Gmail

Nagendra Rapaka <r.nagendra@gmail.com>

Code

1 message

Harsh Patel <harsh.patel54@gmail.com>
To: r.nagendra@gmail.com

Sun, Nov 11, 2018 at 2:39 PM

```
pragma solidity ^0.4.18;

// File: srcContracts/Ownable.sol

/**
@title OwnableContract
*/
contract Ownable {
  address public owner;

  event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

  constructor ()
    public
  {
    owner = 0x001;
  }

  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }

  /**
  @dev transferOwnership of Owner to a new owner, This function can be only be called by owner
  @param newOwner The address of the New Owner
  */
  function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0));
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
  }
}

// File: srcContracts/regRegis.sol

/**
  @title regRegis
  @author Harsh Patel
*/
 contract regRegis is Ownable {

   mapping ( bytes32 => Regis ) public reg;
   mapping ( address => bytes32 ) public addrMap;

   struct Regis{
     string verifier;
     bool state;
   }

   event setterEVNT(bytes32 indexed userID);
   event removeEVNT(bytes32 indexed userID);

   constructor ()
     public
   {
   }

   modifier isValidSet(bytes32 H) {
     require(((reg[H].state == false) && (addrMap[msg.sender] == 0x0)) || ( addrMap[msg.sender] == H ));
     _;
   }

   /**
   @dev Associates senders address to the hash of the username and allocates Hash of username and only sender can update the verifier.
   @param H Hash of the username
   @param _verifier Verifier for the username
   */
   function set(bytes32 H,string _verifier )
     public
     isValidSet(H)
   {
     reg[H].verifier   = _verifier;
     reg[H].state      = true;
     addrMap[msg.sender] = H;
     emit setterEVNT(H);
   }

   /**
   @dev Removes users verifier and all assocaited details from the system
   @param H Hash of the username
   */
   function remove(bytes32 H)
     onlyOwner
     public
   {
     reg[H].verifier  = "";
     addrMap[msg.sender] = 0x0;
     reg[H].state      = false;
     emit removeEVNT(H);
   }

 }
```

```
contract remoteRegis{

    regRegis r1;
    address regAddr;
    constructor( address _regAddr){
        regAddr = _regAddr;
        r1 = regRegis(regAddr);
    }
    function getaddrMap(address _addr) public constant returns ( bytes32 H ){
        return r1.addrMap(_addr);
    }
    function setValues(bytes32 H,string _verifier  ) public
    {
        r1.set(H,_verifier);
    }


}
```