```
var Web3 = require('web3');
var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

var subject = "0x0d1f8a489b1312689f11f7fe79dfc3b61ffa4160";
var regAddr = "0xddf27a729d05be6f11be50b1905daa6e7b508c91";
var regAbi = [
    {
        "constant": true,
        "inputs": [
            {
                "name": "_str",
                "type": "string"
            }
        ],
        "name": "stringToBytes32",
        "outputs": [
            {
                "name": "",
                "type": "bytes32"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "",
                "type": "bytes32"
            }
        ],
        "name": "lookupTable",
        "outputs": [
            {
                "name": "scName",
                "type": "string"
            },
            {
                "name": "subject",
                "type": "address"
            },
            {
                "name": "object",
                "type": "address"
            },
            {
                "name": "creator",
                "type": "address"
            },
            {
                "name": "scAddress",
                "type": "address"
            },
            {
                "name": "abi",
                "type": "bytes"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
```

```
        "inputs": [
            {
                "name": "_methodName",
                "type": "string"
            }
        ],
        "name": "getContractAddr",
        "outputs": [
            {
                "name": "_scAddress",
                "type": "address"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "_methodName",
                "type": "string"
            }
        ],
        "name": "getContractAbi",
        "outputs": [
            {
                "name": "_abi",
                "type": "bytes"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_methodName",
                "type": "string"
            },
            {
                "name": "_abi",
                "type": "bytes"
            }
        ],
        "name": "methodAbiUpdate",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_methodName",
                "type": "string"
            },
            {
                "name": "_scAddress",
                "type": "address"
            }
        ],
```

```
        "name": "methodAcAddressUpdate",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_name",
                "type": "string"
            }
        ],
        "name": "methodDelete",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_oldName",
                "type": "string"
            },
            {
                "name": "_newName",
                "type": "string"
            }
        ],
        "name": "methodNameUpdate",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_methodName",
                "type": "string"
            },
            {
                "name": "_scname",
                "type": "string"
            },
            {
                "name": "_subject",
                "type": "address"
            },
            {
                "name": "_object",
                "type": "address"
            },
            {
                "name": "_creator",
                "type": "address"
            },
            {
                "name": "_scAddress",
                "type": "address"
            },
            {
```

```
                "name": "_abi",
                "type": "bytes"
            }
        ],
        "name": "methodRegister",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_methodName",
                "type": "string"
            },
            {
                "name": "_scName",
                "type": "string"
            }
        ],
        "name": "methodScNameUpdate",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    }
];


var methodName = "Method1";
var register = web3.eth.contract(regAbi).at(regAddr);
var accAddr = register.getContractAddr(methodName);
var accAbiBytes = register.getContractAbi(methodName);
var accAbi = JSON.parse(web3.toAscii(accAbiBytes));
var myACC = web3.eth.contract(accAbi).at(accAddr);

var myEvent = myACC.ReturnAccessResult({_from: subject},{from: 'latest'});
var previousTxHash = 0;
var currentTxHash = 0;
const readline = require('readline');
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.setPrompt('Send access request?(y/n)');
rl.prompt();

rl.on('line',  (answer) => {
    if('y' == answer) {
        var currentTime = new Date().getTime()/1000;
        currentTxHash = myACC.accessControl.sendTransaction("File A", "write", currentTime, {from:
web3.eth.accounts[0], gas:3000000});

        myEvent.watch(function(err, result){
            if(!err){
                if(previousTxHash != result.transactionHash && currentTxHash ==
result.transactionHash){//avoid dupicate event captured
                    console.log("Contract: "+result.address);
                    console.log("Block Number: " + result.blockNumber);
                            console.log("Tx Hash: " + result.transactionHash);
                            console.log("Block Hash: "+ result.blockHash);
                                    console.log("Time: " + result.args._time.toNumber());
                            console.log("Message: " + result.args._errmsg);
```

```
                                console.log("Result: " + result.args._result);
                                            if (result.args._penalty > 0)
                                        console.log("Requests are blocked for " + result.args._penalty + "
minutes!");
                        console.log('\n');
                        previousTxHash = result.transactionHash;
                        rl.prompt();
                    }
                }
            });
        }
        else{
            console.log("Ummmmm....");
            rl.prompt();
        }
    });

    rl.on('close', function() {
            console.log('Have a great day!');
        process.exit(0);
    });
```