

The Orderbook DEX

What is *The Orderbook DEX*?

The *Orderbook DEX* project's main objective is to develop **viable** and **fully decentralized** exchanges with **orderbook functionality** that work on the **EVM** (Ethereum Virtual Machine).

- By **viable** we mean not only its **practical viability** but also its **economic viability**.

▮ The gas cost of using it should be **predictable** and **reasonable**.

- By **fully decentralized** we mean in particular that it should **not depend** on a third party or organization (either centralized or decentralized) to operate.
- By **orderbook functionality** we mean that it should work as similar as possible as a **traditional orderbook** exchange.

▮ **Liquidity** should be provided **organically** by the market, and orders are filled **honoring the order** in which they were placed.

How will it be implemented?

- **Several versions** of *The Orderbook DEX* are to be developed, where each major version is not necessarily an extension, improvement, or upgrade of the previous one, but an **entirely new approach** to the same problem.
- The **objective** is not to provide direct replacements of the previous version but **alternatives**, each with their own **balancing of features** and aimed at different market players and use cases.
- **Exchanges** running different versions can either indirectly affect each other through **arbitrage** or be made to work together as one exchange through an **integration interface**.

Where can I learn more about...

- [...The Orderbook DEX V1?](#)
- [...The Orderbook DEX V2?](#)
- [...The Orderbook DEX V3?](#)
- [...The Orderbook DEX Client?](#)
- [...the utility token?](#)
- [...the Roadmap?](#)
- [...The Orderbook DEX Team?](#)
- [...how to stay in the loop?](#)

The Orderbook DEX V1

Features

- It allows **Limit Orders**.
- It requires **no additional fee** to function (other than gas fees).
- Orders **at market** are executed **immediately** when possible, working similarly to a **swap**.
- Almost all functions are **O(1) complexity**.
 - This means that the **gas needed** for these functions is always equal or below a predetermined **constant** value.
- Order ownership can be **transferred**.
 - This also means that an order can be wrapped as an **NFT**.

Trade-offs

- It **only** allows Limit Orders.
- Order placers must **lock** the funds to be traded.
- **Canceling** an order is **O(n) complexity**.
 - Where **n** is relative to the amount of **orders after yours**.
 - Which means the more orders after yours, the **more gas** will be needed to cancel it.

Considerations

- Keep in mind that the funds you are locking in for trading might remain locked **for a long time**.
- It might take some time for people to start canceling orders from a price point that has **moved away from market**, which would give you a chance to cancel for a **viable gas cost**.
- Since order ownership can be transferred, you might be able to exit a position by **trading your order**, either directly or by wrapping the **order as an NFT**.
- [The Orderbook DEX V2](#) might be a better option if you are interested in **short-term trading**.

More about V1

- [How does V1 work?](#)
- [Why is V1 designed the way it is?](#)
- [What is planned for V1?](#)

How does V1 work?

The basics, explained in under 5 minutes

- *The Orderbook* keeps a record of `total contracts placed` on a price point.
- It also keeps a record of `total contracts filled`.
- Each order keeps a record of the amount of `contracts placed before the order`.
- If the `total contracts filled > contracts placed before the order`
 - The order has been **filled**.
 - The difference between the values indicates **how much** of it has been filled.

Regarding canceling orders

- For an order to be canceled, orders placed after it **must** also be updated to keep the system **consistent**.
- Each order keeps track of the `previous order` and the `next order`.
 - ▮ This is what is commonly referred to as a `doubly linked list`.
- When an order is canceled, it can be **deleted** and the `previous order` and the `next order` updated accordingly.
 - ▮ Search `doubly linked list deletion` for details on how this work.
 - ▮ Deletion in a `doubly linked list` is **O(1)** and doesn't change the order of items.
- This way other orders that wish to be canceled don't have to go through **all orders** after it, only the **remaining ones**.
 - ▮ Transversing a `doubly linked list` is still **O(n)**, but this way **n** can shrink.
 - ▮ If orders start being canceled from the tail of the queue upwards, everyone eventually gets a chance to cancel at a reasonable gas cost.

Why is V1 designed the way it is?

Design Philosophy

- The first release of **V1** was designed with **readability** as a top priority.

The key here is to prioritize the ease of understanding of *The Orderbook's* logic.

- **V1** was designed expecting that its core functions will be called **a lot**.

- Therefore, optimizations favor reduction of gas for these functions.

This is why **V1** does not use **proxies** or **libraries** to reduce the cost of deploying a new orderbook.

- Some optimizations were disregarded in an attempt to avoid **premature optimization**.

We had no way of knowing how **V1** would be used effectively, so there was no way of knowing what optimizations would actually be required.

- For both **efficiency** and **security** reasons, V1 only has and will ever have the **minimal required** features.

- If something can be solved **externally**, without adding extra code, it will be solved that way exclusively.

This is why **V1** only works and will ever work with **strictly standard** ERC20 tokens.

Tokens that don't fit the requirement (e.g. fee-on-transfer or rebase tokens) have to be wrapped to work.

This is also why **V1** does not and will not provide orders as NFTs directly. It can be done by using a wrapper.

- **Observer** (view) functions provide only the **internal** state, and do **no validity checks**, neither on input nor output data.

- Consumers of the smart contract are expected to do both the high level interpretation of the data and the validity checking.

- This way the consumer can be more efficient by avoiding redundant validity checks and storage access.

For other smart contracts we provide a library with all the high level functions that interpret the internal data of *The Orderbook*.

What is planned for **V1**?

Feedback Phase

- We will monitor how **V1** is being used, with a focus on the **UX** quality.
- We will collect **feedback** from its users.
- From all of this we will try to determine where **optimizations** are actually required, or if **extra functionality** is needed.

Revision and Optimization Phase

- We will implement any optimization or functionality alteration deemed necessary by the **Feedback Phase**.
- We will at this stage introduce changes that will reduce the code's **readability** in favor of **performance**.

Integration Phase

- After **V2** is released we will work on mechanisms for making **V1** and **V2** work together.
- This integration might be **external**, a **common gateway** for interacting with a **V1** and **V2** orderbook as if they were one.
- Alternatively, an integration might be a modification of **V1** which allows for a deeper connection with a **V2** orderbook.

The Orderbook DEX V2

The Orderbook DEX V2 is **under development**, and more details will be shared when it is ready for **testing**.

Features

- It allows to place orders with an **expiration time**.
- It has **Similar features** to [The Orderbook DEX V1](#)
- The **utility token** will be required to operate.

Most interactions with the orderbook will **require** users to pay a **fee** in the utility token.

Some will **occasionally** give back the collected fees as a **reward** for having to do extra work.

- ...more to be disclosed when **V2** is **ready for testing**.

The Orderbook DEX V3

The Orderbook DEX V3 is in the **early stages of design**, so details aren't available and features might change.

Features

- **More types** of orders.
- The **utility token** will have a more **important role**, and will be used not only for **rewards**, but also for **staking**.
- ...more details to come as **system design** progresses.

Client App

Features

- Initially developed as a **web app**, it will be later on provided as a **desktop app**.
- It only uses **on-chain** data.

This means that it does not rely on any **centralized** data provider. While this does not compromise the **decentralized** nature of the system, it also means that it takes more time to boot up the market data.

- It's **distributed** both through decentralized and centralized means.

The app will be hosted on a **traditional web domain**, but the recommended way to access the app is through **decentralized distribution**, which will be eventually the sole way to access the app.

- For **UX reasons**, the app interacts with orderbooks through an **Operator** smart contract owned by the user.

Check [this page](#) for details.

More info

- [The Operator smart contract](#)

The *Operator* smart contract

Why is it used?

- The **Orderbook** smart contracts are not designed with **end user experience** in mind, as they are meant to also be used by **other smart contracts**.
- Particularly a transaction to place a limit order **close to market** might fail because the market moved.
- There is also the issue of **handling token approvals**.

We **rather not** make the user give **full access** of their funds to the orderbook smart contract, as this is considered a **poor practice** regarding security.

- Finally it helps give **proper feedback** of the result of an operation to the user.

EVM based chains are particularly **unreliable** when trying to obtain the **reason** why a transaction **failed**.

The **Operator** helps exposing this info to the user.

What does it do?

- The *Operator* acts as a **wallet** for the funds to be traded in *The Orderbook DEX* ecosystem.

Keep in mind that the *Operator* is a smart contract owned **solely** by the user.

- The *Operator* handles the required **token approvals** for each operation.

So there is no need to give any **permissions** to spend tokens.

- An operation to place a **limit order** will be filled through a **market order** if the market **moved**.

Further considerations

- Keep in mind that the **Operator** smart contract is an **upgradable proxy**, and a malicious implementation will be able to do **anything** with the funds held by the *Operator*.

Therefore, **do not upgrade** to a newly released version of the *Operator* until you have **checked** that it's **safe to use**.

The Utility Token

What is its purpose?

- To **finance** *The Orderbook DEX* project.

The Orderbook DEX Team will be able to finance the project through **vested token reserves**.

- To **compensate** the users who end up having to do **extra work** to keep the DEX working.

When the DEX requires some of its users to **ocasionally** do extra work, they'll be **rewarded** tokens collected from other users as **fees**.

- To allow users to **participate** in the running of a DEX through **staking**.

When the DEX requires its users to run **maintenance** functions to keep it running, the users will have to **stake tokens** to be able to do so and get **fees** as a reward.

Staked tokens would also act as **collateral** to keep the participants honest.

What is it not?

- It is **not** a **governance** token.

Though we are open to it **in the future**.

Regarding Chains & Bridges

- We are **not** going to use **bridges**.

We don't want the DEX to be **dependent** on a bridge. Not until we are sure this won't **compromise** the DEX's **decentralization**. We are open to revise this in the future.

Meanwhile, we would rather explore **alternative solutions** to cross-chain interactions. Check [The Roadmap](#) to learn more about it.

- Total supply of tokens will **not increase**.

New tokens might be created for the purpose of **upgrading** or **deploying to new chains**, but the total supply will remain the same when doing so.

- When deploying *The Orderbook DEX* on a **new chain**, token holders will be given a chance to **transfer tokens** to the new chain by **burning them** in another.

Keep in mind this is a **one-way** transfer.

- *The Orderbook DEX Team* will also have a chance to transfer **vested tokens** to the new chain while keeping the same **vesting rules**.

This is to make sure that a chain might not end up **undersupplied** with tokens. Vested tokens will **remain locked** the time they are expected to.

More about the token

- [Tokenomics](#)

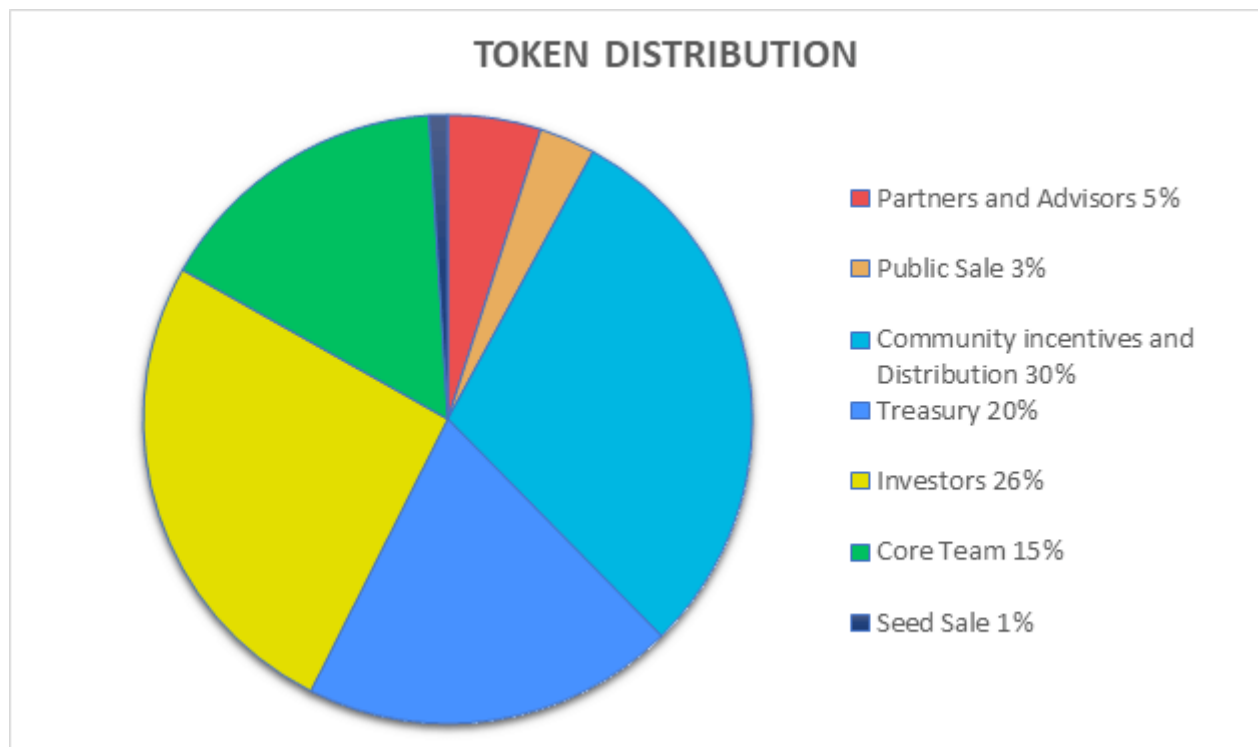
Tokenomics

Token Supply

- The token total supply is 1,500,000,000

Token Distribution

- Partners and Advisors: 5% (75,000,000)
- Public Sale: 3% (45,000,000)
- Community incentives and Distribution: 30% (450,000,000)
- Treasury: 20% (300,000,000)
- Investors: 26% (390,000,000)
- Seed Sale: 1% (15,000,000)
- Core Team: 15% (225,000,000)



Token Allocation

Stage	Vesting Period
Investors	20% to be released on TGE. After a two month cliff, 80% will be released over 6 months on a monthly basis at a monthly linear rate.
Public Sale	100% to be released on TGE.
Treasury	After a 6 month Cliff, 100% will be released over 42 months on a monthly basis at a monthly linear rate.
Community incentives and Distribution	After a 6 month Cliff, 100% will be released over 42 months on a monthly basis at a monthly linear rate.
Partners and Advisors	After a 3 month cliff, 100% will be released over 36 months on a monthly basis at a monthly linear rate.
Core Team	After a 12 months Cliff, 20% will be released, while the remaining 80% will be released over 42 months on a quarterly basis at a quarterly linear rate.
Seed Sale	20% to be released on TGE. After a two month Cliff, 80% will be released over 6 months on a monthly basis at a monthly linear rate.

Roadmap

Keep in mind that dates **might change** depending on **funding** available and **feasibility**. Join [our socials](#) to stay in the loop.

Q3 2022

- Deploy **V1** on **testnet**
- Release **temporary** Web App

Q4 2022

- Deploy **V1** on **mainnet**
- Redesign and release **alpha version** of Web App

We had to do this to make sure the Web App's code is **open source compatible** and can be disclosed and distributed without problems.

Q1 2023

- Deploy **V2** on **testnet**
- Improve **analysis tools** of Web App
- Revise and optimize **V1**

Q2 2023

- Deploy **V2** on **mainnet**
- **First** iteration of **automation** tools of Web App

Q3 2023

- Revise and optimize **V2**
- V1 and V2 **integration**

Q4 2023

- Deploy **V3** on **testnet**
- **Second** iteration of **automation** tools of Web App

Q1 2024

- Deploy **V3** on **mainnet**
- **Alpha** version of **Desktop App**

TBD

Please consider most of the following to be **tentative** and kind of a "*wishlist*" of problems we want to tackle, but we are not sure at this stage if they are entirely feasible.

- Deploy on **more** EVM compatible chains
- Expand **beyond EVM** chains
- **Conditional** orders
- **Short** Trading
- **Options** Trading
- **Cross-chain** DEX / bridge alternative solution
- DEX with **no fund locking** requirement
- **NFT** Orderbook

Team

- **DM**

▮ Lifelong alternative assets investor. Market analyzer and trader.

- **JIDC**

▮ Bachelor's Degree in Human Resources, Investment, Financial Advisor and trader.

- **MA** (aka *Frugal Wizard*)

▮ All-round software developer, obsessed with making code beautiful.

- **PC**

▮ Bachelor's Degree in Economics, Investment and Financial Advisor.

- **RC**

▮ 35 years economist, trader, businessman and ice cream enjoyer.

- **RI**

▮ Builder and entrepreneur by day. NFT degen and cardboard collector by night.

- **TA**

▮ Bachelor's Degree in Economics. Market analyzer and trader.

Our Socials

- [Twitter](#)
- [Discord](#)
- [Reddit](#)
- [Medium](#)
- [GitHub](#)

