

ERC20 Token

BcTech Engineer Community Meetup #1

2018/07/07

今日の目的

- ERC20 に準拠した独自トークンを作成する
- ERC20 とは?
 - トークンの為の標準API
 - トークンを転送するための基本的な機能を提供する
 - トークンを承認して、他のチェーン上の第三者が使うことができるようにする
 - 標準化によってウォレットや交換所などのアプリケーションを再利用できる
- 想定
 - 初めてコントラクト開発を行う

私

- ソフトウェア/インフラエンジニア
 - 主にゲームのサーバサイド
 - アーケードゲーム
 - スマートフォンゲーム とかを作っています
- 非ブロックチェーン/非仮想通貨エンジニア
 - ブロックチェーン
 - Bitcoin のコードも読んだ事がない…

私

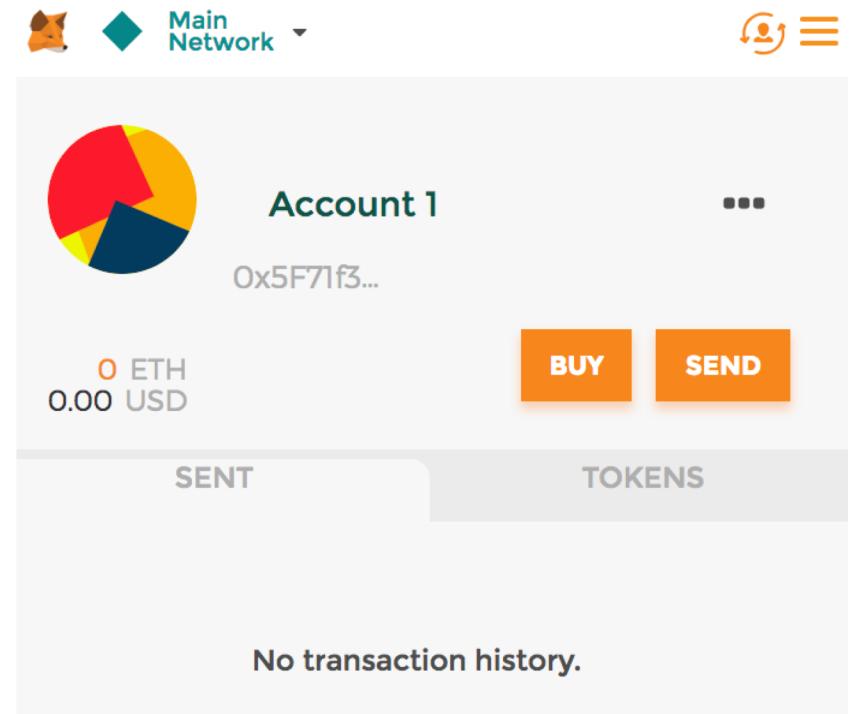
- ソフトウェア/インフラエンジニア
 - 主にゲームのサーバサイド
 - アーケードゲーム
 - スマートフォンゲーム とかを作っています
- 非ブロックチェーン/非仮想通貨エンジニア
 - ブロックチェーン
 - Bitcoin のコードも読んだ事がない…
- 初学者の初学者向けのハンズオンと聞いてきました 😕

作業の流れ

- Metamaskのインストール
- Etherを取得する
- ERC20 Token Standard について
- 実装, デプロイ
- トークンの送付
- 履歴の確認
- スライドで使うソースコード
 - <https://git.io/fNelk>

Metamaskのインストール

- Metamask
 - Chrome アドオンの Ethereum wallet
 - <https://metamask.io/>
 - ブラウザ上のEthereumアプリケーションを利用できるようになる

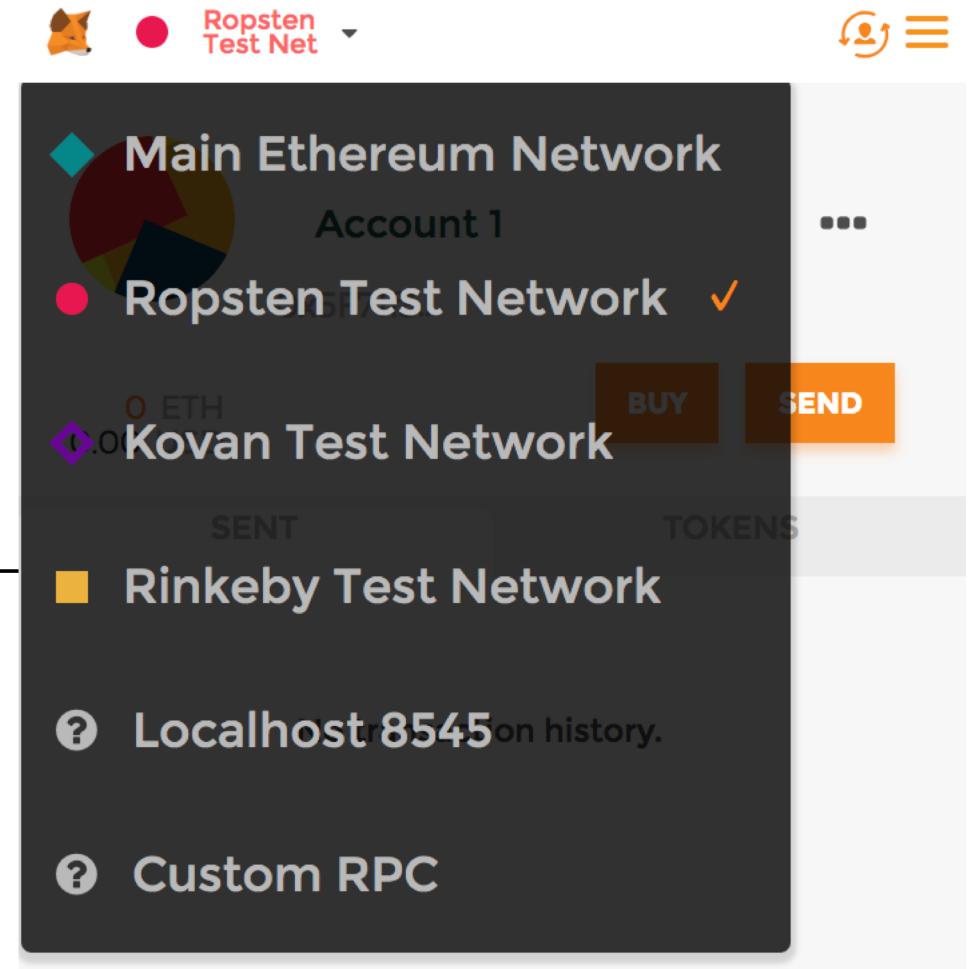


Ethereum ネットワーク

- メインネット
 - 所謂本番ネットワーク
- テストネット
 - 検証用ネットワーク
 - Ropsten
 - Rinkeby など
- プライベート・ネットワーク
 - 個人/または閉じた組織内でプライベートチェーンを立ち上げ

Ethereum ネットワーク

- メインネット
 - 所謂本番ネットワーク
- テストネット
 - 検証用ネットワーク
 - Ropsten
 - Rinkeby など
- プライベート・ネットワーク
 - 個人/または閉じた組織内でプライベー



Ether の受け取り

- Etherはトークンの生成, 送付などに必要
- Ethereum Ropsten Faucet から乞食る
 - <http://faucet.ropsten.be:3001/>

Ethereum Ropsten Faucet.

Enter your testnet account address
0x5f71f3a864301bada6cd138680e94e88b5f45eb0

Send me 1 test ether!

This faucet drips 1 Ether every 7 seconds. You can register your account in our queue. Max queue size is currently 5 . Serving balance 2767196 ETH)

The screenshot shows the Ethereum Ropsten Faucet interface. At the top, it displays the network as "Ropsten Test Net". Below this, there are two sections: "SENT" and "TOKENS".
SENT Section: Shows a pie chart with 0.00 ETH (0.00 USD) and a button labeled "SEND".
TOKENS Section: Shows a pie chart with 1.000 ETH (470.52 USD) and buttons labeled "BUY" and "SEND".
Account Section: Shows an account summary for "Account 1" with address 0x5F71F3... and a balance of 1.000 ETH (470.52 USD).
History Section: Shows a message "No transaction history."

ERC20 Standard Token

```
contract ERC20Interface {  
    function totalSupply() public constant returns (uint);  
    function balanceOf(address owner)  
        public constant returns (uint balance);  
    function transfer(address to, uint tokens)  
        public returns (bool success);  
    function transferFrom(address from, address to, uint tokens)  
        public returns (bool success);  
    function approve(address spender, uint tokens)  
        public returns (bool success);  
    function allowance(address owner, address spender)  
        public constant returns (uint remaining);  
  
    event Transfer(address indexed from, address indexed to, uint tokens);  
    event Approval(address indexed owner, address indexed spender, uint tokens);  
}
```

ERC20 Standard Token

function `totalSupply()` **public constant** `returns (uint)`;

合計供給量を返す

function `balanceOf(address owner)`
public constant `returns (uint balance)`;

ownerアカウントの残高を返す

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

toアカウントに指定したtokens分のトークンを送る

Transferイベントを発火させる必要がある

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

fromアカウントからtoアカウントに指定されたtokens分のトークンを送る

ERC20 Standard Token

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

spenderアカウントからtransferFromで送る際に指定したtokens分のトークンを送れるように承認する
Approvalイベントを発火する必要がある

```
function allowance(address owner, address spender)
    public constant returns (uint remaining);
```

spender アカウントが ownerアカウントから承認されるトークンの量を取得する

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

トークンが送られたタイミングで発火し「どのアドレスからどのアドレスへ送られた」という情報を出力する

```
event Approval(address indexed owner, address indexed spender, uint tokens);
```

承認が行われた承認情報を出力するイベント

上記に加えオプションとしてトークン名、トークンシンボルなどを指定する

実装 – 開発環境

- Remix
 - ブラウザベースのIDE
 - Ethereumノードの環境構築をせずに
コントラクトのコンパイル、デプロイが可能
 - <https://remix.ethereum.org/>

実装 – 開発環境

- Remix
 - ブラウザ
 - Ethereum コントラクト
 - <https://remix.ethereum.org>

The screenshot shows the Remix Solidity IDE interface. The left sidebar displays project files: `browser/ballot.sol` (selected), `browser/ballot_test.sol`, and `config`. The main editor area contains the `ballot.sol` source code:

```
1 pragma solidity ^0.4.0;
2 contract Ballot {
3     struct Voter {
4         uint weight;
5         bool voted;
6         uint8 vote;
7         address delegate;
8     }
9     struct Proposal {
10        uint voteCount;
11    }
12
13     address chairperson;
14     mapping(address => Voter) voters;
15     Proposal[] proposals;
16
17     /// Create a new ballot with _numProposals different
18     function Ballot(uint8 _numProposals) public {
19         chairperson = msg.sender;
20         voters[chairperson].weight = 1;
21         proposals.length = _numProposals;
22     }
23 }
```

A warning icon is visible near line 19. On the right, the interface includes a toolbar with **Compile**, **Run**, **Settings**, **Analysis**, **Debugger**, and **Support** buttons. A message box states: "Static Analysis raised 2 warning(s) that requires your attention. Click here to show the warning(s.)". A detailed warning message is shown in a yellow box: "browser/ballot.sol:19:5: Warning: Defining constructors
function Ballot(uint8 _numProposals) public {
^ (Relevant source part starts here and spans across". The bottom of the editor has tabs for "エディタ" (Editor) and "remix", with a status bar indicating "[2] only remix transactions, script".

実装

- サンプルプロジェクトを削除
 - ballot.sol, ballot_test.sol を削除
- ERC20を実装するトークンコントラクトを追加
 - ここでは MyToken とする

The screenshot shows a code editor interface with the following details:

- Toolbar:** Includes icons for file operations like new, open, save, and copy.
- Title Bar:** Displays the file path "browser/MyToken.sol" and standard window controls.
- File Structure:** On the left, there's a tree view:
 - browser** (expanded):
 - MyToken.sol** (selected)
 - config** (collapsed)
- Code Editor:** The main area contains the following Solidity code:

```
1 pragma solidity ^0.4.24;
2
3 contract MyToken {
4 }
```

ERC20 Standard Token (再掲)

```
contract ERC20Interface {  
    function totalSupply() public constant returns (uint);  
    function balanceOf(address owner)  
        public constant returns (uint balance);  
    function transfer(address to, uint tokens)  
        public returns (bool success);  
    function transferFrom(address from, address to, uint tokens)  
        public returns (bool success);  
    function approve(address spender, uint tokens)  
        public returns (bool success);  
    function allowance(address owner, address spender)  
        public constant returns (uint remaining);  
  
    event Transfer(address indexed from, address indexed to, uint tokens);  
    event Approval(address indexed owner, address indexed spender, uint tokens);  
}
```

実装

- **totalSupply**

```
//public 修飾子を指定すると getter が生成される  
// => function totalSupply() returns (uint)  
uint public totalSupply;
```

- **balanceOf**

```
// ハッシュテーブル  
// => function balanceOf(address owner) returns (uint)  
mapping(address => uint) public balanceOf;
```

- **transfer**

```
function transfer(address to, uint tokens) public returns (bool success) {  
....// msg.sender: 関数の呼び出し元のアドレス  
....return transferFrom(msg.sender, to, tokens);  
}
```

実装

- transferFrom

```
function transferFrom(address from, address to, uint tokens) public returns (bool success) {
    ...// 送信元が必要な量のトークンを所持しているか確認する
    ...require(balanceOf[msg.sender] >= tokens);

    ...

    ...// 呼び出し元と送信元が同じでない時、承認されているか確認する
    ...if (msg.sender != from) {
        ...
        ...require(allowance[from][msg.sender] >= tokens);
        ...allowance[from][msg.sender] -= tokens;
    }

    ...

    ...// トークンを移動する
    ...balanceOf[from] -= tokens;
    ...balanceOf[to] += tokens;

    ...

    ...// Transfer イベントを発火する
    ...emit Transfer(from, to, tokens);

    ...

    ...return true;
}
```

実装

- approve

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

- events

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

実装

- **name**
 - トークンの名前
- **symbol**
 - トークンのシンボル
- **コンストラクタ**
 - トークンのデプロイ時に name, symbol, トークンの所持数を指定できるようにする

```
// トークン名
string public name;
// トークンシンボル
string public symbol;
//public 修飾子を指定すると getter が生成される
// => function totalSupply() returns (uint)
uint public totalSupply;

constructor(string _name, string _symbol, uint tokens) public {
    name = _name;
    symbol = _symbol;
    ...
    balanceOf[msg.sender] += tokens;
    totalSupply += tokens;
}
```

デプロイ

- MyToken を Ropsten ネットワークにデプロイする
 - デプロイ時の引数で名前, シンボルなどを指定する

The screenshot shows the Remix IDE interface with the Solidity code for the `MyToken` contract on the left and the deployment configuration on the right.

Contract Code:

```
1 pragma solidity ^0.4.24;
2
3 contract MyToken {
4     // トークン名
5     string public name;
6     // トークンシンボル
7     string public symbol;
8     //public 修飾子を指定すると getter が生成される
9     // => function totalSupply() returns (uint)
10    uint public totalSupply;
11
12    constructor(string _name, string _symbol, uint tokens) {
13        name = _name;
14        symbol = _symbol;
15
16        balanceOf[msg.sender] += tokens;
17        totalSupply += tokens;
18    }
19
20    // ハッシュテーブル
21    // => function balanceOf(address owner) returns (uint)
22    mapping(address => uint) public balanceOf;
23
24 }
```

Deployment Configuration:

- Environment: Injected Web3 (Ropsten (3))
- Account: 0x5f7...45eb0 (1 ether)
- Gas limit: 3000000
- Value: 0 wei

Contract Selection: MyToken

Deployment Buttons: Deploy "ABCToken", "ABC", 1000

Logs: Transactions recorded: 0

Deployed Contracts: Currently you have no contract instances to interact with.

デプロイ

- パラメータを確認
 - 引数で与えた値と等しい
 - name
 - symbol
 - balanceOf(デプロイアカウント)
 - = totalSupply

Deployed Contracts

MyToken at 0xe9e...07ddc (blockchain)	
approve	address spender, uint256 tokens
transfer	トークンコントラクトのアドレス
transferFrom	address from, address to, uint256 tokens
allowance	address , address
balanceOf	0x5F71f3A864301bAdA6Cd138680e94eE
0:	uint256: 1000
name	
0:	string: ABCToken
symbol	
0:	string: ABC
totalSupply	
0:	uint256: 1000

デプロイ

- Etherscan でデプロイされたトークンコントラクトを確認する
 - <https://ropsten.etherscan.io/>

The screenshot shows the Etherscan interface for the Ropsten (Revival) TESTNET. The top navigation bar includes the Etherscan logo, the network name "ROPSTEN (Revival) TESTNET", a search bar, and menu options for HOME, BLOCKCHAIN, TOKEN, and MISC.

The main content area displays information for the token "ABCToken". The summary table on the left shows:

Summary	
Total Supply:	1,000 ABC
Holders:	0 addresses
Transfers:	0

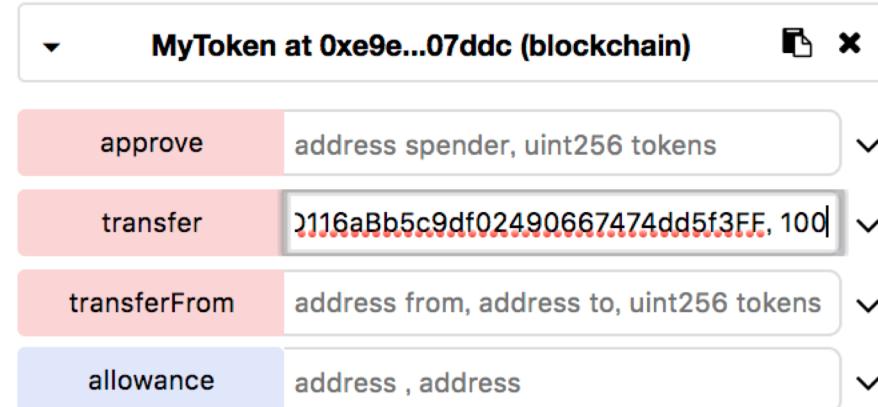
The right side of the screen provides detailed token metadata:

Rep	UNKNOWN ⓘ
Contract:	0xe9ebf0804c02e0fd2fa30489d5e1f3e5e707ddc
Decimals:	0
Links:	Not Available, Update ?

At the bottom, there is a "Filter By:" input field with placeholder "Enter Address / TxHash" and a "Apply" button.

トークンの送付

- トークンを作成したアカウントとは別のアカウントに送付する
 - 別の参加者 または Metamask でもう一つアカウントを作成する



トークンの送付

- トークンを作成したアカウントとは別のアカウントに送付する
 - 別の参加者 または Metamask でもう一つアカウントを作成する

 TOKEN ABCToken [i](#)

Home / TokenTracker / ABCToken

Summary Rep UNKNOWN [i](#)

Total Supply:	1,000 ABC	Contract:	0xe9ebf0804c02e0fd2fa30489d5e1f3e5e707ddc
Holders:	2 addresses	Decimals:	0
Transfers:	-	Links:	Not Available, Update ?

Filter By:

Token Transfers Holders Read Contract Write Contract [Beta](#)

 [TokenHolders Chart](#)

A total of 2 Token Holders First Prev Page 1 of 1 Next Last

Rank	Address	トーカン製作者	Quantity	Percentage
1	0x5f71f3a864301bada6cd138680e94e88b5f45eb0	送付先	900	90.0000%
2	0x0fee352bb3d116abb5c9df02490667474dd5f3ff		100	10.0000%

送付履歴の確認

TOKEN ABCToken 

[Home](#) / [TokenTracker](#) / [ABCToker](#)

Summary

Total Supply: 1,000 ABC

Holders: 2 addresses

Transfers: 1

Rep UNKNOWN 

Contract: [0xe9eebf0804c02e0fd2fa30489d5e1f3e5e707ddc](#)

Decimals: 0

Links: Not Available, [Update ?](#)

Filter By:

Enter Address / TxHash

Apply

Token Transfers

Holders

Read Contract

Write Contract Beta

 A Total of 1 event found

First

Prev

Page 1 of 1

Next

Last

TxHash	Age	From	To	Quantity
0xd7440ac29f2eb6...	10 mins ago	0x5f71f3a864301ba...	 0x0fee352bb3d116...	100

まとめ

- ERC20 で定められたインターフェースを実装したトークンコントラクトを作成した
- ERC20 を満たす事でEtherscanやMetamaskなど既存のサービスを容易に利用する事ができる
- ユーザーアカウント同士でトークンの送付ができる事を確認した
- TODO:
 - 今回は時間の都合上触れられなかつたが、ERC20 トークンを用いるサービスではapproveとtransferFromを利用する
 - 今回は供給量をデプロイ時に決めてしまっているが、発行体が発行できるようにするなど要件によって拡張する