Activity 4: ERC-20 token.

EIP- Ethereum Improvement Proposals and Ethereum Request for Comments

Ethereum Improvement Proposals (EIPs) describe standards for the Ethereum platform, including core protocol specifications, client APIs, and contract standards. It includes ERC (Ethereum Request for Comments) for Application-level standards and conventions, including contract standards such as token standards.

ERC-20 token.

ERC-20 is a standard for digital asset (currency, bonus points).

Tokens can be exchanged through smart contracts.

Simple to deploy.

Accepted by many cryptocurrency wallets, most Ethereum contracts are ERC-20 compliant.

ERC-20 Token specification.

Token creator must define **fields**:

Token name,

Token symbol,

Number of Tokens created,

Subdivisions

ERC – 20 standard defines **6 functions** which developers must implement:

TotalSupply, BalanceOf, transfer, transferFrom, approve, allowance.

These functions allow wallet app to interrogate user’s balance or transfer tokens to another user.

```
function totalSupply()
            public view returns (uint256);

function balanceOf(address tokenOwner)
            public view returns (uint);

function allowance(address tokenOwner, address spender)
            public view returns (uint);

function transfer(address to, uint tokens)
            public returns (bool);

function approve(address spender, uint tokens)
            public returns (bool);

function transferFrom(address from, address to, uint tokens)
            public returns (bool);
```

The **events** defined by ERC-20 are:

```
event Approval(address indexed tokenOwner, address indexed
spender,

      uint tokens);
event Transfer(address indexed from, address indexed to, uint
tokens);
```

Step 1: Define fields:

```
uint256 nbTokens;

mapping(address => uint256) balances;
mapping(address => mapping (address => uint256)) spendlimit;

string public name = 'Token optional BC';
uint8 public decimals = 0;
string public symbol = 'TOP';
```

Step 2: Define events and modifiers:

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


modifier checkBalance (address owner, uint tokens) {
        require(tokens <= balances[owner], 'Insufficient funds!');
        _;
}


modifier checkApproval (address owner, address delegate, uint tokens) {
    require(tokens <= spendlimit[owner][delegate], 'Insufficient allowance!');
        _;
}
```

Step 3: Set the total number of tokens and set the balance of the owner to the total number of tokens created:

```
constructor(uint256 tokens) {
   nbTokens = tokens;
   balances[msg.sender] = tokens;
}
```

Step 4: Get total supply:

```
function totalSupply() public view returns (uint256) {

      return nbTokens;

}
```

Step 5: Gat balance for an account:

```
function balanceOf(address tokenOwner) public view returns (uint) {

    return balances[tokenOwner];

}
```

Step 6: Implement transfer function:

```

function transfer(address receiver, uint tokens) public checkBalance (msg.send
er ,tokens)
                                                 returns (bool) {
    balances[msg.sender] = balances[msg.sender] - tokens;
    balances[receiver] = balances[receiver] + tokens;
    emit Transfer(msg.sender, receiver, tokens);
    return true;

}
```

Step 7: Set the number of tokens allowed to be transferred by a delegate.

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

Step 8: Implement the method that returns the number of tokens allowed to be transferred by a delegate:

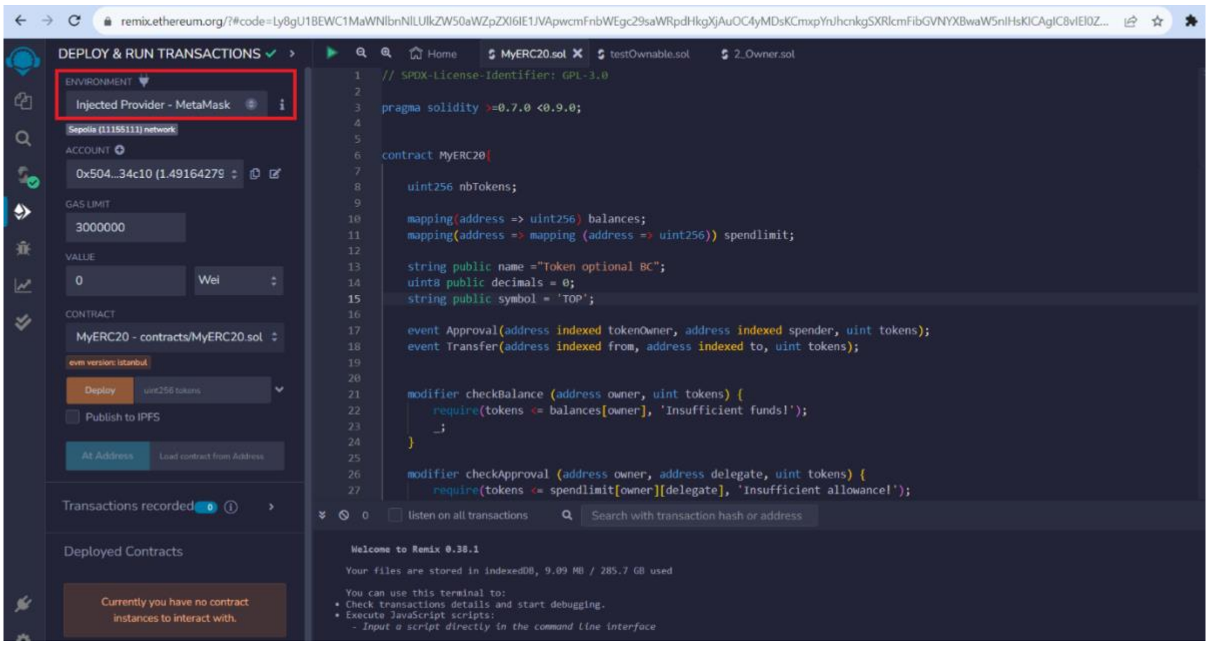
```
function allowance(address tokenOwner, address spender) public view
                                                  returns(uint) {
        return spendlimit[tokenOwner][spender];
}
```

Step 9: Implement the functions that transfers from another account, based on the maximum number of tokens allowed for transfer:

```
function transferFrom(address from, address to, uint tokens)
            public  checkBalance (from, tokens)
                    checkApproval(from, msg.sender, tokens) returns (bool) {

        balances[from] = balances[from] - tokens;
        spendlimit[from][msg.sender] = spendlimit[from][msg.sender]- tokens;
        balances[to] = balances[to] + tokens;
        emit Transfer(from, to, tokens);
        return true;
}
```

Step 10: Deploy on Sepolia network, with truffle or Remix IDE (Injected Web3) and Metamask. Find contract using contract address on EtherScan. Add tokens to Metamask.


remix.ethereum.org/?#code=Ly8gU1BEWC1MaWNlbnNlLUlkZW50aWZpZXI6IE1JVApwcmFnbWEgc29saWRpdHkgXjAuOC4yMDsKCmxpYnJhcnkgSXRlcmFibGVNYXBwaW5nIHsKICAgIC8vIE0Z...
DEPLOY & RUN TRANSACTIONS
Home
MyERC20.sol
testOwnable.sol
2_Owner.sol

ENVIRONMENT
Injected Provider - MetaMask
Sepolia (11155111) network
ACCOUNT
0x504...34c10 (1.49164279
GAS LIMIT
3000000
VALUE
0 Wei
CONTRACT
MyERC20 - contracts/MyERC20.sol
evm version: istanbul
Deploy uint256 tokens
Publish to IPFS
At Address Load contract from Address
Transactions recorded 0
Deployed Contracts
Currently you have no contract instances to interact with.

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;


contract MyERC20{

    uint256 nbTokens;

    mapping(address => uint256) balances;
    mapping(address => mapping (address => uint256)) spendlimit;

    string public name ="Token optional BC";
    uint8 public decimals = 0;
    string public symbol = 'TOP';

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


    modifier checkBalance (address owner, uint tokens) {
        require(tokens <= balances[owner], 'Insufficient funds!');
        _;
    }

    modifier checkApproval (address owner, address delegate, uint tokens) {
        require(tokens <= spendlimit[owner][delegate], 'Insufficient allowance!');
```

listen on all transactions
Search with transaction hash or address

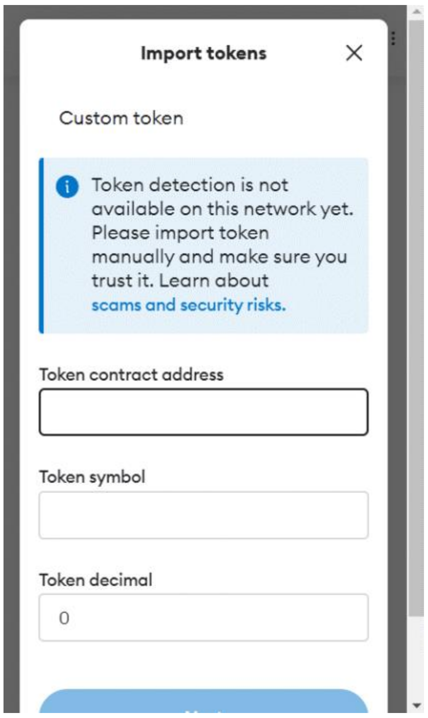
```
Welcome to Remix 0.38.1

Your files are stored in indexedDB, 9.09 MB / 285.7 GB used

You can use this terminal to:
• Check transactions details and start debugging.
• Execute JavaScript scripts:
  - Input a script directly in the command line interface
```

Deplyed contract: 0xf6C2A77A4873cEC7df03d51305888B52546FFD7b

0x821496b3B9880b9eA6e7C6aafb0268638501EE5A


Import tokens

Custom token

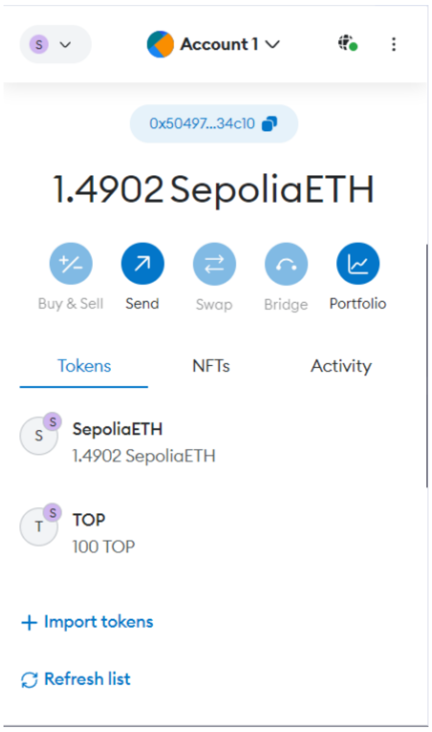
Token detection is not available on this network yet. Please import token manually and make sure you trust it. Learn about scams and security risks.

Token contract address

Token symbol

Token decimal

0

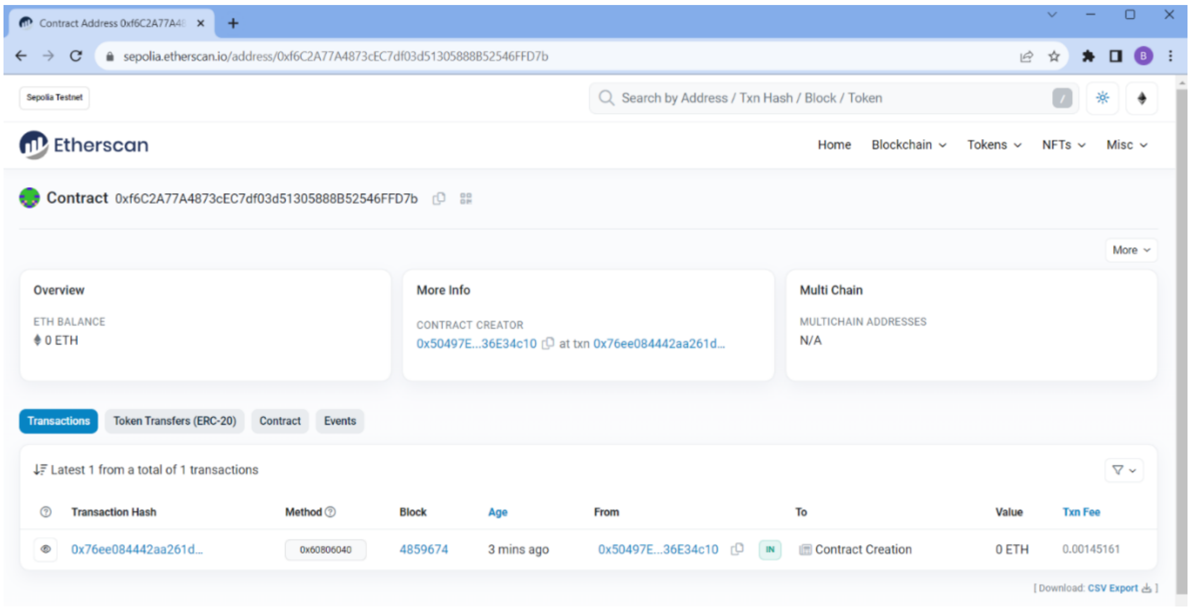

Account 1

0x50497...34c10

1.4902 SepoliaETH

SepoliaETH
1.4902 SepoliaETH

TOP
100 TOP


Contract Address 0xf6C2A77A48
Sepolia Testnet
Search by Address / Txn Hash / Block / Token
Etherscan
Home
Blockchain
Tokens
NFTs
Misc
More

sepolia.etherscan.io/address/0xf6C2A77A4873cEC7df03d51305888B52546FFD7b

Contract 0xf6C2A77A4873cEC7df03d51305888B52546FFD7b

Overview
ETH BALANCE
0 ETH

More Info
CONTRACT CREATOR
0x50497E...36E34c10 at txn 0x76ee084442aa261d...

Multi Chain
MULTICHAIN ADDRESSES
N/A

Latest 1 from a total of 1 transactions

Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
0x76ee084442aa261d...	0x60806040	4859674	3 mins ago	0x50497E...36E34c10	IN	Contract Creation	0 ETH	0.00145161

[Download: CSV Export]

Step 11: Transfer tokens to another Metamask account.

Truffle

npm modules

Check that Truffle is installed:

>> truffle –version

Create a new Truffle project in an empty folder:

>> truffle init

Install truffle-hdwallet-provider

>> npm install truffle-hdwallet-provider

Install dotenv

>> npm install dotenv

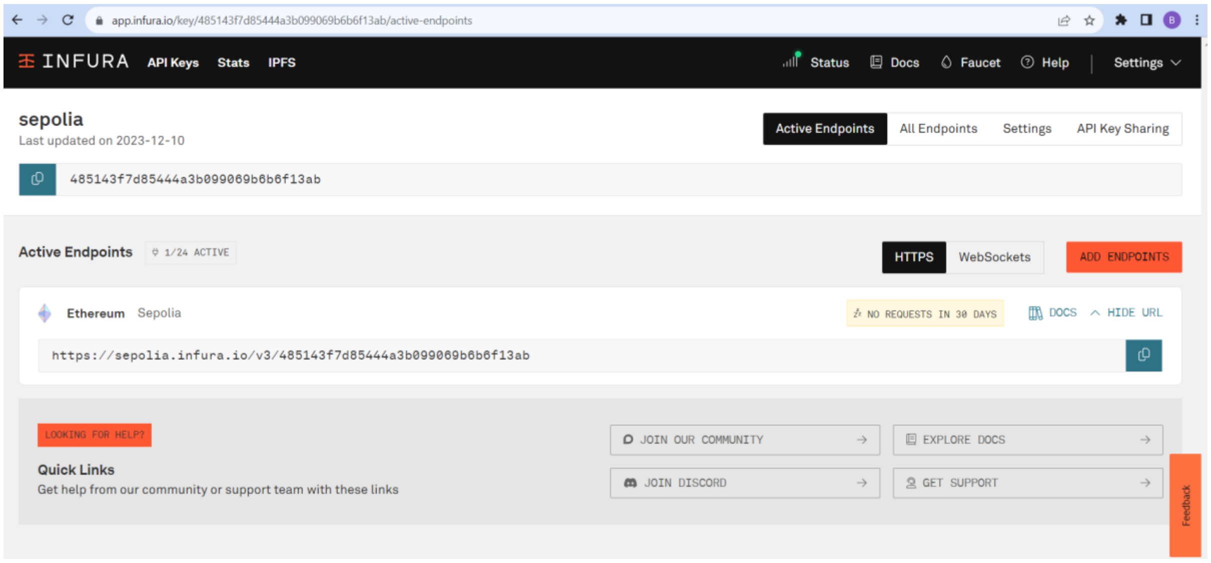
File structure

Add in /contract the file MyERC20.sol

Add file migrations/1_initial_migration.js

Infura config

Create an endpoint for Sepolia network with INFURA:



app.infura.io/key/485143f7d85444a3b099069b6b6f13ab/active-endpoints

INFURA API Keys Stats IPFS Status Docs Faucet Help Settings

sepolia
Last updated on 2023-12-10

Active Endpoints | All Endpoints | Settings | API Key Sharing

485143f7d85444a3b099069b6b6f13ab

Active Endpoints 1/24 ACTIVE

HTTPS | WebSockets | ADD ENDPOINTS

Ethereum Sepolia NO REQUESTS IN 30 DAYS DOCS HIDE URL

https://sepolia.infura.io/v3/485143f7d85444a3b099069b6b6f13ab

LOOKING FOR HELP?
Quick Links
Get help from our community or support team with these links

JOIN OUR COMMUNITY | EXPLORE DOCS | JOIN DISCORD | GET SUPPORT

Feedback

Add file .env and fill it with INFURA MNEMONIC and PROJECT_ID

MNEMONIC = "…"

PROJECT_ID = "…"

Deploy

Modify truffle-config.js:

```
require('dotenv').config();

const { MNEMONIC, PROJECT_ID } = process.env;

const HDWalletProvider = require('truffle-hdwallet-provider');


networks: {

sepolia: {

     provider: () => new HDWalletProvider(MNEMONIC, `https://sepolia.infura.io/v3/${PROJECT_ID}`),

      network_id: 11155111,      // Sepolia's id

      confirmations: 2,    // # of confirmations to wait between deployments. (default: 0)

      timeoutBlocks: 200,  // # of blocks before a deployment times out  (minimum/default: 50)

      skipDryRun: true     // Skip dry run before migrations? (default: false for public nets )

    }

}
```

Deploy the contract:

```
>> truffle deploy --network sepolia
```

Openzeppelin framework to write secure smart contracts.

Check IERC20, ERC20

```
>>    npm install openzeppelin-solidity
```

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

import {ERC20} from "openzeppelin-
solidity/contracts/token/ERC20/ERC20.sol";

contract MyOZERC20 is ERC20 {
    constructor(string memory name_, string memory symbol_,
uint256 initialSupply) ERC20(name_, symbol_) {
        _mint(msg.sender, initialSupply);
    }
}
```

truffle deploy --f 2 --network sepolia

Bibliography:

[1] https://eips.ethereum.org/

[2] https://github.com/ethereum/eips/issues/20