

Comunion Sprint01 文档

李泽辉 *

2020 年 7 月 5 日

Contents

1 登录鉴权流程	1
2 startup	3
3 setting	6
A 附录合约 ABI, 创建合约实例的时候需要使用	9
A.1 startup ABI	9
A.2 setting ABI	11

1 登录鉴权流程

本系统使用 metamask 第三方账号作为登录验证识别身份, 以此来保护 API 接口, 登录鉴权流程如 figure1所示

用户点击登陆后, 代码如 listing1

```
1  /**
2   * @description 点击连接按钮
3   */
4  async connectAccount() {
5    this.loading = true;
6    try {
7      await this.$store.dispatch('login');
8    } catch (error) {}
9    this.loading = false;
10 }
```

Listing 1: 参考文件 HeaderConnect.vue

*王帅 (后端), 王欣 (合约)

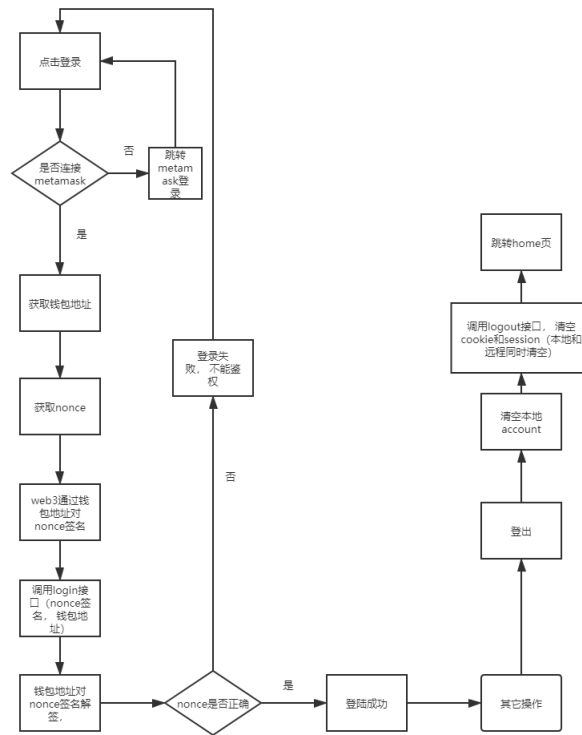


Figure 1: 登录鉴权

dispatch 转入 vuex 判断 metamask 插件是否被引入当前浏览器, 如果浏览器没有引入 metamask, 则跳转 metamask 官网下载页, 如果已安装, 则吊起 metamask 登录, 登录 metamask 后, 前端获取到用户钱包; 调用接口获取 nonce, 使用钱包地址对 nonce 签名, 发给后端, 后端使用钱包地址对 nonce 签名解签, 解签结果与后端传给前端的 nonce 一至, 则认为用户已经登陆了 communion 系统, 完成身份验证, 代码如 listing2

```

1  /**
2   * descriptions 用户登录
3   * @param dispatch
4   * @param commit
5   * @returns {Promise<boolean>}
6   */
7  async login({ dispatch, commit }) {
8    // 检查用户是否安装了metamask
9    if (!!window.ethereum && window.ethereum.isMetaMask) {
10     try {
11       // 调用登录metamask
12       const accounts = await ethereum.enable();
13       // 取第个账号1
14       const account = accounts[0].toLowerCase();
15       // 初始化web3
16       initWeb3();
17       // 获取nonce
18       const nonce = await getNonce(account);
19       // 获取当前连接网络的ID
20       const netWorkId = await web3.eth.net.getId();
21       commit('HANDLE_NEW_NETWORK', netWorkId + '');
    }
  }

```

```

22
23     if (nonce) {
24         // 对签名nonce
25         let signature = await web3.eth.personal.sign(nonce, account);
26         // 登录
27         const ret = await login({ publicKey: account, signature });
28         if (!ret) {
29             throw new Error('Error when login, please try again');
30         }
31         commit('SET_USER', ret);
32         // 注册事件metamask
33         // FIXME: 之前需要先onoff
34         ethereum.autoRefreshOnNetworkChange = true;
35         ethereum.on('chainIdChanged', arg => commit('HANDLE_NEW_CHAIN', arg));
36         ethereum.on('networkChanged', arg => commit('HANDLE_NEW_NETWORK', arg));
37         ethereum.on('accountsChanged', accounts => commit('UPDATE_ACCOUNT', accounts[0]));
38     } else {
39         throw new Error('Error when get nonce, please try again. ');
40     }
41     commit('UPDATE_ACCOUNT', account);
42
43     commit('UPDATE_LOGIN_TIME', new Date().getTime());
44
45     /** 不刷新网页的情况下， 个小时自动登出6 */
46     setTimeout(() => {
47         dispatch('logout');
48     }, 2 * 60 * 60 * 1000);
49 } catch (error) {
50     message.error(error?.message || 'Error occured. ');
51 }
52 return true;
53 } else {
54     // 打开官网metamask
55     window.open('https://metamask.io/');
56     message.warning('You have not installed metamask. ');
57 }
58 },

```

Listing 2: 参考文件 user.js

2 startup

startup 的创建是 Comunion 的第一步， 包括创建和上链两步， 点击创建 startup， 表单提交， 会根据提交的数据创建合约， 然后吊起 metamask， 等待签名确认后返回交易 hash， 签名请求发出后， 把交易 hash 和 startup 的数据一起提交给后端。 流程图如 fig2

Table 1: 上链状态表

	0	1	2	3
startup 状态 (字段: state)	无意义状态	待确认	上链成功	未确认到交易
setting 状态 (字段: settingState)	无意义状态	待确认	上链成功	未确认到交易

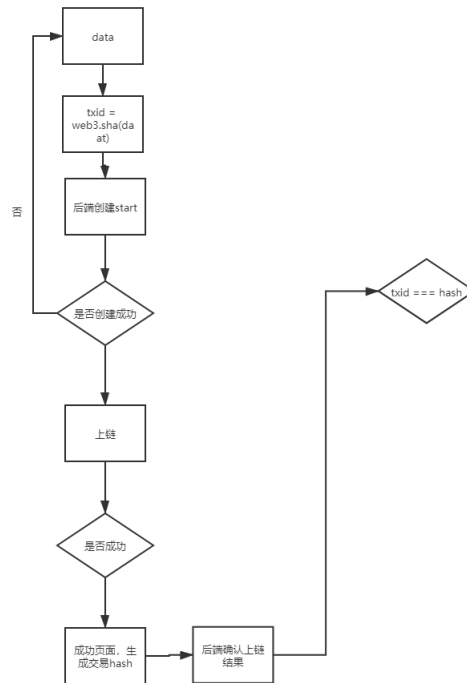


Figure 2: 上链的过程

后端会根据交易 hash 查询上链结果. 更新 startup 的状态¹, 状态表见 table1. 步骤如下:

1. 提交表单, 获取 startup 的 id, 根据合约 abi 创建合约实例
2. 发起交易
3. 交易发出后, 返回交易 hash
4. 提交 startup 表单数据和 hash
5. 后端根据 hash 查询上链结果

```

1 /**
2  * @description 提交表单
3  */
4  onSubmit() {
5    if (this.canNotTransaction) {
6      return;
7    }
8    this.$refs.ruleForm.validate(async valid => {

```

¹由于创建合约实例, 需要获取 startup 的 id, 但是创建 startup 需要合约发起交易后的交易 hash, 因此, startup 的 id 在还未创建前, 调用接口获取, 在创建合约实例和创建 startup 的时候, 都要携带 startup id

```

9      if (valid) {
10        try {
11          this.createState = 'creating';
12          const startupId = await getPrepareStartupId();
13          if (startupId) {
14            const id = startupId.id;
15            this.ethSendTransaction(this.form, id);
16          }
17        } catch (e) {
18          console.error(e);
19        }
20      }
21    });
22  },
23
24  /**
25   * @description 发起交易
26   * @param formData: 表单数据startup
27   * @param : startupId startup id
28   * @returns {Promise<void>}
29   */
30  async ethSendTransaction(formData, startupId) {
31    const contractStatpUp = await this.getContractInstance(formData, startupId);
32    const codeData = await contractStatpUp.encodeABI();
33    const countAll = await web3.eth.getTransactionCount(this.account, 'pending');
34    const chainId = await web3.eth.getChainId();
35
36    const tx = {
37      from: this.account,
38      to: COMUNION_RECEIVER_ACCOUNT,
39      data: codeData,
40      value: web3.utils.numberToHex(Math.pow(10, 17)),
41      nonce: web3.utils.numberToHex(countAll),
42      gasPrice: web3.utils.numberToHex(Math.pow(10, 9)),
43      gasLimit: web3.utils.numberToHex(183943),
44      chainId: chainId
45    };
46    window.ethereum.sendAsync(
47      {
48        method: 'eth_sendTransaction',
49        params: [tx],
50        from: window.ethereum.selectedAddress
51      },
52      (err, result) => {
53        if (err) {
54          return console.error(err);
55        }

```

```

56         const txid = result.result;
57         this.createStartup(formData, startupId, txid);
58     }
59 );
60 },
61
62 /**
63  * @description 构建hex, 生成txid
64  * @param formData
65  * @param startupId
66  * @param txid
67  */
68 async createStartup(formData, startupId, txid) {
69     try {
70         if (this.isEdit) {
71             // 更新
72             const startup = await updateStartup(this.$router.query.id, { ...formData, txid });
73             if (startup) {
74                 this.createState = 'succeeded';
75             }
76         } else {
77             // 后端创建startup
78             const startup = await createStartup({ ...formData, txid, id: startupId });
79             if (startup) {
80                 this.createState = 'succeeded';
81             }
82         }
83     } catch (e) {
84         console.log('%c\n e :::----->', 'font-size:30px;background: purple;', e);
85     }
86 },

```

Listing 3: 参考文件 New.vue

3 setting

setting 的设置, 是 comunion 的第二步, 点击 setting 后, 可以查看到, 刚才创建的 startup 状态, 如果此时后端通过交易 hash 在链上查询到了交易, 则 startup 的状态是 waitting setting, 点击这个 startup 就可以开始 setting 了。

setting 设置完成后, 提交数据, 会根据提交的数据创建合约, 然后吊起 metamask, 等待签名确认后返回交易 hash, 签名请求发出后, 把交易 hash 和 startup 的数据一起提交给后端, 后端会根据交易 hash 查询上链结果, 与 start up 的创建类似! 步骤如下:

1. 提交表单 setting 表单, 根据合约 abi 创建合约实例
2. 发起交易

3. 交易发出后，返回交易 hash
4. 提交 setting 表单数据和 hash, 后端创建 setting
5. 后端根据 hash 查询上链 setting 结果, 待后端确认到上链后，在 home 页面，可以看到完成后的自己创建的 comunion

```
1  async onOk() {
2      // save data
3      const body = {
4          ...this.form.finance,
5          ...{ ...this.form.governance }
6      };
7      // 时间转小时
8      body.voteMinDurationHours = body.minDuration.days * 24 + body.minDuration.hours;
9      body.voteMaxDurationHours = body.maxDuration.days * 24 + body.maxDuration.hours;
10     delete body.maxDuration;
11     delete body.minDuration;
12
13     body.voteTokenLimit = body.voteTokenLimit ? body.voteTokenLimit : -1;
14
15     const id = this.$route.params.id;
16     this.ethSendTransaction(body, id);
17     // 关闭loading
18     this.$refs.launch.loading = false;
19 },
20
21 /**
22  * @description 发起交易
23  * @param formData
24  * @param id
25  * @returns {Promise<void>}
26  */
27 async ethSendTransaction(formData, id) {
28     const contractStatpUp = await this.getContractInstance(formData, id);
29     const codeData = await contractStatpUp.encodeABI();
30     const countAll = await web3.eth.getTransactionCount(this.account, 'pending');
31     const chainId = await web3.eth.getChainId();
32
33     const tx = {
34         from: this.account,
35         to: COMMUNION_SETTING_RECEIVE_ACCOUNT,
36         data: codeData,
37         value: 0,
38         nonce: web3.utils.numberToHex(countAll),
39         gasPrice: web3.utils.numberToHex(Math.pow(10, 9)),
40         gasLimit: web3.utils.numberToHex(183943),
41         chainId: chainId
```

```

42     };
43     window.ethereum.sendAsync(
44         {
45             method: 'eth_sendTransaction',
46             params: [tx],
47             from: window.ethereum.selectedAddress
48         },
49         (err, result) => {
50             if (err) {
51                 return console.error(err);
52             }
53             const txid = result.result;
54             this.createSetting(formData, txid);
55         }
56     );
57 },
58
59 /**
60  * @description 获取合约实例
61  * @returns {Promise<*>}
62  */
63 async getContractInstance(formData, id) {
64     const data = JSON.parse(JSON.stringify(formData));
65     const contract = new web3.eth.Contract(settingAbi, COMMUNION_SETTING_RECEIVE_ACCOUNT);
66     const walletAddrs = data.walletAddrs.map(item => item.addr);
67     let voteAssignAddrs = [];
68     data.voteAssignAddrs.forEach(item => {
69         voteAssignAddrs.push(item || '0x0000000000000000000000000000000000000000');
70     });
71     /** 发起合约 */
72     const contractSetting = await contract.methods.newSetting(
73         id,
74         data.tokenName,
75         data.tokenSymbol,
76         data.tokenAddr,
77         walletAddrs,
78         data.voteType,
79         // POS
80         data.voteTokenLimit.toString(),
81         // Founder assign
82         voteAssignAddrs,
83         data.voteSupportPercent.toString(),
84         data.voteMinApprovalPercent.toString(),
85         data.voteMinDurationHours.toString(),
86         data.voteMaxDurationHours.toString()
87     );
88     return contractSetting;

```



```
89  },
```

Listing 4: 参考文件 SettingDetail.vue

A 附录合约 ABI，创建合约实例的时候需要使用

A.1 startup ABI

```
1  /**
2   * @description 合约的abi
3   */
4  export const startupAbi = [
5    {
6      inputs: [],
7      payable: false,
8      stateMutability: 'nonpayable',
9      type: 'constructor'
10   },
11   {
12     constant: false,
13     inputs: [
14       {
15         internalType: 'address payable',
16         name: 'addr',
17         type: 'address'
18       }
19     ],
20     name: 'setCoinBase',
21     outputs: [],
22     payable: false,
23     stateMutability: 'nonpayable',
24     type: 'function'
25   },
26   {
27     constant: false,
28     inputs: [
29       {
30         internalType: 'string',
31         name: 'id',
32         type: 'string'
33       },
34       {
35         internalType: 'string',
36         name: 'name',
37         type: 'string'
38       }
39     ],
```

```

39     {
40         internalType: 'string',
41         name: 'categoryId',
42         type: 'string'
43     },
44     {
45         internalType: 'string',
46         name: 'mission',
47         type: 'string'
48     },
49     {
50         internalType: 'string',
51         name: 'descriptionAddr',
52         type: 'string'
53     }
54 ],
55 name: 'newStartup',
56 outputs: [],
57 payable: true,
58 stateMutability: 'payable',
59 type: 'function'
60 },
61 {
62     constant: true,
63     inputs: [
64         {
65             internalType: 'string',
66             name: 'id',
67             type: 'string'
68         }
69     ],
70     name: 'getStartup',
71     outputs: [
72         {
73             internalType: 'string',
74             name: 'name',
75             type: 'string'
76         },
77         {
78             internalType: 'string',
79             name: 'categoryId',
80             type: 'string'
81         },
82         {
83             internalType: 'string',
84             name: 'mission',
85             type: 'string'

```

```

86     },
87     {
88         internalType: 'string',
89         name: 'descriptionAddr',
90         type: 'string'
91     }
92 ],
93 payable: false,
94 stateMutability: 'view',
95 type: 'function'
96 }
97 ];
98
99

```

Listing 5: (参考文件 startup.js)

A.2 setting ABI

```

1  /**
2  * @description setting 上链的api
3  * @type {}
4  */
5  export const settingAbi = [
6      {
7          inputs: [],
8          payable: false,
9          stateMutability: 'nonpayable',
10         type: 'constructor'
11     },
12     {
13         constant: false,
14         inputs: [
15             {
16                 name: 'id',
17                 type: 'string'
18             },
19             {
20                 name: 'tokenName',
21                 type: 'string'
22             },
23             {
24                 name: 'tokenSymbol',
25                 type: 'string'
26             },
27             {

```

```

28     name: 'tokenAddr',
29     type: 'string'
30 },
31 {
32     name: 'walletAddrs',
33     type: 'address[] '
34 },
35 {
36     name: 'voteType',
37     type: 'string'
38 },
39 {
40     name: 'voteTokenLimit',
41     type: 'string'
42 },
43 {
44     name: 'voteAssignAddrs',
45     type: 'address[] '
46 },
47 {
48     name: 'voteMSupportPercent',
49     type: 'string'
50 },
51 {
52     name: 'voteMinApprovalPercent',
53     type: 'string'
54 },
55 {
56     name: 'voteMinDurationHours',
57     type: 'string'
58 },
59 {
60     name: 'voteMaxDurationHours',
61     type: 'string'
62 }
63 ],
64 name: 'newSetting',
65 outputs: [],
66 payable: true,
67 stateMutability: 'payable',
68 type: 'function'
69 },
70 {
71     constant: true,
72     inputs: [
73         {
74             name: 'id',

```

```

75         type: 'string'
76     }
77 ],
78 name: 'getTokenSetting',
79 outputs: [
80     {
81         name: 'tokenName',
82         type: 'string'
83     },
84     {
85         name: 'tokenSymbol',
86         type: 'string'
87     },
88     {
89         name: 'tokenAddr',
90         type: 'string'
91     },
92     {
93         name: 'walletAddrs',
94         type: 'address[] '
95     }
96 ],
97 payable: false,
98 stateMutability: 'view',
99 type: 'function'
100 },
101 {
102     constant: true,
103     inputs: [
104         {
105             name: 'id',
106             type: 'string'
107         }
108     ],
109     name: 'getVoteSetting',
110     outputs: [
111         {
112             name: 'voteType',
113             type: 'string'
114         },
115         {
116             name: 'voteTokenLimit',
117             type: 'string'
118         },
119         {
120             name: 'voteAssignAddrs',
121             type: 'address[] '

```

```
122     },
123     {
124       name: 'voteMSupportPercent',
125       type: 'string'
126     },
127     {
128       name: 'voteMinApprovalPercent',
129       type: 'string'
130     },
131     {
132       name: 'voteMinDurationHours',
133       type: 'string'
134     },
135     {
136       name: 'voteMaxDurationHours',
137       type: 'string'
138     }
139   ],
140   payable: false,
141   stateMutability: 'view',
142   type: 'function'
143 }
144 ];
```

Listing 6: (参考文件 setting.js)