---

title: ERC-20 Token Standard description: lang: en

---

## Introduction {#introduction}

### What is a Token?

Tokens can represent virtually anything in Ethereum:

- reputation points in an online platform
- skills of a character in a game
- lottery tickets
- financial assets like a share in a company
- a fiat currency like USD
- an ounce of gold
- and more...

Such a powerful feature of Ethereum must be handled by a robust standard, right? That's exactly where the ERC-20 plays its role! This standard allows developers to build token applications that are interoperable with other products and services.

### What is ERC-20?

The ERC-20 introduces a standard for Fungible Tokens, in other words, they have a property that makes each Token be exactly the same (in type and value) as another Token. For example, an ERC-20 Token acts just like the ETH, meaning that 1 Token is and will always be equal to all the other Tokens.

## Prerequisites {#prerequisites}

- Accounts
- Smart Contracts
- Token standards

## Body {#body}

The ERC-20 (Ethereum Request for Comments 20), proposed by Fabian Vogelsteller in November 2015, is a Token Standard that implements an API for tokens within Smart Contracts.

Example functionalities ERC-20 provides:

- transfer tokens from one account to another
- get the current token balance of an account
- get the total supply of the token available on the network
- approve whether an amount of token from an account can be spent by a third-party account

If a Smart Contract implements the following methods and events it can be called an ERC-20 Token Contract and, once deployed, it will be responsible to keep track of the created tokens on Ethereum.

From EIP-20:

### Methods {#methods}

```
solidity function name() public view returns (string) function symbol() public view returns (string) function
decimals() public view returns (uint8) function totalSupply() public view returns (uint256) function
balanceOf(address _owner) public view returns (uint256 balance) function transfer(address _to, uint256 _value)
public returns (bool success) function transferFrom(address _from, address _to, uint256 _value) public returns
(bool success) function approve(address _spender, uint256 _value) public returns (bool success) function
allowance(address _owner, address _spender) public view returns (uint256 remaining)
```

### Events {#events}

```
solidity event Transfer(address indexed _from, address indexed _to, uint256 _value) event Approval(address indexed _owner, address indexed _spender, uint256 _value)
```

### Examples {#web3py-example}

Let's see how a Standard is so important to make things simple for us to inspect any ERC-20 Token Contract on Ethereum. We just need the Contract Application Binary Interface (ABI) to create an interface to any ERC-20 Token. As you can see below we will use a simplified ABI, to make it a low friction example.

#### Web3.py Example {#web3py-example}

First, make sure you have installed Web3.py Python library:

```
pip install web3
```

```python from web3 import Web3

w3 = Web3(Web3.HTTPProvider("https://cloudflare-eth.com"))

dai_token_addr = "0x6B175474E89094C44Da98b954EedeAC495271d0F" # DAI weth_token_addr = "0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2" # Wrapped ether (WETH)

acc_address = "0xA478c2975Ab1Ea89e8196811F51A7B7Ade33eB11" # Uniswap V2: DAI 2

# This is a simplified Contract Application Binary Interface (ABI) of an ERC-20 Token Contract.

# It will expose only the methods: balanceOf(address), decimals(), symbol() and totalSupply()

simplified_abi = [ { 'inputs': [{'internalType': 'address', 'name': 'account', 'type': 'address'}], 'name': 'balanceOf', 'outputs': [{'internalType': 'uint256', 'name': '', 'type': 'uint256'}], 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [], 'name': 'decimals', 'outputs': [{'internalType': 'uint8', 'name': '', 'type': 'uint8'}], 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [], 'name': 'symbol', 'outputs': [{'internalType': 'string', 'name': '', 'type': 'string'}], 'stateMutability': 'view', 'type': 'function', 'constant': True }, { 'inputs': [], 'name': 'totalSupply', 'outputs': [{'internalType': 'uint256', 'name': '', 'type': 'uint256'}], 'stateMutability': 'view', 'type': 'function', 'constant': True } ]

dai_contract = w3.eth.contract(address=w3.to_checksum_address(dai_token_addr), abi=simplified_abi) symbol = dai_contract.functions.symbol().call() decimals = dai_contract.functions.decimals().call() totalSupply = dai_contract.functions.totalSupply().call() / 10**decimals addr_balance = dai_contract.functions.balanceOf(acc_address).call() / 10**decimals

# DAI

print("===== %s =====" % symbol) print("Total Supply:", totalSupply) print("Addr Balance:", addr_balance)

weth_contract = w3.eth.contract(address=w3.to_checksum_address(weth_token_addr), abi=simplified_abi) symbol = weth_contract.functions.symbol().call() decimals = weth_contract.functions.decimals().call() totalSupply = weth_contract.functions.totalSupply().call() / 10**decimals addr_balance = weth_contract.functions.balanceOf(acc_address).call() / 10**decimals

# WETH
```

print("===== %s =====" % symbol) print("Total Supply:", totalSupply) print("Addr Balance:", addr_balance) ```

## Further reading {#further-reading}

- EIP-20: ERC-20 Token Standard
- OpenZeppelin - Tokens
- OpenZeppelin - ERC-20 Implementation
- Alchemy - Guide to Solidity ERC20 Tokens