6 | Fungible Tokens

6.1 INTRODUCTION

Blockchain is all about keeping record of assets. A token is a **digitally transferable asset** that is issued and recorded on the blockchain. These assets can represent money, ownership, reputation points, art, electricity kWh, and many other things. Ether (ETH), the native token of Ethereum is an example of a token: it is a digital asset that has a certain utility[1] , and you can freely transfer your ether to other users if you want to.

To accomodate different types of assets, Ethereum uses distinct token-types identified by ERC-codes. ERC stands for Ethereum Request for Comments, and is a technical document that describes a standard or convention at the application level of Ethereum. ERC proposals get a number, for instance, ERC-20 is the twentieth improvement proposal in the EIP[2] repository within the ERC category. ERC proposals encompass a.o., name registries, libraries, wallets, and, of course token standards which will be our main focus in this chapter. It is perfectly possible for a blockchain project to issue tokens that are not ERC-compliant, but doing so is disadvantageous as it would lessen interoperability, which is key to platform and ecosystem growth.

By following the ERC standards, projects ensure that their tokens are compatible with crypto-wallet software, can be exchanged for other tokens, or used as lego-blocks in other protocols. Furthermore, these standards are by now battle-tested as trillions of dollars have passed (virtual) hands through them.

[1]You use ether to perform transactions on the blockchain or as collateral for financial transactions.

[2]Ethereum Improvement Proposal

For a full list of ERC proposals and their technical documentation, we refer the reader to the Ethereum EIP overview. The table below shows a quick summary of the tokens that we will be covering in this book:

Standard	Usage
ERC-20	Fungible, transferable tokens
ERC-777	Improved version of ERC-20
ERC-721	Royalty standard for NFTs
ERC-1155	Multiple-token standard
ERC-4626	Yield-bearing token

6.2 THE ERC-20 STANDARD

WHAT IT IS

ERC-20 is the most important token standard on Ethereum. It describes a convention of how to represent digital assets that are \emph{fungible}. A fungible asset is an asset that has no unique identity associated with it. It can be divided, interchanged, and re-combined without affecting the value of the asset. Common examples of assets that are fungible include sovereign currencies such as dollars, crypto-native currencies such as ether, and bitcoin, and alternative existing currencies such as airline miles. Convince yourself that these are indeed fungible: splitting 10 dollars into 10 one dollar bills does not reduce its value. Trading one 10 dollar bill for another one does not change its value either.

Examples of ERC-20 tokens in the wild include:

- The BAT (basic attention) coin is a token used by the Brave browser for measuring user attention. When advertisers want to capture some of the user's attention, they can purchase BAT on an exchange.

- Helium is a blockchain-based network for IoT devices that allows users to share their internet connectivity to passerbys. In doing so they earn HNT token which can be converted into credits for bandwidth.

- Energy market tokens like LO3's Exergy token allow users to receive tokens for renewable energy that they generate but do not need at the time of generation. They can exchange these tokens for energy at a later time when they do need the energy.

- The DAI stablecoin is a coin that is designed to follow the price of the USD. It serves to reduce volatility in users' portfolios and it thereby bridges the gap between fiat and cryptocurrency. We will talk more about stablecoins in the next chapter.

Another dominant use-case of ERC20's are "governance tokens" : fractional shares of an asset or a decentralized company. Holders of such tokens are often imbued with some type of share of the revenue as well as voting rights (both proportional to the number of tokens owned). Even so, they are distinct from traditional equity shares as they are often traded outside of a regulatory framework and do not convey the same rights.

ERC-20 contracts are incredibly popular and their existence has led to the ICO (Initial Coin Offering) boom of 2017 where every Joe would create their own project token and then sell it for the promise of participation in the future of the project. We will talk more about token valuation and ICO's in a later chapter, but needless to say many of these ICO's ended-up being pump-and-dump schemes where users would invest in tokens that ended up being worthless.

Over 242,000 ERC20-compatible tokens exist at the time of writing, some with market caps in the range of billions of dollars, others with only 1 real user. In fact, by the end of this chapter and its hands-on exercises, you should be able to create your own token as well.

THE STANDARD

The ERC-20 standard specifies, among other things, the following functions which must be implemented:

- Providing basic information such as the name, symbol, number of decimals, and the total supply.

- Allow retrieving of the balance/allowance of an address (wallet or contract).

- Approving/transferring money from one address to another.

> **Example:** DAI is a stable-coin that represents a virtual dollar. It is implemented as an ERC20 contract which you can read on Etherscan by searching for DAI or going to the contract address directly 0x6b175474e89094c44da98b954eedeac495271d0f.
>
> You can find out what the name and total supply are by clicking on the relevant buttons on etherscan under "Contract" - "Read Contract". We can for instance, tell that the total supply of DAI is capped at a whopping 34,585,368,686,992,979,253,32,481,973.

6.3 HOW TO USE TOKENS

MINTING TOKENS

A token starts by the developers uploading an ERC20 smart contract with its specification. Once it is appended to the blockchain, tokens can be issued through a process of *minting*. The developers are free to specify who is allowed to mint tokens and how minting is implemented, but often it is a simply matter of increasing a counter variable in the smart contract to indicate that the total supply has gone up and then assign the additional supply to a wallet address. Before you interact and/or pay for a token a prudent first step is to check who has these special minting privileges as they have complete power over the token's supply and - therefore - its price.

TRANSFERING TOKENS

Users can tranfer tokens to other users by telling the ERC20 smart-contract of that specific token to do so. The smart-contract will then first check whether or not you have the necessary balance, after which it decreases your balance and increases the other user's balance. Sometimes, you will see people sending tokens to special addresses such as the zero-address 0x0000...0. These special addresses are not owned by anyone and are therefore the equivalent of "destroying" currency. This - usually intentional - reduction of the supply of the token is called *burning*. Similar to traditional money, anyone can burn their own tokens and sometimes - unintentionally - tokens get lost when they are sent to the wrong address. The difference between a controlled burn and the latter case is that the latter will not show-up as a supply reduction where as the first can be accounted for.

ALLOWING SMART CONTRACTS TO USE YOUR FUNDS

When you use DeFi products like a decentralized exchange, you often temporarily need to authorize the platform that you are interacting with to manage some of your funds. This is called *approving* and its crucial that you understand the implications of doing so. Consider for instance the scenario shown in Figure 1 where we are trade some of our BAT tokens to MANA tokens using Uniswap. Before Uniswap is allowed to perform this operation for us, we need to allow Uniswap to use our tokens.

When you approve a contract you are telling the ERC20 of the token (BAT in this case) that you allow the other smart-contract (Uniswap's router in this case) to transfer up to a certain amount of tokens (the *allowance*) from your wallet. With the same click, you are also giving the smart-contract information about how many of the tokens you currently own (the *proof-of-ownership*). This ensures that the smart-contract cannot transfer more than you allow it to, and that users cannot try to exchange more tokens than they own. This is step 1 in the process shown in Figure 2. After the approval, you can initiate the trade by pressing swap. Upon doing so, you are asking Uniswap's router contract to execute the swap. It will then request the required tokens (BAT) from the ERC20. The ERC20 will check whether or not Uniswap has the allowance (it does because we approved it) and then send the tokens to Uniswap[3].

[3] In reality, Uniswap may actually request to spend them somewhere else as well instead of sending the tokens to the Uniswap smart contract.

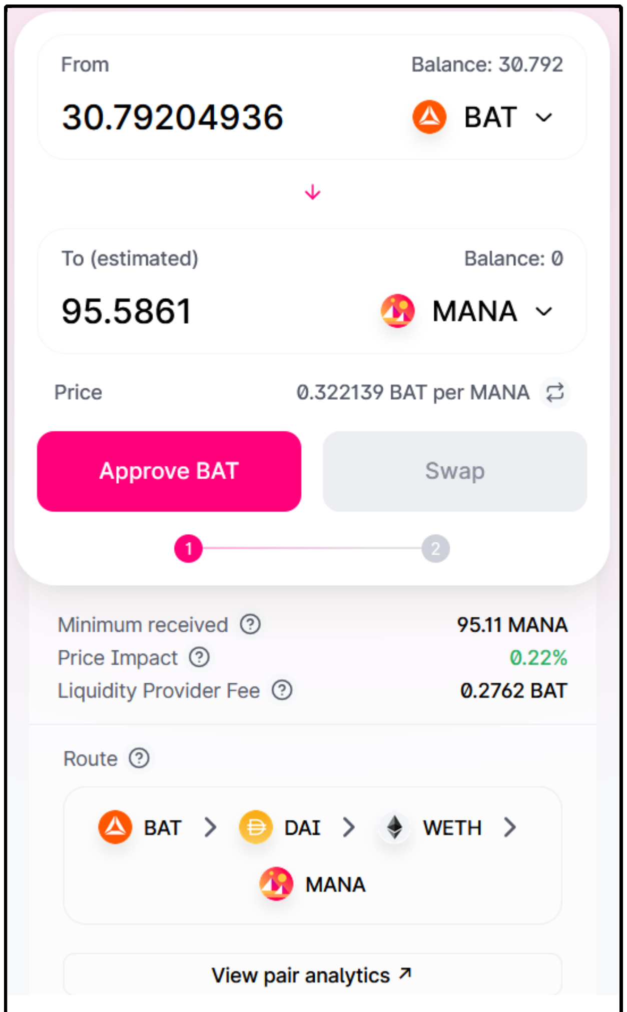
From
Balance: 30.792
30.79204936 BAT
To (estimated)
Balance: 0
95.5861 MANA
Price 0.322139 BAT per MANA
Approve BAT
Swap
Minimum received 95.11 MANA
Price Impact 0.22%
Liquidity Provider Fee 0.2762 BAT
Route
BAT > DAI > WETH > MANA
View pair analytics ↗

Figure 1: Swapping one token to another on Uniswap requires an approval first.

6.4 CAVEATS

INFINITY APPROVALS

Some smart-contracts will automatically (or manually) request "*infinite approvals*". With such approvals, you are setting the maximum spending limit for the smart contract to infinity, thereby bypassing the need to re-approve the smart contract every time to use it. For small amounts of tokens in trusted projects that you use every day, this is probably not too big of an issue as it can save you some gas fees. If you are dealing with large values of tokens it is strongly advisable that you do not infinity-approve. If an attacker finds a flaw that can result in unauthorized transfer of tokens in the smart-contract, they may very well end-up transferring all your tokens away! This is what happened in the Bancor attack causing a loss of funds of 23.5 million USD and news of similar hacks appear every month or so. Approvals typically cost very little so it's almost always worth it to re-approve contracts every time you use them. Alternatively, you can also revoke infinity-approvals after a while of usage via "token allowance checkers" like TAC[4].

WRAPPED ETHER

When Ethereum was created, ERC20s did not exist yet. In order to be fully compatible with smart contracts, Ethereum developers created *wrapped ether* (symbol: WETH). It is guaranteed to always have a 1-to-1 mapping to ether by design. During your DeFi journey, you will probably also encounter other wrapped tokens such as *wrapped bitcoin* (symbol: WBTC), these allow you to interact with crypto-currencies that are not native to the Ethereum chain.

RECEIVING UNKNOWN TOKENS

While token transfers out of your wallet require your approval, token transfer into your wallet do not. This means that you could be sent random tokens that you did not request. This can be great in the case of an airdrop where a protocol gifts you protocol tokens as a reward for using them. Even so, there are some scams where the fraudulent project will send you tokens. When you try to redeem or exchange these tokens, they will instead try to trick you into you approving to transfer your funds to the scam. You should therefore never interact with a token before doing your due diligence. Moreover, be extra careful with approvals on unknown websites and double check what you are approving.

[4]`https://tac.dappstar.io/`

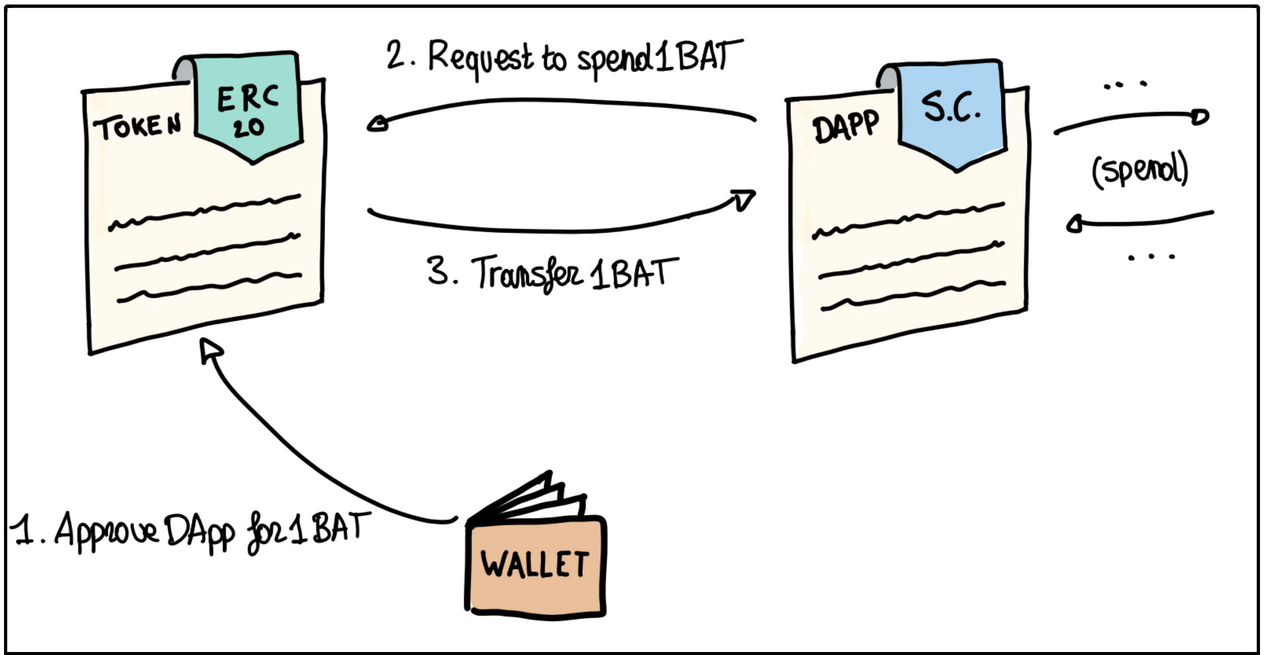

Figure 2: The approval-spend process of ERC20's.

NAMES VS CONTRACTS

The ERC20 standard requires that developers specify the name and symbol of their tokens, but there is no provision to avoid duplicates. A well-known scam is to create false copies of a legit token and having non-suspecting users buy these worthless tokens. *You should always verify for yourself that the token that you are buying/transferring has the correct contract-address when doing the transfer*, this is the only way to ensure that you are really getting what you paid for! Another way to avoid being scammed is to only transact via reputable Dapps. As many clones are made under similar addresses (e.g., www.btcoin.com instead of www.bitcoin.com), it is advisable that you always triple check the address of the website that you are visiting and bookmark them in your browser to avoid mistyping them.

CENTRALIZATION AND RUG PULLS

While Ethereum is decentralized, tokens are not necessarily decentralized. If a project wants to build in a mechanism that allows the project owner to disable or transfer the tokens away from you, they will be able to do so. Most tokens that work with fiat money and want to be compliant with local laws need to build-in such functionality as regulators love the ability to freeze user's funds.

Example: Centre, the organization behind Coinbase issues one of the largest stablecoins called USDC. Just like DAI it represents a virtual dollar, but unlike DAI it is not decentralized. When you read the contract over at etherscan or their open their github, you will notice that it has a black-listing functionality built-in. If a user is blacklisted (by Centre), then that user is no longer able to transfer her tokens.

In July 2020, Circle enforced this blacklist to a couple of addresses thereby seizing more than 100,000 USD from the wallet owners. In a reply toTheBlock's journalists, they stated that "\emph{it blacklisted an address in response to a request from law enforcement. While we cannot comment on the specifics of law enforcement

> requests, Centre complies with binding court orders that have appropriate jurisdiction over the organization}".
>
> **Example2:** EURS is a stablecoin that represents virtual euros. It too needs to blacklist tokens to be in compliance with local laws in Malta. Their contract implements the transfer function in such a way that in the future a taxable fee can be made to users whenever they want to transfer their tokens. When I reached out to them, they stated that they had "no such functionality" in their contracts and were not planning to have it in the future. The contract reads otherwise.

Malicious examples exist where project owners have used such functionalities to transfer all funds from their project's ERC20 contracts to themselves. Furthermore, centralization features incite abuse from bad actors in the network. This moral hazard is well-documented and you should consider buying into tokens only after a abundance of caution and scrutiny on your part. An important part of DYOR is therefore to ensure that no such mechanisms exist within the token that you are buying, or to at least be aware of them.

In conclusion, the only way to be sure that a token is truly decentralized and beyond the control of others at the application layer is to read the contract (or, alternatively read trusted sources that describe the protocols and their associated risks).

6.5 ERC777 IMPROVEMENTS

ERC-20s work well, but over time it became clear that they lacked some important features. ERC777 brings many quality-of-life improvements and the concept of "hooks" and operators to tokens. **Operators** enable users to allow smart contracts to send tokens on their behalf. Hooks are actions that trigger when certain events happen (i.e., when tokens are sent, minted, burned, or when operators are authorized). This allows users to perform the approval and transfer action atomically (in one go), and forces developers to build in extra checks on token transfers that ensure not tokens get lost. ERC777 contracts are backward compatible with the ERC20 standard, but cost a little bit more gas to use.

6.6 ERC-X

Many different types of ERC's have been proposed for fungible tokens, most of which are self-explanatory once you decide to read the relevant documentation. When you encounter exotic tokens or standards, I recommend that you take a look at the ethereum website to see what they are used for. Notable examples that we will not cover in depth here include:

Standard	Usage
ERC-223	ERC-20 with extra checks to avoid loss of tokens, reduces gas cost
ERC-827	Like ERC-223 but also allows the inclusion of data with transactions.
ERC-621	Like ERC-20, but allows for the supply to increase and decrease.

EXERCISE

Check out Ethereum's documentation for ERC20's: `https://ethereum.org/en/developers/docs/standards/tokens/erc-20/`.