Token-Ize Me - The Basics of ERC20 Tokens

Author	josh
Date	2019-07-11 12:55:46

Overview

One of the most interesting and powerful capabilities of the Ethereum network is the ability to create new currencies that operate *on top of* the base network. By using smart contracts, it's possible to create a currency called a *token* without having to fork an existing currency's code and create a new decentralized network. Coins such as Litecoin follow the latter approach - Litecoin exists as its own network that runs on a modified version of Bitcoin's code. However, tokens such as 0x (ZRX), Basic Attention Token (BAT), and Paxos (PAX) exist by leveraging the existing Ethereum network's smart contract capability.

However, there are so many tokens out there that it would be nearly impossible for wallet creators and exchanges to keep up with each individual token's smart contract API. In order to make it easier to create and support Ethereum tokens, the ERC20 standard was introduced. This standard specifies a common set of smart contract functions and events for a token. Let's take a look at this standard smart contract interface and how it works.

ERC20 Contracts

The Standard Contract Format

The ERC20 standard defines a set of *functions* and *events* that a token smart contract should define. The functions are:

```
function totalSupply() public view returns (uint);
function balanceOf(address tokenOwner) public view returns (uint balance);
function allowance(address tokenOwner, address spender) public view 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);
```

There are two events associated with the contract as well:

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

The list of required fields is fairly simple, but let's break down more how a contract like this works and summarize what each function does.

Token Contract Creation

Any time a smart contract is *deployed* on the Ethereum network, it becomes a permanent part of the blockchain that others can interact with by sending a transaction to the *contract address* with a function call in the *data* section of the transaction.

When a fixed-supply token contract is created (a common and simple type of token), the first thing it does is assign the total supply balance to the contract address owner, so the total amount of tokens is accounted for. Different types of tokens might allow a mining process where when a new Ethereum block is found, that miner also gets some of the tokens. But we're focused on simple fixed-supply contracts here, so assume when the contract is created the owner of the Ethereum contract address now owns the full supply of this token.

At any time, another user can see the total supply of tokens by creating a tx that calls `totalSupply`, and balances can be checked by calling `balanceOf` with an address.

Token Transactions

There are now two types of token transactions our smart contract interface allows. The first is pretty simple: it's the `transfer` functionality from the contract address owner to another recepient. Ethereum tokens use Ethereum addresses for sending and receiving, so it's simple in that regard. So to do a `transfer`, the owner creates a transaction on Ethereum that calls the transfer function, specifying an amount and the new owner.

The second type of transfer is a little more interesting and complex. The ERC20 standard allows a withdrawal type mechanism. First, the contract address owner must call `approve` in a transaction, specifying a new address that is allowed to withdraw and what the maximum withdrawal amount is. Then, the withdrawal address owner can call `allowance` to see their "account balance" and use `transferFrom` to withdraw from the approved account to a new address.

Finally, most contracts will *emit* an event on `Transfer` or `Approval` so that other contracts can listen and do something when the event is fired off. For example, we may have a contract that waits for approval to conduct a withdrawal of a token.

Tokens Made Simple

There's a lot going on when it comes to understanding Ethereum smart contract execution, but thankfully the ERC20 interface can be understood at a high level by discussing the basic functionality of each defined function and event.

With ERC20, we can create a token, and easily retrieve the *total supply* or an *address balance*. We can do a simple *transfer* out of an address we own, or *approve* withdrawals by another account that will *transfer from* the approved account. Finally, other contracts can listen for *approval* and *transfer* events so they can do something interesting.

For further reading, the Ethereum Wiki has a section on the ERC20 standard with a code example that may be useful. There are tons of tokens out there you may be interested in exploring as well.