EIP-20: Token Standard ‹›

Author	Fabian Vogelsteller, Vitalik Buterin
Status	Final
Type	Standards Track
Category	ERC
Created	2015-11-19

Table of Contents

Simple Summary

A standard interface for tokens.

Abstract

The following standard allows for the implementation of a standard API for tokens within smart contracts. This standard provides basic functionality to transfer tokens, as well as allow tokens to be approved so they can be spent by another on-chain third party.

Motivation

A standard interface allows any tokens on Ethereum to be re-used by other applications: from wallets to decentralized exchanges.

Specification

Token

Methods

NOTES:

- The following specifications use syntax from Solidity `0.4.17` (or above)
- Callers MUST handle `false` from `returns (bool success)`. Callers MUST NOT assume that `false` is never returned!

name

Returns the name of the token - e.g. `"MyToken"`.

OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present.

```
function name() public view returns (string)
```

symbol

Returns the symbol of the token. E.g. “HIX”.

OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present.

```
function symbol() public view returns (string)
```

decimals

Returns the number of decimals the token uses - e.g. `8`, means to divide the token amount by `100000000` to get its user representation.

OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present.

```
function decimals() public view returns (uint8)
```

totalSupply

Returns the total token supply.

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

balanceOf

Returns the account balance of another account with address `_owner`.

```
function balanceOf(address _owner) public view returns (uint256 balance)
```

transfer

Transfers `_value` amount of tokens to address `_to`, and MUST fire the `Transfer` event. The function SHOULD `throw` if the message caller's account balance does not have enough tokens to spend.

Note Transfers of 0 values MUST be treated as normal transfers and fire the `Transfer` event.

```
function transfer(address _to, uint256 _value) public returns (bool success)
```

transferFrom

Transfers `_value` amount of tokens from address `_from` to address `_to`, and MUST fire the `Transfer` event.

The `transferFrom` method is used for a withdraw workflow, allowing contracts to transfer tokens on your behalf. This can be used for example to allow a contract to transfer tokens on your behalf and/or to charge fees in sub-currencies. The function SHOULD `throw` unless the `_from` account has deliberately authorized the sender of the message via some mechanism.

Note Transfers of 0 values MUST be treated as normal transfers and fire the `Transfer` event.

```
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
```

approve

Allows `_spender` to withdraw from your account multiple times, up to the `_value` amount. If this function is called again it overwrites the current allowance with `_value`.

NOTE: To prevent attack vectors like the one described here and discussed here, clients SHOULD make sure to create user interfaces in such a way that they set the allowance first to `0` before setting it to another value for the same spender. THOUGH The contract itself shouldn't enforce it, to allow backwards compatibility with contracts deployed before

```
function approve(address _spender, uint256 _value) public returns (bool success)
```

allowance

Returns the amount which `_spender` is still allowed to withdraw from `_owner`.

```
function allowance(address _owner, address _spender) public view returns (uint256 remaining)
```

Events

Transfer

MUST trigger when tokens are transferred, including zero value transfers.

A token contract which creates new tokens SHOULD trigger a Transfer event with the `_from` address set to `0x0` when tokens are created.

```
event Transfer(address indexed _from, address indexed _to, uint256 _value)
```

Approval

MUST trigger on any successful call to `approve(address _spender, uint256 _value)`.

```
event Approval(address indexed _owner, address indexed _spender, uint256 _value)
```

Implementation

There are already plenty of ERC20-compliant tokens deployed on the Ethereum network. Different implementations have been written by various teams that have different trade-offs: from gas saving to improved security.

Example implementations are available at

- OpenZeppelin implementation
- ConsenSys implementation

History

Historical links related to this standard:

- Original proposal from Vitalik Buterin: https://github.com/ethereum/wiki/wiki/Standardized_Contract_APIs/499c882f3ec123537fc2fccd57eaa29e6032fe4a
- Reddit discussion: https://www.reddit.com/r/ethereum/comments/3n8fkn/lets_talk_about_the_coin_standard/
- Original Issue #20: https://github.com/ethereum/EIPs/issues/20

Copyright

Citation

Please cite this document as:

Fabian Vogelsteller, Vitalik Buterin, "EIP-20: Token Standard," *Ethereum Improvement Proposals*, no. 20, November 2015. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-20.