

Department of Computer Engineering

Academic Term: Jan-May 23-24

Class: B.E Computer Sem -VII

Subject: Blockchain Technology Lab

Subject Code : CSDL7022

Practical No:	4
Title:	Creating Tokens/Coins on Ethereum Network
Date of Performance:	18/08/2023
Date of Submission:	18 08/2023
Roll No:	9427
Name of the Student:	Atharva Prashant Pawar

Evaluation:

Sr. No	Rubric	Grade
1	Time Line (2)	
2	Output (3)	
3	Code optimization (2)	
4	Post lab (3)	

Signature of the Teacher :

Experiment No. 4

Creating Tokens/Coins on Ethereum Network

Aim: Creating a Metamask wallet and performing the transactions to add Ethers and Tokens.



Theory:

Step 1: Create a wallet at meta-mask

- Install MetaMask in Chrome browser and enable it. After installation, click on its icon on the top right of the browser page. It will open in a new tab of the browser. Click on “Create Wallet” and click “I agree” to proceed further. Now you are agreeing to the terms and conditions. Create a password and then it will send you a secret backup phrase used for backing up and restoring the account.
- It should not be disclosed or shared with anyone, as this phrase can steal your Ethers. Ensure that you are in the “Main Ethereum Network.” If you find a checkmark next to “Main Ethereum Network”, you are in the right place. You can see following pages.



New to MetaMask?

 <p>No, I already have a Secret Recovery Phrase</p> <p>Import your existing wallet using a Secret Recovery Phrase</p> <p>Import wallet</p>	 <p>Yes, let's get set up!</p> <p>This will create a new wallet and Secret Recovery Phrase</p> <p>Create a Wallet</p>
---	--

Secret Recovery Phrase

Your Secret Recovery Phrase makes it easy to back up and restore your account.

WARNING: Never disclose your Secret Recovery Phrase. Anyone with this phrase can take your Ether forever.

mechanic omit honey slim pony
casino brush bag amused knife
clarify spin

Tips:

Store this phrase in a password manager like 1Password.


Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

[Download this Secret Recovery Phrase and keep it stored safely on an external encrypted hard drive or storage medium.](#)


Confirm your Secret Recovery Phrase

Please select each phrase in order to make sure it is correct.

 METAMASK

Ethereum Mainnet

Account 1
0x352...38DE


0 ETH
\$0.00 USD


Buy

Send

Swap

Assets

Activity

 0 ETH
\$0.00 USD

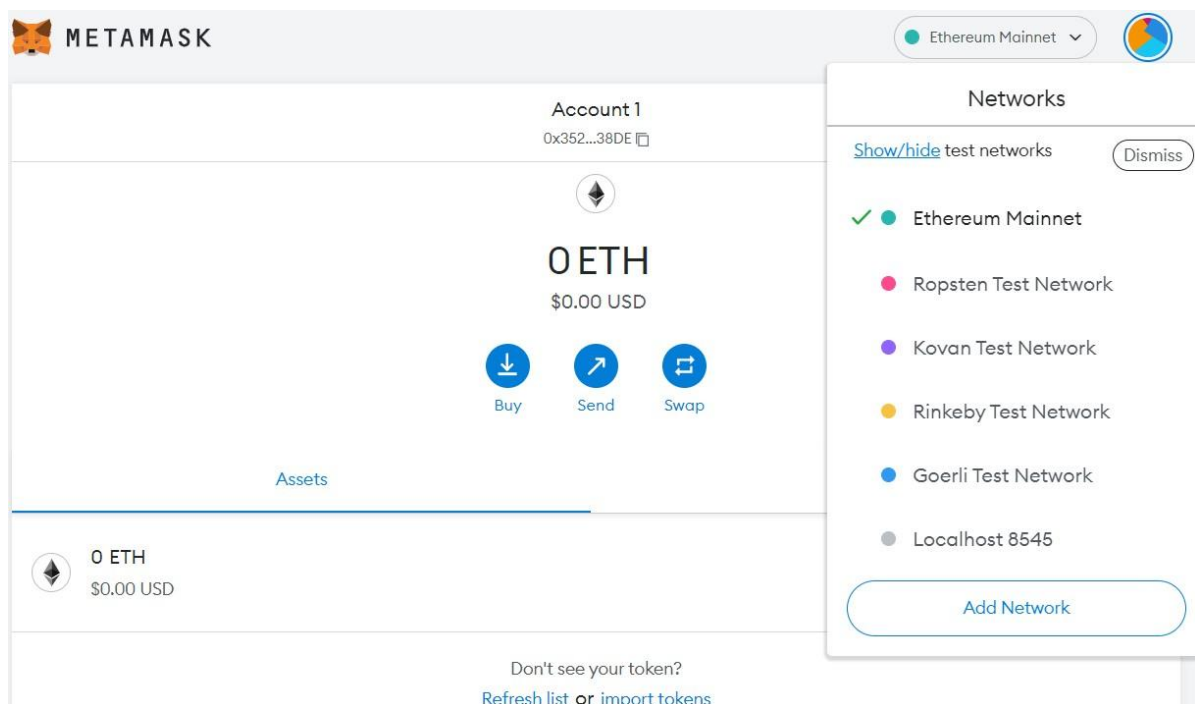
Don't see your token?
[Refresh list](#) or [import tokens](#)

Step 2: Select any one test network

Click on show networks. Enable show networks and other information. Following test networks can be seen in your MetaMask wallet. These networks are only for the purpose of testing and ethers involved in it have no value.

- Ropsten Test Network
- Kovan Test Network
- Rinkeby Test Network
- Goerli Test Network

Select Ropsten Test Network.



Step 3: Add some dummy Ethers to your wallet

- To test the smart contract, your MetaMask wallet should contain some dummy ethers. For example, if you test a contract using the Ropsten test network, select it and you will find 0 ETH as the initial balance in your account. Click on the “Deposit” and “Get Ether” buttons under
- Test Faucet in order to add dummy ethers. Open site <https://faucet.metamask.io/>. Link your wallet account and add one dummy ether. You will receive one ether in account as below. Click Request 1 Ether from faucet.
- Click on Buy button. Buy test ethers.

faucet

address: 0x81b7e08f65bdf5648606c89998a9cc8164397647

balance: 98894180.57 ether

request 1 ether from faucet

user

address: 0xdb45a091367e0ac1a58636bcb11c67ad16dd8aa9

balance: 4.00 ether

donate to faucet:

1 ether

10 ether

100 ether

transactions

0x97ee970e682d0eae5c82e1b95b1fc6e66d33e08c7209d19c22a92a1f10b95a63
0x4b416e8477e0550d249248a9574524036de93298d194a8d1e85774a5a39da1f4
0xcfcfad84b16a977889581ea4d127781029a27e4179a981ffcbec56f9b404ff
0x75132675292b37d3c0470a9649adf099c03e03bb870bc9c431f9f9a37aceb410
0xa04e7b8839e14c63568c43c6ed14707e8bec60cb6858fe2939d8ca178028b3d8
0x224fc0ceb0896bf62caeb10faeb0b89900c42085513e500935849803a8eb5388
0x95d128fa7139112c5a7c58c3056f98683a58cdb6c112ab7bc953904826766633

Step 4: Compile and Deploy following contract in Ethereum Remix IDE. Give name to file token.sol. Deploy by selecting **QKCToken-contract/token.sol**. Select **Environment as Injected Provider-Metamask**. Replace **YOUR_METAMASK_WALLET_ADDRESS** in code by your wallet address which you have created.

```
pragma solidity ^0.4.24;.
```

```
//Safe Math Interface
```

```
contract SafeMath {
```

```
    function safeAdd(uint a, uint b) public pure returns (uint c)
```

```
    {c = a + b;
```

```
    require(c >= a);
```

```
}
```

```
function safeSub(uint a, uint b) public pure returns (uint c)
```

```
{require(b <= a);
```

```

        c = a - b;
    }
    function safeMul(uint a, uint b) public pure returns (uint c)
    {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    function safeDiv(uint a, uint b) public pure returns (uint c)
    {
        require(b > 0);
        c = a / b;
    }
}

//ERC Token Standard #20 Interface
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
    function allowance(address tokenOwner, address spender) public constant returns (uint
remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}

//Contract function to receive approval and execute function in one call
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data) public;
}

//Actual token contract

contract QKCToken is ERC20Interface, SafeMath
{
    string public symbol;
    string public name;
    uint8 public decimals;
    uint public _totalSupply;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;

    constructor() public
    {
        symbol = "QKC";
        name = "QuikNode Coin";
        decimals = 2;
    }
}

```

```

    _totalSupply = 100000;
    balances[YOUR_METAMASK_WALLET_ADDRESS] = _totalSupply;
    emit Transfer(address(0), YOUR_METAMASK_WALLET_ADDRESS, _totalSupply);
}

function totalSupply() public constant returns (uint)
{
    return _totalSupply - balances[address(0)];
}

function balanceOf(address tokenOwner) public constant returns (uint balance)
{
    return balances[tokenOwner];
}

function transfer(address to, uint tokens) public returns (bool success)
{
    balances[msg.sender] = safeSub(balances[msg.sender], tokens);
    balances[to] = safeAdd(balances[to], tokens);
    emit Transfer(msg.sender, to, tokens);
    return true;
}

function approve(address spender, uint tokens) public returns (bool success)
{
    allowed[msg.sender][spender] = tokens;
    emit Approval(msg.sender, spender, tokens);
    return true;
}

function transferFrom(address from, address to, uint tokens) public returns (bool success)
{
    balances[from] = safeSub(balances[from], tokens);
    allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
    balances[to] = safeAdd(balances[to], tokens);
    emit Transfer(from, to, tokens);
    return true;
}

function allowance(address tokenOwner, address spender) public constant returns (uint remaining) {
    return allowed[tokenOwner][spender];
}

function approveAndCall(address spender, uint tokens, bytes data) public returns (bool success) {
    allowed[msg.sender][spender] = tokens;
    emit Approval(msg.sender, spender, tokens);
    ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data);
}

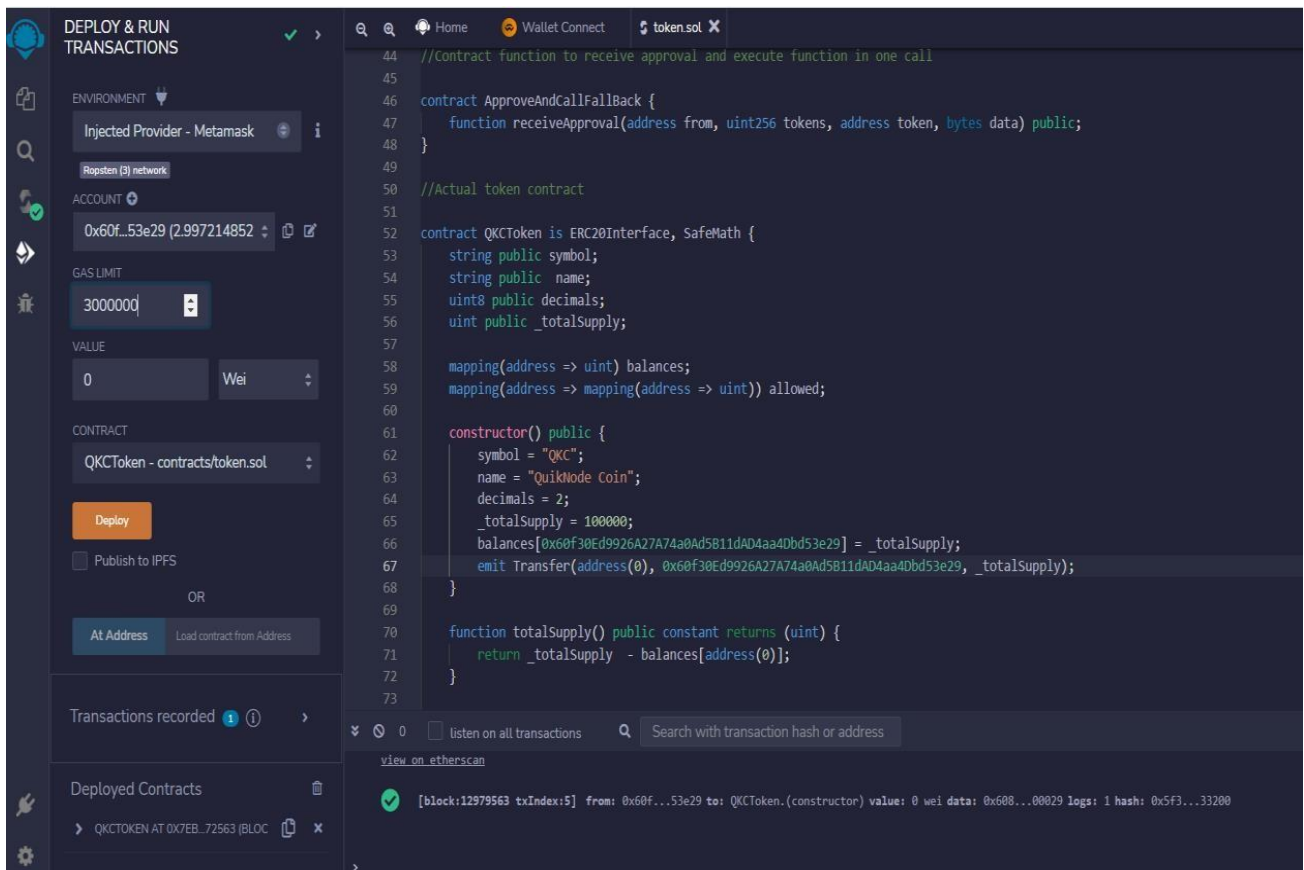
```

```

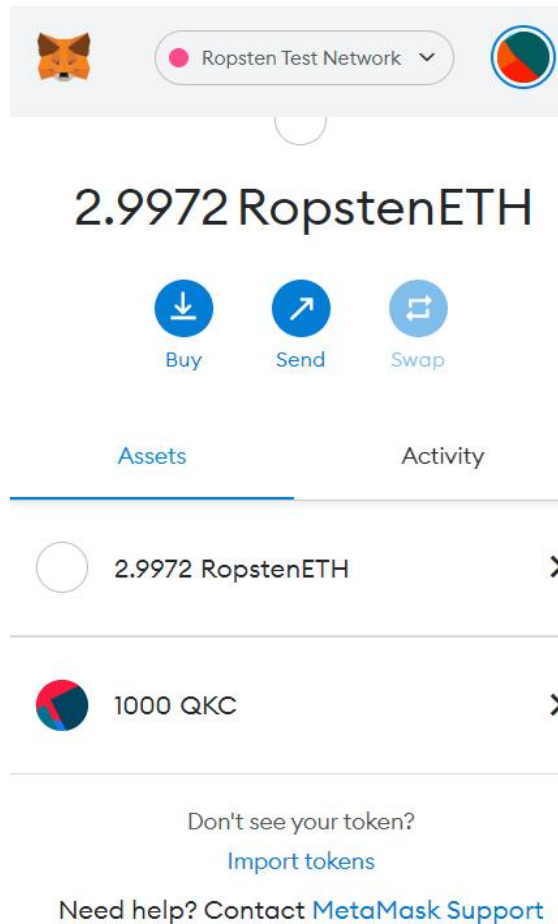
    return true;
}

function () public payable
{revert();
}
}

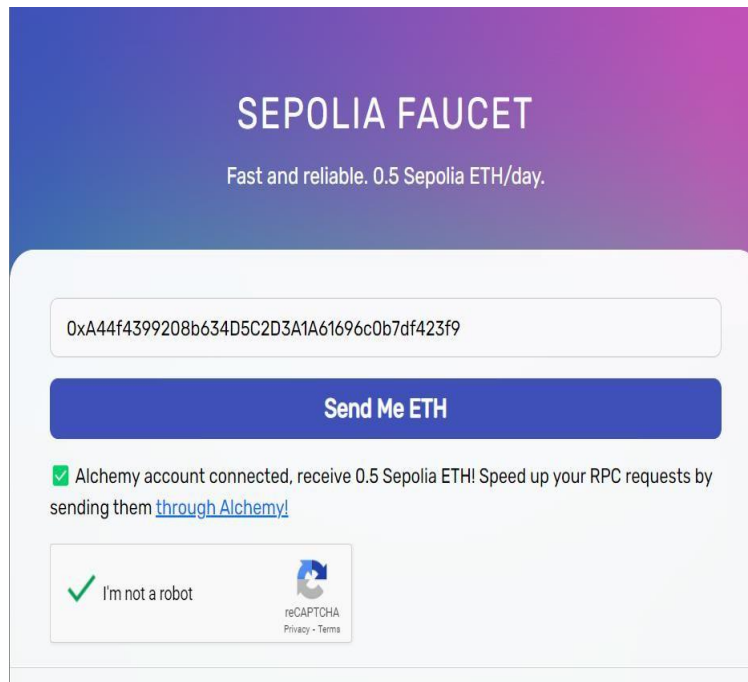
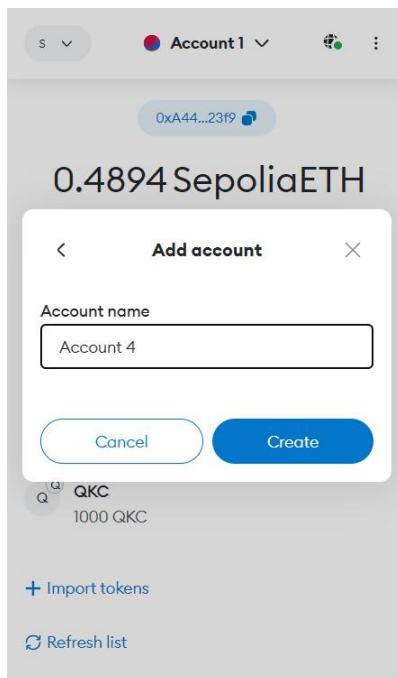
```



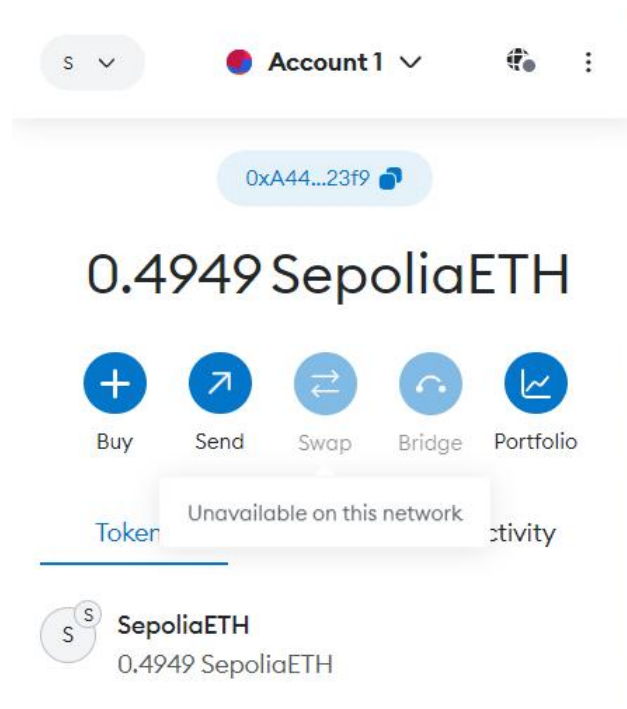
Step 5: Copy deployed contract address. Open Metamask. Click on Assest. Goto import token. Paste Token contract address, Select Token Symbol as QKC and Token decimal as 2 and add tokens. You will get 1000 QKC added as below.



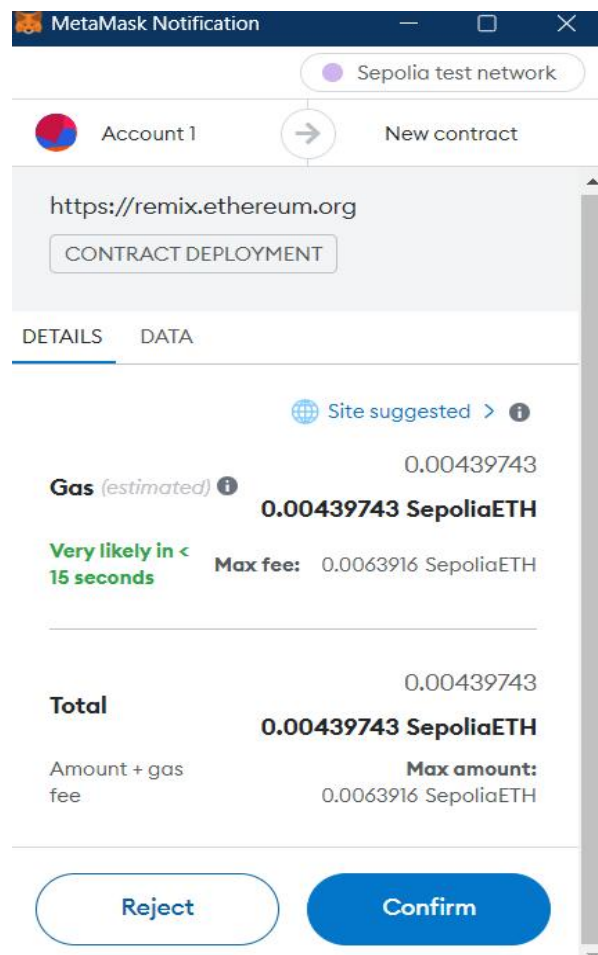
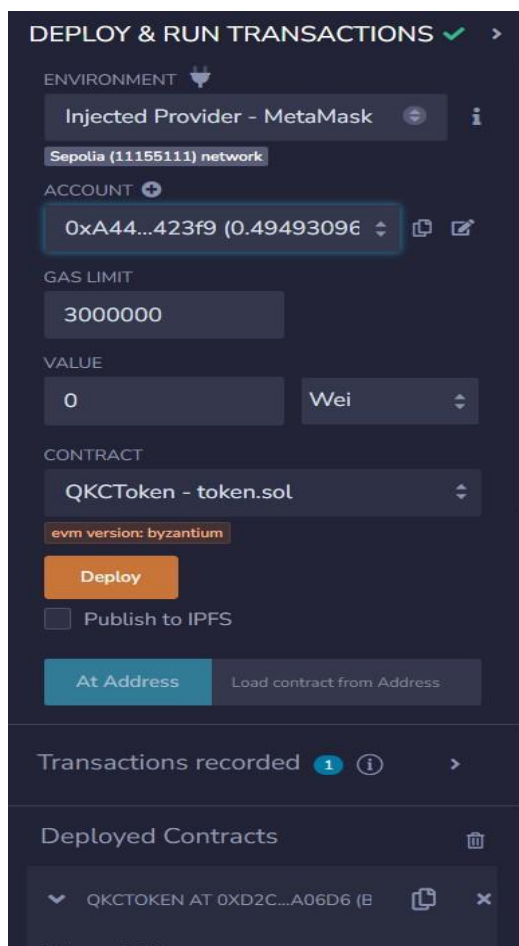
Output :



1) Creating new metamask account 4 and getting test eth from sepolia faucet



2) Sepolia test network




3) Deployed Contract -

Address - 0xA44f4399208b634D5C2D3A1A61696c0b7df423f9

4)Adding custom QKC tokens - 1000 QKC

S ▼

Account 1 ▼



⋮

Import tokens ×

Custom token

and make sure you trust it. Learn about [scams](#) and security risks.

Token contract address

0xFa0F5b1C565F1102CbC8161eE9d6

Token symbol Edit

QKC


Token decimal

2

Add custom token

S ▼

Account 1 ▼



⋮

0.4894 SepoliaETH

+

Buy

↗

Send

↔

Swap

↻

Bridge

📈

Portfolio

Tokens

NFTs

Activity

S

SepoliaETH

0.4894 SepoliaETH

Q

QKC

1000 QKC

+ Import tokens

↻ Refresh list

🗨 MetaMask support

Conclusion: We have succesfully created Metamask Wallet and Added dummy Ethers and Tokens.