Merchant Coin (MRC) Token API

Token Name: MERC Token Token Symbol: MRC

The smallest unit of MERC Token is 1 MRC (equal to 1 Satoshi). The decimal of MERC Token is 0.

Token Contract: 0x1Fb7F56FCAC7934dc303D38f8B1f6b3209C1A69e

host: https://bsearchau.accziom.com

port: 8886

1. Register

Before user use the MERC token system, he/she should register to MERC token system. Registering does not exploit private key of user but user generate new key to use in Layer2. User should register his/her address of Layer1 and new private key of Layer2. for example, user can use sha1(private key) to generate new key of Layer2. Since function sha1() is irreversible, attacker can not get private key of Layer 1 from private key of Layer2. Therefore, this function is **secure**.

> Endpoints: GET https://bsearchau.accziom.com:8886/register
>
> Parameters:
>
> - address: address of Layer1 for account.
> - layer2PrivKey: the private key of Layer2
>
> Response: One user can register only once. If user try to register twice or more, it returns false. If otherwise, returns true.
>
> Example:

```
$.ajax({
        type: "GET",
        url: "https://bsearchau.accziom.com:8886/deposit",
        dataType: "json",
        data: {
            "address": "0x...",
            "layer2PrivKey": "..." },
        success: function(result){
            // if success, returns true and if failed, returns false.
        }
});
```

2. Deposit

2.1. Deposit from Accziom Server

After Accziom Server successfully transfered fees to main account of MERC Token, this function will be called. To guarantee sercurity of deposit transaction, all parameters are encrypted with a private key.

> Endpoints: GET https://bsearchau.accziom.com:8886/deposit
>
> Parameters:
>
> - param: encrpyted parameters after decrypt this parameter, the following parameters are obtained.
>
>   ```
>   address: address of Layer1 for account.
>   fee: the amount of deposit fee. The value should be integer and be greater than 0.
>   ```
>
> Response: If success, returns true and if otherwise, returns false.

```
// Accziom Server Code
// function call_encrypt_deposit should be called after confirmed that transfer had been done successfully.

const request = require('request');
const rsa = require('./rsa_encrypt');

async function httprequest(url, requestData, method="GET"){
    return new Promise((resolve, reject)=>{
        var option ={
          url: url,
          method: method,
          json: true,
          headers: {
              "content-type": "application/x-www-form-urlencoded",
          },
          form: requestData
        }
        request(option, function(error, response, body) {
          if (!error && response.statusCode == 200) {
              resolve(body);
          } else {
              resolve({});
          }
        });
    });
};

async function call_encrpyt_deposit(hosturl, address, fee) {
    var param = {
```

```
        address: address,
        fee: fee
    };
    var encrypted_param = rsa.encrypt(JSON.stringify(param));
    return await httprequest(hosturl + "/encrypted_deposit", {
        param: encrypted_param
    })
}
```

2.2. Getting MERC Token information

To transfer to main account of MERC token, Accziom server has to know the address of the main account and the address of MERC token contract. Using the following api, these info can be obtained.

> Endpoints: GET https://bsearchau.accziom.com:8886/merc_token
>
> Response: MERC token information with JSON format. contract: the address of MERC token contract ownerAddress: the address of main account of MERC token
>
> Example:

```
$.ajax({
        type: "GET",
        url: "https://bsearchau.accziom.com:8886/merc_token",
        dataType: "json",
        data: {},
        success: function(result){
            // process response
        }
});
```

- Response:

```
{
    contract: "0x1Fb7F56FCAC7934dc303D38f8B1f6b3209C1A69e",
    ownerAddress: "0x5AB7966568BB67c58a8c12EE6fB1858249bE1a36",
}
```

2.3. Getting ACCZIOM NFT Token information

To create a ACCZIOM NFT, Accziom server has to know the address of the ACCZIOM NFT contract. Using the following api, these info can be obtained.

> Endpoints: GET https://bsearchau.accziom.com:8886/accziom_token
>
> Response: ACCZIOM NFT token information with JSON format. contract: the address of ACCZIOM NFT token contract
>
> Example:

```
$.ajax({
        type: "GET",
        url: "https://bsearchau.accziom.com:8886/accziom_token",
        dataType: "json",
        data: {},
        success: function(result){
            // process response
        }
});
```

- Response:

```
{
    contract: "0x7E7BdA04ca478F30b2D2e106d47a6D6Cb4858b9C",
}
```

3. Request

Before users carry out any transaction with Layer2, they should verify that they own the correct private key of Layer2.

All Layer2-based Transactions are done through following steps:

First, user should request a certain transaction, and then server records the transaction request and returns a certain seed string.

Second, user should encode the seed string with their private key of Layer2 and send the encoded string to server.

Third, the server verifies the encoded string and if success, then do the requested transaction.

This function is corresponding to first step. This function does not exploit any privacy of user and is **secure**.

The kind of transaction:

```
withdraw: transfers fees from Layer2 to Layer1.
balance: get balances of Layer1 and Layer2.
history: get transaction history of Layer2.
nft: get info of NFT that user owns.
info: all info including balance and transaction history and nft info.
spend: spend fees from Layer2.
transfer: transfers fees from one address to another address in Layer2.
```

> Endpoints: GET https://bsearchau.accziom.com:8886/request
>
> Parameters:
>
> - address: address of Layer1 for account.
> - fee: the amount of deposit fee.
> - type: the kind of transaction, i.e. "withdraw", "balance", "history", "nft", "info", "spend", "transfer"
> - target: address of receiver. This parameter is used only for "transfer" transaction
>
> Response: returns seed string. if returned seed is empty string, it indicates that inputed parameters may be not correct.
>
> Example:

```
$.ajax({
        type: "POST",
        url: "https://bsearchau.accziom.com:8886/request",
        dataType: "json",
        data: {
            "address": "0x...",
            "fee": 10000000,
            "type": "withdraw"},
        success: function(result){
            // process response
            // result: seed string
        }
});
```

4. Verify

Once user requested a certain transaction to server, he/she should send the correct encoded to prove that he/she own the correct private key. If verification success, requested transaction will be carried out.

> Endpoints: POST https://bsearchau.accziom.com:8886/verify
>
> Parameters:
>
> - address: address of user.

- seed: the string sended from server when user requested
- verification - the encodeded string of seed.
- type - (optional) the kind of transaction. it should be equal to request type.

Response: If verification successfully done, returns the corresponding result data of requested transaction.

withdraw Transaction: returns true if success, or false if otherwise.

spend Transaction: returns true if success, or false if otherwise.

transfer Transaction: returns true if success, or false if otherwise.

transferNFT Transaction: returns true if success, or false if otherwise.

balance Transaction:

```
    eth_bal: the balance of Ethereum

    layer1: the balance of Layer1 (MRC Token)

    layer2: the balance of Layer2
```

history Transaction: returns array of transaction. each item has following subitems:

```
    address: address of user

    fee: the amount of the transaction

    event: the kind of transaction. i.e. "deposit", "withdraw" or
"spend"

    time: timestamp of the transaction
```

nft Transaction: returns array of nft. each item has following subitems:

```
    Name: NFT Name

    URI: NFT Token URI

    Total Rewards: the total rewards that the NFT obtained.
```

info Transaction: returns entire info of user, including balance, history and nft

```
    balacnce: balance info, please reference __balance__ transaction.

    transaction: transaction history info, please reference __history__
  transaction

    nft: NFT info, please reference __nft__ transaction
```

Example:

```
$.ajax({
        type: "POST",
        url: "https://bsearchau.accziom.com:8886/verify",
        dataType: "json",
        data: {
            address: "0x...",
            seed: "...",
            verification: "..."
            },
        success: function(result){
            // process response
        }
});
```

5. Registering NFT

Endpoints: POST https://bsearchau.accziom.com:8886/registerNFT

Parameters:

- nftID: nft id in ACCZIOM NFT Token.

Response: If sucess, returns true and if failed, returns false.

6. WSS communication

Endpoints: POST wss://bsearchau.accziom.com:8887

Register: After client created websocket, client should send address of user to register.

```
  websocket.send(address)
```

Message: When status of user is changed, server sends the changed status To use the sended string from server, client should parse to JSON structure using JSON.parse() The json structure is as follows:

> balance: balances of Layer1 and Layer2. it is same as the result of __balance__ transaction
>
> transaction: transaction histories of Layer2. it is same as the result of __history__ transaction
>
> nft: NFT info. it is same as the result of __nft__ transaction