ERC-20 Token Standard

ERC-20 is the most popular standard and has widely been used both for ICOs and STOs. It provides the general functionality for a token in order to manage the balance of account holders and enable transfers.
In addition to the chapter Token Standards of Assets on Blockchain, this document provides a detailed description of the methods of an ERC-20 token.

The ERC-20 functions

The ERC-20 token consists of six different functions as described here. The words within the brackets are the parameters that need to be set in order to perform the action. For example, you need to set the receiver and the number of tokens to execute the *transfer* method.

- **totalSupply()**
  When this function is called, it returns the token supply (*How many units of this token exist?)* of a specific ERC-20 token.

- **balanceOf(*tokenOwner*)**
  Manages token balance in every Ethereum wallet. When this function is called, it returns the balance (amount of tokens) of the wallet address that is entered as the parameter.

- **transfer(*to*, *tokens*)**
  When this function is called, the Smart Contract checks if the sender has enough tokens for the transfer according to the second parameter *tokens*. If the condition is met, the amount (as defined by the parameter *tokens*) is subtracted from the balance of the sender and is added to the balance of the receiver (parameter *tokens*). The address of the sender is automatically known by the Smart Contract without setting an own parameter since it is the executor of the Smart Contract. However, it gets complicated when you are using this function for more than a simple transfer. If you want to send tokens to a more complicated Smart Contract that needs to process the senders address, he might not be able to access that information. This is the reason why there is another *transferFrom* method as described below.

- **transferFrom(*from*, *to*, *amount*)**

  When this function is called, the Smart Contract checks if the sender has enough tokens for the transfer according to the second parameter *tokens*. If the condition is met, the amount (as defined by the parameter *tokens*) is subtracted from the balance of the sender and is added to the balance of the receiver (parameter *tokens*). In contrast to the transfer function, this function explicitly transmits the senders address (parameter *from*) as a parameter, so it can be processed by a Smart Contract for any purpose.

- **approve(spender, amount)**
  Smart Contracts can be very complex and they can e.g. perform token transfers on behalf of the party who called the Smart Contract. You can imagine a user Bob and a Smart Contract called RealEstateInvestmentContract. This Smart Contract shall automatically invest the tokens of Bob into real estate when a lucrative property is found. Therefore, Bob needs to allow the Smart Contract to access his funds and use his tokens within certain boundaries. He does that by executing the function approve with two parameters: spender is the address of the party who should be given the permission to withdraw from the account and *amount* is the maximum amount is the maximum amount of tokens that can be withdrawn.

- **allowance()**
  After the function approve was called, this function allowance returns the amount of tokens that the spender is allowed to withdraw. The function does not change any values or perform any actions, it only returns the value.