

블록체인의 혁신 심버스!

설계

Transaction

❖ Types of transaction

- Original transaction
 - General transaction: Ethereum convention
 - Smart contract transaction: Ethereum convention
- SCT transaction: Symverse Contract Template
- Deposit transaction: Managing deposit

❖ Transaction data

```
params:
  [{
    "from": <address>      //[10]byte, sender address
    "to": <address>        //[10]byte, receiver address or contract address or nil
    "gas": <int>            //[int, gas amount for executing transaction
    "gasPrice": <int>       //[int, gas price per gas unit
    "value": <int>          //[int, the value sent with this transaction or amount of deposit
    "nonce": <int>          //[int, the count of transaction publication in the account
    "type": <int>           //[int, transaction type (0: original(default), 1: sct, 2: deposit)
    "workNodes": []<address> //array, list of work nodes what deliver the transaction (1 <= count <= 3)
    "term": <int>           //[int, deposit term in case of deposit type transaction (0: unlimited, >0: term from current block)
    "input": <data>         //[[]byte, rlp encoded data (contract, sct, and deposit term)
  ]}
```

※ Note

- 'type' decides the usage of transaction. If this is not specified, the transaction will be original transaction by default.
 - type '0': Original transaction (default)
 - type '1': SCT transaction
 - type '2': Deposit transaction
- 'workNodes' should be included in the transaction by wallet before it is signed. If this is not specified, it will be set from symbase of node by default.
- 'term' is only valid as API parameter for deposit set transaction, and it is not included in signing.
- Some fields – type, gas, gasPrice, value, nonce, and workNodes will be default value, when it is not specified.

❖ Type '0' – Original transaction

- When 'type' is zero(0), it indicates original transaction compatible with Ethereum. If 'type' is not specified, the transaction will be original transaction by default.
- If 'to' is nil, it is smart contract creation. Otherwise it is general transaction or contract call.
- 'input' is mandatory for contract creation or contract call.
- In case of contract call, 'to' should be contract address.

General transaction

```
params:
[[
  "from": <address> //[[10]byte, sender address
  "to": <address> //[[10]byte, receiver address
  "gas": <int> //int, gas amount for executing transaction
  "gasPrice": <int> //int, gas price per gas unit
  "value": <int> //int, the value sent with this transaction
  "nonce": <int> //int, the count of transaction publication in the account
  "type": <int> //int, transaction type (0: original(default), 1: sct, 2: deposit)
  "workNodes": []<address> //array, list of work nodes what deliver the transaction (1 <= count <= 3)
  "term": <int> //int, deposit term in case of deposit type transaction
  "input": <data> //[]byte, rlp encoded data
]]
```

Contract transaction

```
params:
[[
  "from": <address> //[[10]byte, sender address
  "to": <address> //[[10]byte, contract address or nil
  "gas": <int> //int, gas amount for executing transaction
  "gasPrice": <int> //int, gas price per gas unit
  "value": <int> //int, the value sent with this transaction
  "nonce": <int> //int, the count of transaction publication in the account
  "type": <int> //int, transaction type (0: original(default), 1: sct, 2: deposit)
  "workNodes": []<address> //array, list of work nodes what deliver the transaction (1 <= count <= 3)
  "term": <int> //int, deposit term in case of deposit type transaction
  "input": <data> //[]byte, rlp encoded data (contract)
]]
```

Example)

```
// General transaction
sym.sendTransaction({from: "0x00021000000000010002", to: "0x00021000000000070002", workNodes: ["0x00021000000000010002", "0x00021000000000080002", "0x00021000000000090002"], value: web3.toHug(100, "sym")})

// Contract creation
sym.sendTransaction({from: "0x00021000000000010002", workNodes: ["0x00021000000000010002", "0x00021000000000080002", "0x00021000000000090002"],
input: "0xf8418080f83d9a30783533373936643736363537323733363534333666363936658830783533353934648b39313834653732613030308c307830303030303030303031", gas: 150000})

// Contract call
sym.sendTransaction({from: "0x00021000000000010002", to: "0x0CED1024Eed02B234df2", workNodes: ["0x00021000000000010002", "0x00021000000000080002", "0x00021000000000090002"], input:
"0xf8418080f83d9a30783533373936643736363537323733363534333666363936658830783533353934648b39313834653732613030308c307830303030303030303031", gas: 150000})

// Raw transaction
sym.sendRawTransaction("0xf8738a000000000000000000901850430e2340083015f908a00000000000000000908002cb8a00000000000000000901a068c19c97383288faa6373c8b058ed386753c767a3e4976937b2afca1515df8
75a0142de5cf2687167da8d13f51a4767536ea1473b6d46f76edfa644b04aa428901")

// Check balance
sym.getBalance("0x00021000000000010002")
```


❖ Type '2' – Deposit transaction

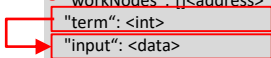
- When 'type' is 2, it indicates deposit operation.
- If 'to' is nil, it is deposit creation. Otherwise it is deposit restoration.
- 'from' and 'to' should be same for deposit restoration transaction. In the case of exception, the deposit manager account can restore the deposit of personal account, then 'from' should be deposit manager's account and 'to' should be personal account.
- 'value' is amount of deposit to be set
- 'term' is only valid for deposit creation, and used for setting deposit term. This is encoded with rlp and inserted to 'input' field during sending transaction, but it's not included in signing. Note that 'term' is the parameter only for convenient usage.

```

params:
[[
"from": <address>           //[10]byte, sender address
"to": <address>              //[10]byte, receiver address or nil
"gas": <int>                 //[int, gas amount for executing transaction
"gasPrice": <int>           //[int, gas price per gas unit
"value": <int>               //[int, amount of deposit
"nonce": <int>               //[int, the count of transaction publication in the account
"type": <int>                //[int, transaction type (0: original(default), 1: sct, 2: deposit)
"workNodes": []<address>    //[array, list of work nodes what deliver the transaction (1 <= count <= 3)
"term": <int>                //[int, deposit term in case of deposit type transaction (0: unlimited, >0: term from current block)
"input": <data>              //[[]byte, rlp encoded data (deposit term) when deposit creation
]]

```

rlp encoding



Example)

```

// Deposit creation
sym.sendTransaction({from: "0x0002A000000000010002", type: "0x2", value: web3.toHug(20, "sym"), term: 100})
sym.sendTransaction({from: "0x0002A000000000010002", type: "0x2", value: web3.toHug(20, "sym"), input: "0x8207d0"})
// Deposit restoration
sym.sendTransaction({from: "0x0002A000000000010002", to: "0x0002A000000000010002", type: "0x2"}) //self restore
sym.sendTransaction({from: "0x00020000000000020002", to: "0x0002A000000000010002", type: "0x2"}) //force restore by deposit manager

// Raw transaction
sym.sendRawTransaction("0xf8758a000000000000000000901850430e2340083015f90808901158e460913d00000838207d002cb8a00000000000000000980a0dceb7d07d0ba181f8d918a2b61cf9e55f8cac1b19dc94729f56dd7210e0a4b9aa0511c581b649ce6748586f176be60042b6b020fbc22615c5248c14ef54c97fc05")

// Check deposit
sym.getDeposit("0x0002A000000000010002")

```