

```

1  var Web3 = require('web3');
2  var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
3
4  var subject = "0x0d1f8a489b1312689f11f7fe79dfc3b61ffa4160";
5  var regAddr = "0xddf27a729d05be6f11be50b1905daa6e7b508c91";
6  var regAbi = [
7      {
8          "constant": true,
9          "inputs": [
10             {
11                 "name": "_str",
12                 "type": "string"
13             }
14         ],
15         "name": "stringToBytes32",
16         "outputs": [
17             {
18                 "name": "",
19                 "type": "bytes32"
20             }
21         ],
22         "payable": false,
23         "stateMutability": "view",
24         "type": "function"
25     },
26     {
27         "constant": true,
28         "inputs": [
29             {
30                 "name": "",
31                 "type": "bytes32"
32             }
33         ],
34         "name": "lookupTable",
35         "outputs": [
36             {
37                 "name": "scName",
38                 "type": "string"
39             },
40             {
41                 "name": "subject",
42                 "type": "address"
43             },
44             {
45                 "name": "object",
46                 "type": "address"
47             },
48             {
49                 "name": "creator",
50                 "type": "address"
51             },
52             {
53                 "name": "scAddress",
54                 "type": "address"
55             },
56             {
57                 "name": "abi",
58                 "type": "bytes"
59             }
60         ],
61         "payable": false,
62         "stateMutability": "view",
63         "type": "function"
64     },
65     {
66         "constant": true,

```

```

67     "inputs": [
68         {
69             "name": "_methodName",
70             "type": "string"
71         }
72     ],
73     "name": "getContractAddr",
74     "outputs": [
75         {
76             "name": "_scAddress",
77             "type": "address"
78         }
79     ],
80     "payable": false,
81     "stateMutability": "view",
82     "type": "function"
83 },
84 {
85     "constant": true,
86     "inputs": [
87         {
88             "name": "_methodName",
89             "type": "string"
90         }
91     ],
92     "name": "getContractAbi",
93     "outputs": [
94         {
95             "name": "_abi",
96             "type": "bytes"
97         }
98     ],
99     "payable": false,
100    "stateMutability": "view",
101    "type": "function"
102 },
103 {
104     "constant": false,
105     "inputs": [
106         {
107             "name": "_methodName",
108             "type": "string"
109         },
110         {
111             "name": "_abi",
112             "type": "bytes"
113         }
114     ],
115     "name": "methodAbiUpdate",
116     "outputs": [],
117     "payable": false,
118     "stateMutability": "nonpayable",
119     "type": "function"
120 },
121 {
122     "constant": false,
123     "inputs": [
124         {
125             "name": "_methodName",
126             "type": "string"
127         },
128         {
129             "name": "_scAddress",
130             "type": "address"
131         }
132     ],

```

```

133     "name": "methodAcAddressUpdate",
134     "outputs": [],
135     "payable": false,
136     "stateMutability": "nonpayable",
137     "type": "function"
138 },
139 {
140     "constant": false,
141     "inputs": [
142         {
143             "name": "_name",
144             "type": "string"
145         }
146     ],
147     "name": "methodDelete",
148     "outputs": [],
149     "payable": false,
150     "stateMutability": "nonpayable",
151     "type": "function"
152 },
153 {
154     "constant": false,
155     "inputs": [
156         {
157             "name": "_oldName",
158             "type": "string"
159         },
160         {
161             "name": "_newName",
162             "type": "string"
163         }
164     ],
165     "name": "methodNameUpdate",
166     "outputs": [],
167     "payable": false,
168     "stateMutability": "nonpayable",
169     "type": "function"
170 },
171 {
172     "constant": false,
173     "inputs": [
174         {
175             "name": "_methodName",
176             "type": "string"
177         },
178         {
179             "name": "_scname",
180             "type": "string"
181         },
182         {
183             "name": "_subject",
184             "type": "address"
185         },
186         {
187             "name": "_object",
188             "type": "address"
189         },
190         {
191             "name": "_creator",
192             "type": "address"
193         },
194         {
195             "name": "_scAddress",
196             "type": "address"
197         },
198         {

```

```

199         "name": "_abi",
200         "type": "bytes"
201     }
202 ],
203     "name": "methodRegister",
204     "outputs": [],
205     "payable": false,
206     "stateMutability": "nonpayable",
207     "type": "function"
208 },
209 {
210     "constant": false,
211     "inputs": [
212         {
213             "name": "_methodName",
214             "type": "string"
215         },
216         {
217             "name": "_scName",
218             "type": "string"
219         }
220     ],
221     "name": "methodScNameUpdate",
222     "outputs": [],
223     "payable": false,
224     "stateMutability": "nonpayable",
225     "type": "function"
226 }
227 ];
228
229
230 var methodName = "Method1";
231 var register = web3.eth.contract(regAbi).at(regAddr);
232 var accAddr = register.getContractAddr(methodName);
233 var accAbiBytes = register.getContractAbi(methodName);
234 var accAbi = JSON.parse(web3.toAscii(accAbiBytes));
235 var myACC = web3.eth.contract(accAbi).at(accAddr);
236
237 var myEvent = myACC.ReturnAccessResult({_from: subject},{from: 'latest'});
238 var previousTxHash = 0;
239 var currentTxHash = 0;
240 const readline = require('readline');
241 const rl = readline.createInterface({
242     input: process.stdin,
243     output: process.stdout
244 });
245
246 rl.setPrompt('Send access request?(y/n)');
247 rl.prompt();
248
249 rl.on('line', (answer) => {
250     if('y' == answer) {
251         var currentTime = new Date().getTime()/1000;
252         currentTxHash = myACC.accessControl.sendTransaction("File A", "write", currentTime, {from:
web3.eth.accounts[0], gas:3000000});
253
254         myEvent.watch(function(err, result){
255             if(!err){
256                 if(previousTxHash != result.transactionHash && currentTxHash ==
result.transactionHash){//avoid duplicate event captured
257                     console.log("Contract: "+result.address);
258                     console.log("Block Number: " + result.blockNumber);
259                     console.log("Tx Hash: " + result.transactionHash);
260                     console.log("Block Hash: " + result.blockHash);
261                     console.log("Time: " + result.args._time.toNumber());
262                     console.log("Message: " + result.args._errmsg);

```

```
263             console.log("Result: " + result.args._result);
264             if (result.args._penalty > 0)
265                 console.log("Requests are blocked for " + result.args._penalty + "
minutes!");
266             console.log('\n');
267             previousTxHash = result.transactionHash;
268             rl.prompt();
269         }
270     }
271 });
272 }
273 else{
274     console.log("Ummmmmm...");
275     rl.prompt();
276 }
277 });
278
279 rl.on('close', function() {
280     console.log('Have a great day!');
281     process.exit(0);
282 });
```