not enough balance.
102	Invalid address.
105	Invalid fee.
108	Invalid name length.
110	Invalid name owner.
201	Wallet does not exist.
203	Wallet is locked.
401	Name does not exist.
410	Name is not for sale.

POST namesales/buy/{name}

Used to purchase the given name.
Returns the transaction in JSON when successful.

Format

```
{
  "buyer":"QVeHoptRAeLj5DqGq2TKHVL4w51KFGS5R5",
  "fee":"1.00001"
}
```

Errors

Error	Description
1	Json error.
2	Not enough balance.
102	Invalid address.
105	Invalid fee.
108	Invalid name length.
111	Invalid buyer.
201	Wallet does not exist.
203	Wallet is locked.
401	Name does not exist.
410	Name is not for sale.
411	Buyer is already the owner.

Polls

GET polls

Returns an array of all the polls created by your accounts.

Errors

Error	Description
201	Wallet does not exist.

GET polls/address/{address}

Returns an array of all the polls owned by a specific address in your wallet.

Errors

Error	Description
102	Invalid address.
201	Wallet does not exist.
202	Address does not exist in wallet.

GET polls/{name}

Return details about the poll with the given name.

Errors

Error	Description
501	Poll does not exist.

GET polls/network

Returns an array of all the polls.
For performance this array only contains the names of the polls and not the details.

Format

```
[
  "qora",
  "test"
]
```

POST polls

Used to create a new poll.
Returns the transaction in JSON when successful.

Format

```
{
  "creator":"QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6",
  "name":"testpoll",
  "description":"this is a testpoll",
  "options": [
                        "option one",
                        "option two"
            ],
  "fee":"1.00001"
}
```

Errors

Error	Description
1	Json error.
2	Not enough balance.
3	Not yet released.
102	Invalid address.
105	Invalid fee.
108	Invalid name length.
109	Invalid description length.
113	Invalid options length.
114	Invalid option length.
201	Wallet does not exist.
202	Address does not exist in wallet.
203	Wallet is locked.
404	Name must be lowercase.
502	Poll already exists.
503	Duplicate option.

POST polls/vote/{name}

Used to vote on a poll with the given name.
Returns the transaction in JSON when successful.

Format

```
{
  "voter":"QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6",
  "option": "option one",
  "fee":"1.00001"
}
```

Errors

Error	Description
1	Json error.
2	Not enough balance.
3	Not yet released.
102	Invalid address.
105	Invalid fee.
108	Invalid name length.
114	Invalid option length.
201	Wallet does not exist.
202	Address does not exist in wallet.
203	Wallet is locked.
404	Name must be lowercase.
501	Poll does not exist.
504	Polloption does not exist.
505	Already voted for that option.

ArbitraryTransactions (NEW)

POST arbitrarytransactions

Used to send an arbitrary transaction.
The data of the arbitrary transaction must be base58 encoded and must be between 1-4000 bytes.
Returns the transaction in JSON when successful.

Format


```
{
  "creator":"QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6",
"data":"4GFHMAo9fmbUq7usopgntwUfAiLtpL98K6QCosAJsqQmY95tfd5KoUaKu34v6Qwp7RtYEhobCx7LVi7aYbbtpzfA",
  "service": 555,
  "fee":"1.00001"
}
```


Errors

Error	Description
1	Json error.
2	Not enough balance.
3	Not yet released.
102	Invalid address.
105	Invalid fee.
115	Invalid data.
116	Invalid data length.
201	Wallet does not exist.
202	Address does not exist in wallet.
203	Wallet is locked.

Response objects

Block


```
{
  "fee":"4.00000000",
  "transactions":[ ],
  "timestamp":1399319724713,
"generatorSignature":"3k7jpRRCJNexhf8cdR7w1HAD7ppmo7DK2wzXBekFmKpwfVmZF5SiM9q8b5MwjCHtHmyoBTSXbq9iTodGxpf2qeri",
  "generatingBalance":855786957,
  "generator":"QUDPJRGS8EreTvZWMDs5imyp3rAqU9hCPK",
"reference":"HnhRcmrXp13dZwtP7T35Pzaav278pm8CDJmWRZStVpGVkzs6Bo4VnrbZ8XrouokReEKLGLZfLsbmodmRrhiJgkd3b67vDjgVMtKcR9WRRom8zHsE6FGvRgTv8pivBb3VrYDryECj96bpXgMqxPtZdmQUQHTUM1BA81XtUQukw5svw3K",
"transactionsSignature":"2pE2sn7jGHruhHzv4gWmr1sdgTXCBv8nvAZKsHDaCJETLVPX7XYhaYPSfwr5pz6WsAC99QRhPQihBeDyWDDqLxQP",
"signature":"F3Q5ihLGC2iFcS4RmRjbqSV85KWr13mXvSKDQCRBrBjG5h5n8hfd2sDWfLDSTiCrDhahojNmRQGKfwza57XFKs1vKFgK4xqQEAw7BjnEEAvvUWHndgE56LnaMDnfyGNfMGB3wfHNdQa8DQWoMptWbmUYKACzKyFEt2ssH6brknESgVM",
  "version":1
}
```


Transactions

Type 1: Genesis transaction


```
{
        "fee":"0",
        "timestamp":1399139274713,
        "amount":"500000000.00000000",
        "type":1,
        "reference":"1",
        "confirmations":0,
"signature":"4GFHMAo9fmbUq7usopgntwUfAiLtpL98K6QCosAJsqQmY95tfd5KoUaKu34v6Qwp7RtYEhobCx7LVi7aYbbtpzfA",
        "recipient":"QQNFGuE8iZZ3sHjCnvvdbhfQHXokU8SgCX"
}
```

Type 2: Payment transaction


```
{
        "fee":"1.00000000",
        "sender":"QRsraeFA9xiD3qVWiLSbxdcW2goUeGFVnF",
        "amount":"10.00000000",
        "timestamp":1399656640390,
          "type":2,
"confirmations":0,
"reference":"keFP1SfTn4WVXgew1qUH6G8xmRrLaaSju65Ni36fhcSBMny8tMZjHc8BWdGMbjRRXmzNpBmi7JFwG
1fPCMcvdkq",
"signature":"wXrNfiDTamahSrMKNg96XAaAxojD5Nxm8gKPhM1qp2DzSMHvuaek5kzEQfHvzZe1AtxoQkbc5ELoEL
5F4sXKNnt",
        "recipient":"QhMaXFowsVqdAhvU2xkcLzuVaH5VDyEWsS"
}
```


Type 3: Register name transaction


```
{
        "fee":"1.00000000",
        "timestamp":1399314215363,
        "registrant":"Qdxn4qW8kiPUiBnBSy9mbqMGBrBHRhK2JM",
        "name":"qora",
        "owner":"Qdxn4qW8kiPUiBnBSy9mbqMGBrBHRhK2JM",
        "value":"qora",
          "type":3,
"confirmations":0,
"reference":"5gE3vbzwDUbkR9YUem8RHVV3HcswrCX2ej9bA5MbJyaMrhQGXkFKqnvEtgq3s1vK3LizFEzCLz2HxJtd
gULjMJRr",

"signature":"5a1tyxSzX57sV6cqneyhTtWqV11pZPFsPqhTEPVqkNNFrrs1uCnG4DD3bUHar5PfJsF39YShxNCJYs1tZs
FFtJoX"
}
```

Type 4: Update name transaction


```
{
        "fee":"1.00000000",
        "newValue":"a",
        "timestamp":1399656876336,
        "newOwner":"QNpfdoKjU3r3PDjYStQqKtjQ2CaxQWmzjZ",
        "name":"qora",
          "owner":"QVeHoptRAeLj5DqGq2TKHVL4w51KFGS5R5",
          "type":4,
"confirmations":0,
"reference":"42npsTYYydk798VwtJg4a5JR2g39FC2ASEHPhdYr4x2jq6eLw1au2mjc2gxmvghsPojEmhaEreksj174rw4Uthbg",
"signature":"5s2fqKKnAa8cVkYVng3aF2SqxkKE1ArWKyUwBsTRz5tqRP6RfQXzcA6SNbaXNgGH4T62oh9QaXoB4xbASemHRSGV"
}
```


Type 5: Sell name transaction


```
{
        "fee":"1.00000000",
          "amount":"500.0",
        "timestamp":1399314220865,
        "name":"qora",
        "owner":"Qdxn4qW8kiPUiBnBSy9mbqMGBrBHRhK2JM",
          "type":5,
"confirmations":0,
"reference":"5a1tyxSzX57sV6cqneyhTtWqV11pZPFsPqhTEPVqkNNFrrs1uCnG4DD3bUHar5PfJsF39YShxNCJYs1tZsFFtJoX",
"signature":"4L6XLyPCrAFChdzqmTCrGDp43yPiZk2Bn5Ch3YUUrGsXA8NuGo8dp7xQjsMNtRVSAtGdbXxsngTWmfM4ETVdTr6o"
}
```

Type 6: Cancel namesale transaction


```
{
        "fee":"1.00000000",
        "timestamp":1399657084324,
          "name":"qora",
        "owner":"QUDPJRGS8EreTvZWMDs5imyp3rAqU9hCPK",
          "type":6,
"confirmations":0,
"reference":"2yLvx23qZjBwFkyWFj6ScowYRBi15rzCwEvx1wDYpKonS9P7s13eZLCrC1tvau3uAfZKPFFdBvK9Utnb53hEjWJy",
"signature":"4uNmBZ5aKkApMWvvAKKEpM3CBKCXPq1b3G615UmXfH3cMfVrFFZUURGCbnkc6n9ZfdFkuk6hxpJtZAnMz1KZstB6"
}
```


Type 7: Buy name transaction


```
{
        "fee":"1.00000000",
          "amount":"500.0",
        "timestamp":1399314260133,
        "name":"a",
        "buyer":"QVeHoptRAeLj5DqGq2TKHVL4w51KFGS5R5",
          "type":7,
          "confirmations":0,
          "reference":"PSG56nzDowqhYxzg2mmRXKcgxg8cBtyjEoTF37gXyCy4s3jXE6ygoRv81Sg7wTdHJ7xtwHD3uauWAEYKYGaxM7D",
"signature":"42npsTYYydk798VwtJg4a5JR2g39FC2ASEHPhdYr4x2jq6eLw1au2mjc2gxmvghsPojEmhaEreksj174rw4Uthbg"
}
```


Type 8: Create Poll transaction


```
{
  "fee":"1.00001000",
  "timestamp":1403552417900,
  "confirmations":0,
  "description":"this is a testpoll",
  "name":"testpoll",
  "type":8,

"reference":"3cXi7YQGihx1q75P1YMcgeGnqHY2PGbzpScasMy4UTZ1brTCFuQVKWvDW8Ywj7c3BDh7aLKBoHft79mGZkkzs7bq",
  "signature":"ofuYyAALXzR7x6Yhfh8HXkjDuqt2x5Ao9NP9QCtyRpmrd6A8CAXkb1GLQrntDA9zq",
  "creator":"QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6",
  "options":[
    "option one",
    "option two"
  ]
}
```

Type 9: Vote on poll transaction


```
{
  "fee":"1.00001000",
  "timestamp":1403553008161,
  "poll":"testpoll",
  "confirmations":0,
  "type":9,
"reference":"ofuYyAALXzR7x6Yhfh8HXkjDuz1t7XdZeM446ejPunjF5iEqt2x5Ao9NP9QCtyRpmrd6A8CAXkb1GLQrntDA9zq",
  "signature":"2kErpXY5EToBRk3coHVbnk5P4j5WixukDqedWhTJnCXsVFKy7samMta6QF7sgrcNbdk",
  "option":0,
  "creator":"QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6"
}
```


Type 10: Arbitrary transaction (NEW)


```
{
  "fee":"1.00001000",
  "timestamp":1403553008161,
  "confirmations":0,
  "type":10,
"reference":"ofuYyAALXzR7x6Yhfh8HXkjDuz1t7XdZeM446ejPunjF5iEqt2x5Ao9NP9QCtyRpmrd6A8CAXkb1GLQrntDA9zq",
  "signature":"2kErpXY5EToBRk3coHVbnk5P4j5WixukDqedWhTJnCXsVFKy7samMta6QF7sgrcNbdk",
   "creator":"QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6",
   "service": 5555,
"data":"3cXi7YQGihx1q75P1YMcgeGnqHY2PGbzpScasMy4UTZ1brTCFuQVKWvDW8Ywj7c3BDh7aLKBoHft79mGZkkzs7bq"
}
```

Name


```
{
    "name":"qora",
    "value":"qora",
    "owner":"QUDPJRGS8EreTvZWMDs5imyp3rAqU9hCPK"
}
```


NameSale


```
{
            "amount":"500.0",
            "name":"qora",
    "seller":"QUDPJRGS8EreTvZWMDs5imyp3rAqU9hCPK"
}
```


Poll


```
{
  "creator":"QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6",
  "description":"this is a testpoll",
  "name":"testpoll",
  "options":[
    {
      "name":"option one",
      "votes":"93393168.31001000",
      "voters":[
        "QNbA69dbnmwqJHLQeS9v63hSLZXXGkmtC6",
        "QgpH3K3ArkQTg15xjKqGq3BRgE3aNH9Q2P"
      ]
    },
    {
      "name":"option two",
      "votes":"8772354.53899000",
      "voters":[
        "QRyECqW54ywKVt4kZTEXyRY17aaFUaxzc4"
      ]
    }
  ]
}
```

Errors

When an error happens the API will return a HTTP message 400(bad request) combined with an error.

```
{
        "error" 101:,
        "message":"invalid signature"
}
```

Error	Description
0	Unknown error.
1	Json error.
2	Not enough balance.
3	Not yet released.
101	Invalid signature.
102	Invalid address.
103	Invalid seed.
104	Invalid amount.
105	Invalid fee.
106	Invalid sender.
107	Invalid recipient.
108	Invalid name length.
109	Invalid value length.
110	Invalid name owner.
111	Invalid buyer.
112	Invalid public key.
113	Invalid options length.
114	Invalid option length.
115	Invalid data.

116	Invalid data length.
201	Wallet does not exist.
202	Address does not exist in wallet.
203	Wallet is locked
204	Wallet already exists.
301	Block does not exist.
311	Transaction does not exist.
401	Name does not exist.
402	Name already exists.
403	Name already for sale.
404	Name must be lower case.
410	Name is not for sale.
411	Buyer is already the owner.
501	Poll does not exist.
502	Poll already exists.
503	Duplicate option.
504	Polloption does not exist.
505	Already voted for that option.