```
 1  pragma solidity ^0.4.0;
 2
 3  contract Register{
 4   struct Method{
 5       string scName;       //contract name
 6       address subject;     //the subject of the corresponding subject-object pair of the ACC; for the JC,
this filed is left blank;
 7       address object;      //the subject of the corresponding subject-object pair of the ACC; for the JC,
this filed is left blank;
 8       address creator;     //the peer (account) who created and deployed this contract;
 9       address scAddress;   //the address of the contract;
10       bytes abi;           //the Abi’s provided by the contract.
11   }
12
13   /*As solidity cannot allow dynamically-sized value as the Key, we use the fixed-szie byte32 type as the
keytype*/
14   mapping (bytes32=> Method) public lookupTable;
15
16   /*convert strings to byte32*/
17   function stringToBytes32(string _str) public constant returns (bytes32){
18       bytes memory tempBytes = bytes(_str);
19       bytes32 convertedBytes;
20       if(0 == tempBytes.length){
21           return 0x0;
22       }
23       assembly {
24          convertedBytes := mload(add(_str, 32))
25       }
26       return convertedBytes;
27   }
28
29   /*register an access control contract (ACC)*/
30   function methodRegister(string _methodName, string _scname, address _subject, address _object, address
_creator, address _scAddress, bytes _abi) public {
31      //no duplicate check
32      bytes32 newKey = stringToBytes32(_methodName);
33      lookupTable[newKey].scName = _scname;
34      lookupTable[newKey].subject = _subject;
35      lookupTable[newKey].object = _object;
36      lookupTable[newKey].creator = _creator;
37      lookupTable[newKey].scAddress = _scAddress;
38      lookupTable[newKey].abi = _abi;
39   }
40
41   /*update the ACC information (i.e., scname, scAddress, abi) of an exisiting method specified by the
_methodName*/
42
43
44   function methodScNameUpdate(string _methodName, string _scName) public{
45       bytes32 key = stringToBytes32(_methodName);
46       lookupTable[key].scName = _scName;
47   }
48
49   function methodAcAddressUpdate(string _methodName, address _scAddress) public{
50       bytes32 key = stringToBytes32(_methodName);
51       lookupTable[key].scAddress = _scAddress;
52   }
53
54   function methodAbiUpdate(string _methodName, bytes _abi) public{
55       bytes32 key = stringToBytes32(_methodName);
56       lookupTable[key].abi = _abi;
57   }
58
59   /*update the name (_oldname) of an exisiting method with a new name (_newname) */
60   function methodNameUpdate(string _oldName, string _newName) public{
61       bytes32 oldKey = stringToBytes32(_oldName);
```

```
62        bytes32 newKey = stringToBytes32(_newName);
63        lookupTable[newKey].scName = lookupTable[oldKey].scName;
64        lookupTable[newKey].subject = lookupTable[oldKey].subject;
65        lookupTable[newKey].object = lookupTable[oldKey].object;
66        lookupTable[newKey].creator = lookupTable[oldKey].creator;
67        lookupTable[newKey].scAddress = lookupTable[oldKey].scAddress;
68        lookupTable[newKey].abi = lookupTable[oldKey].abi;
69        delete lookupTable[oldKey];
70    }
71
72    function methodDelete(string _name) public{
73        delete lookupTable[stringToBytes32(_name)];
74    }
75
76     function getContractAddr(string _methodName) public constant returns (address _scAddress){
77        bytes32 key = stringToBytes32(_methodName);
78        _scAddress = lookupTable[key].scAddress;
79    }
80
81    function getContractAbi(string _methodName) public constant returns (bytes _abi){
82        bytes32 key = stringToBytes32(_methodName);
83        _abi = lookupTable[key].abi;
84    }
85
```