

Multi signature

Adding members and give them authorization

The screenshot displays the Remix IDE interface for deploying and running a Solidity contract named `MultiSig1.sol`.

Left Panel: DEPLOY & RUN TRANSACTIONS

- `transferInsura...` (address to, uint256 _id)
- `verify` (uint256 _id)
- `authenticators` (uint256)
- `client` (address, uint256)
- `contractOwner`
- `getAuths`** (0: address[]; 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4, 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
- `insOwner`
- `isAdmin` (address)
- `ownerOf` (uint256 tokenId)
- `productCounter`

Main Editor: MultiSig1.sol

```
180 }
181
182 // -- helper functions
183 function getAuths() public view returns (address[] memory)
184 {
185     return authenticators;
186 }
187 //here will adding members and give them Authorization
188 function addAuths(address _address) public onlyContractOwner() AuthenticatorExist(_address) no
189     isAdmin[_address] = true;
190     Authrize(_address);
191 }
192
193 function Authrize(address _add) private returns (uint _x) {
194     require(isAdmin[msg.sender] == true, "no you can't");
195     authenticators.push(_add);
196     return (authenticators.length);
197 }
198
199 function verify(uint _id) public {
200     require(isAdmin[msg.sender] == true, "You're not Authrize");
201 }
```

Bottom Panel: Transaction Log

- `[vm] from: 0x5B3...eddC4 to: TechInsurance.addAuths(address) 0xd91...39138`
`value: 0 wei data: 0x770...35cb2 logs: 0 hash: 0x6cf...92458` **Debug**
- `call to TechInsurance.getAuths`
- `[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4`
`to: TechInsurance.getAuths() data: 0x6a6...d95d7` **Debug**

available modifier will control adding product after number of signature (M) accomplish

The image displays two screenshots of the Remix IDE interface, illustrating the execution of a Solidity contract named `TechInsurance`.

Top Screenshot:

- Left Panel (DEPLOY & RUN TRANSACTIONS):** Shows the `addProduct` function being executed. The input fields are `_productId: 11`, `_productName: apple`, and `_price: 1`. The `transact` button is highlighted.
- Right Panel (Code Editor):** Displays the Solidity code for `MultiSig1.sol`. A red box highlights the `available` modifier definition:

```
173 //available modifier will control adding product after number of signature(M) accomplish
174 modifier available () {
175     require(getAuths().length >= M,"You are not allow");
176 }
177
```
- Bottom Panel (Transaction Log):** Shows the execution of the `addProduct` function. The log entry is:

```
[vm] from: 0x5B3...eddC4
to: TechInsurance.addProduct(uint256,string,uint256) 0xd91...39138 value: 0 wei
data: 0x426...00000 logs: 1 hash: 0xecc...5dbc6
```

Bottom Screenshot:

- Left Panel (DEPLOY & RUN TRANSACTIONS):** The `addProduct` function is still selected, and the `transact` button is highlighted.
- Right Panel (Code Editor):** Displays the Solidity code for `MultiSig1.sol`. A red box highlights the `addProduct` function definition:

```
57 function addProduct(uint _productId, string memory _productName, uint _price ) public available {
58     Product memory newProduct = Product (_productId, _productName, _price, false);
59     // productIndex[productCounter++] = newProduct;
60     productIndex[_productId] = newProduct;
61     //v_Authenticators[_productId] = 0;
62     _mint(msg.sender, _productId);
63 }
64
```
- Bottom Panel (Transaction Log):** Shows the execution of the `addProduct` function. The log entry is:

```
[vm] from: 0x5B3...eddC4
to: TechInsurance.addProduct(uint256,string,uint256) 0xd91...39138 value: 0 wei
data: 0x426...00000 logs: 1 hash: 0xecc...5dbc6
```

The product will be not offer until number of signed members verify the product

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel displays a transaction configuration for 'MultiSig1.sol'. The 'productIndex' is set to 11. The 'v_Authenticators' array is empty. The 'productCounter' is 0. The 'productIndex' field is highlighted with a red box. The 'Authz' function is selected in the 'Low level interactions' section. The 'Transact' button is visible. The main editor shows the Solidity code for 'MultiSig1.sol', with the 'verify' function highlighted by a red box. The code includes a comment: '//the product will be not offer until number of signed members verify the product'. The execution cost is 'infinite gas'.

After number of members verify the product by product id it become offered

The screenshot shows the Remix IDE interface after a transaction. The 'DEPLOY & RUN TRANSACTIONS' panel displays the same transaction configuration. The 'productIndex' is still 11. The 'v_Authenticators' array now contains one element: 'uint256: 2'. The 'productCounter' is 1. The 'productIndex' field is highlighted with a red box. The 'Authz' function is selected in the 'Low level interactions' section. The 'Transact' button is visible. The main editor shows the Solidity code for 'MultiSig1.sol', with the 'verify' function highlighted by a red box. The code includes a comment: '//the product will be not offer until number of signed members verify the product'. The execution cost is 'infinite gas'.

The insurance will not be available to sell until signed member verify the product by making state of product (offered = true)

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows a list of functions: `buyInsurance`, `changeFalse`, `changePrice`, `changeTrue`, `refund`, and `transferInsura...`. The `buyInsurance` function is selected, and its parameters are set to `_productIndex: 11`. Below this, the `verify` function is also shown with `_id: 11`. On the right, the Solidity source code for the `MultiSig1.sol` contract is visible. The `buyInsurance` function is highlighted with a red box. The code includes a `require` statement that sets `productIndex[_productIndex].offered == true`, followed by a `require` statement that checks the amount of the insurance. The `transferInsurance` function is also visible. At the bottom, the execution results panel shows a transaction to `TechInsurance.buyInsurance` that errored with a VM error: `revert. revert The transaction has been reverted to the initial state. Reason provided by the contract: "check the amount of the Insurance".` A red checkmark is placed next to the execution results.

```
function buyInsurance(uint _productIndex) public payable {
    require(productIndex[_productIndex].offered == true, "The Insurance is not available");
    Client memory newClient = Client(true, block.timestamp);
    client[msg.sender][_productIndex] = newClient;
    require(msg.value == productIndex[_productIndex].price, "check the amount of the Insurance");
    //*****
    time = client[msg.sender][_productIndex].time;
    //*****
    uint256 price = productIndex[_productIndex].price;
    // changeFalse(_productIndex);
    payable(msg.sender).transfer(price);
    _transfer(ownerOf(_productIndex), msg.sender, _productIndex);
}

function transferInsurance(address to, uint256 _id) public {
    require(msg.sender != to, "You are the owner of this Insurance");
    require(ownerOf(_id) == msg.sender, "Your not the Owner");
    _transfer(msg.sender, to, _id);
}
```

transact to TechInsurance.buyInsurance errored: VM error: revert. revert The transaction has been reverted to the initial state. Reason provided by the contract: "check the amount of the Insurance". Debug the transaction to get more information.

transact to TechInsurance.buyInsurance pending ...

[vm] from: 0x4B2...C02db to: TechInsurance.buyInsurance(uint256) 0xd91...39138
value: 1 wei data: 0xa86...0000b logs: 0 hash: 0xb18...d8782

Salwa Alhajjaji