

解析指定合约过去的日志信息

我们以解析 cAAVE 代币合约的 Transfer 事件为例，介绍如何获取该合约在指定过滤器条件下的 Transfer 事件信息。

因为我们只监听 cAAVE 代币合约的 Transfer 事件，那么为了精简，我们可以在创建合约实例时只传入 Transfer 事件对应的 ABI，形式如下：

```
1 const cAAVE_Transfer_Abi = [{"anonymous":false,"inputs":[{"indexed":true,"intern
```

除此之外，我们在解析 data 数据时需要告诉方法如何解析（提供事件的格式），如下：

```
1
2 const inputAbi = [{
3   type: 'address',
4   name: 'from',
5   indexed: true
6 }, {
7   type: 'address',
8   name: 'to',
9   indexed: true
10 }, {
11   type: 'uint256',
12   name: 'value',
13   indexed: false
14 }]
```

准备好了这两个部分，我们就可以尝试获取 cAAVE 合约相关的 Transfer 事件的信息了。有两种方式可供选择：

- 方式 1：调用 cToken 提供的 getPastEvents 方法（推荐）

```
1 const Web3 = require("web3");
2 const web3 = new Web3('https://cloudflare-eth.com/');
3 // https://cloudflare-eth.com/
4
5 const cAAVE_Transfer = [{"anonymous":false,"inputs":[{"indexed":true,"internalTy
6
```

```

7 // 合约对象
8 const cAAVE = new web3.eth.Contract(cAAVE_Transfer, '0xe65cdB6479BaC1e22340E4E75
9
10
11 const inputAbi = [{
12     type: 'address',
13     name: 'from',
14     indexed: true
15 }, {
16     type: 'address',
17     name: 'to',
18     indexed: true
19 }, {
20     type: 'uint256',
21     name: 'value',
22     indexed: false
23 }]
24
25 // 查询历史区块，并过滤出相关事件
26 cAAVE.getPastEvents('Transfer', {
27     filter: {
28         from: ['0xe65cdB6479BaC1e22340E4E755fAE7E509EcD06c', '0x32a59b87352e980d
29         to: ['9bd48e08e3444b30113812acd52458b35c33803a', '0xe65cdB6479BaC1e22340
30     },
31     fromBlock: 17356700,
32     toBlock: 17356800,
33 })
34 .then((res) => {
35     for (let i = 0; i < res.length; i++) {
36         let raw = res[i].raw;
37         // 解码ABI编码的日志数据和索引主题数据。
38         let decodedata = web3.eth.abi.decodeLog(inputAbi, raw.data, [raw.topics[
39         console.log('from: ', decodedata[0]);
40         console.log('to: ', decodedata[1]);
41         console.log('value: ', decodedata[2]);
42     }
43 })
44 .catch((error) => {
45     console.log(error);
46 })
47

```

上述过滤器指定了 from 和 to，可以看到是以数组的形式表示的。

表示了解析那些 from 是过滤器 from 数组中任意一个值且 to 是过滤器 to 数组中任意一个值的事件并打印。

[illegible]

```
$ node pastLog.js
from: 0xe65cdB6479BaC1e22340E4E755fAE7E509EcD06c
to: 0x9BD48e08E3444B30113812acd52458b35C33803A
value: 48496843
```

📦 使用 `getPastEvent('Transer', {})` 方法时，可以设定扫描的区块范围，（比如 0~799） Max range: 800

- 不同于上述 cToken 合约提供的 API，基于订阅的事件监听都需要传入有效的 websocket 节点地址。

```
1 const Web3 = require("web3");
2 const web3 = new Web3('wss://mainnet.infura.io/ws/v3/437b550826914756922619d58d3
3
4 // 事件签名 hash
```

```

5 const transferSign = web3.eth.abi.encodeEventSignature('Transfer(address,address
6 // 合约 Transfer 事件 ABI
7 const cAAVE_Transfer = [{"anonymous":false,"inputs":[{"indexed":true,"internalTy
8
9 // 合约对象
10 const cAAVE = new web3.eth.Contract(cAAVE_Transfer, '0xe65cdb6479BaC1e22340E4E75
11
12 const inputAbi = [{
13     type: 'address',
14     name: 'from',
15     indexed: true
16 }, {
17     type: 'address',
18     name: 'to',
19     indexed: true
20 }, {
21     type: 'uint256',
22     name: 'value',
23     indexed: false
24 }]
25
26 cAAVE.events.Transfer({
27     filter: {
28         from: ['e65cdb6479bac1e22340e4e755fae7e509ecd06c'],
29         to: ['9bd48e08e3444b30113812acd52458b35c33803a'],
30         topics: [transferSign] // 可以省略, 因为调用的就是订阅 Transfer 事件的方法
31     },
32     fromBlock: 1730000
33 }, function(error, result) {
34 })
35 .on('data', function(event) {
36     let raw = event.raw;
37     let decodeData = web3.eth.abi.decodeLog(inputAbi, raw.data, [raw.topics[1],
38     console.log('from: ', decodeData[0])
39     console.log('to: ', decodeData[1])
40     console.log('value: ', decodeData[2])
41 })
42 .on('changed', function(event) {
43     console.log(event);
44 })
45 .on('error', function(error, receipt) {
46     console.log(error, receipt);
47 });

```

脚本逻辑实际上和方式 1 类似, 过滤器多出了一个新的选项 topics, topics[0] 默认为事件的签名哈希值, 之后部分都是事件定义中的 indexed 参数。

- 输出结果

```
# apple @ appledeMacBook-Air1 in ~/Study/Blockchain/ETH/web3.js/event [7:29:00]  
$ node eventTransfer.js  
from: 0xe65cdB6479BaC1e22340E4E755fAE7E509EcD06c  
to: 0x9BD48e08E3444B30113812acd52458b35C33803A  
value: 48496843
```