Limit Order Book and Matching

This document outlines the key differences between centralized exchanges and dYdX Chain, focusing on the decentralized limit order book and matching engine.

Blockchain Overview

dYdX Chain is a p2p blockchain network usingCosmosSDK(opens in a new tab) andCometBFT(opens in a new tab) (formerly Tendermint).

Anyone can run a full node using the open-source software. Full nodes with sufficient delegated governance tokens participate in block building as validators.

The software repository is:https://github.com/dydxprotocol/v4-chain/(opens in a new tab)

Limit Order Book

Each full node in the network maintains an in-memory order book, which undergoes state changes in real time as traders submit order instructions.

Block proposers use trades from their local order book to build blocks, with matches generated by price-time priority. Since message arrival order varies between nodes, the order book may differ across the network at any given point in time. To address this, upon seeing a new consensus-commited block, nodes sync their local books with the block contents.

Clients can subscribe to a node's book state using theFull Node Streaming API .

Matching and Block Processing

The order matching logic is broadly similar to centralized exchanges, with some key differences:

1. On receiving a cancel instruction:
2.
    - The node cancels the order unless it's already matched locally.
3.
    - The cancel instruction is stored until it expires (based on theGTB field(opens in a new tab)
4.
    -).
5. On receiving an order:
6.
    - The order fails to place if it has already been cancelled.
7.
    - Otherwise, it is matched and/or placed on the order book, with optimistic matches stored locally.
8. Validator nodes propose blocks that include all their local matches.
9. When processing a new block:
10.
    - The node starts from the state of the prior block (local state used to propose is temporarily disregarded).
11.
    - The block’s changes are applied.
12.
    - The node replays its local state on top of the new state, during which:* Cancels are preserved.
13.
    -
        - Orders matched in the prior local state are re-placed.
14.
    -
        - Orders may match differently, fail to place due to cancellation, or not match at all in the new state.

Source Code References

For further details on how the protocol handles these actions, refer to the following source code references:

- Seehere(opens in a new tab)
- andhere(opens in a new tab)
- for how the protocol reacts when (1) a cancel is seen.
- When (2)an order is placed(opens in a new tab)
- andchecked to not be already cancelled(opens in a new tab)

- .
- When (3)proposing a block(opens in a new tab)
- .
- And (4) whennodes process blocks committed by consensus(opens in a new tab)
- .

Order Messages

Order instructions are limit order placements, replacements, and cancellations.

Note: This section covers short-term orders which live off-chain, in node memory, until matched.

Stateful orders (on-chain, consensus-speed placement) exist for longer-lived limit orders but aren't recommended for API traders. More info:Short-term vs Stateful Orders

Finality and GTB

Each limit order placement or cancellationincludes a GTB (good-til-block) field(opens in a new tab), which specifies the block height after which the instruction expires.

While rare, it is possible for a cancel instruction to be seen by the current block proposer but not by one or more subsequent proposers (if the instruction isn't gossiped to them in time through the p2p network). In such cases, the order could still match after the sender expects it to have been cancelled.

Therefore, we recommend that API traders consider setting tight GTB values on order placements (e.g. the current chain height + 3) because expiry due to GTB is the only guaranteed way for an order to become unfillable. Consensus does not permit any order to fill at a height greater than its GTB.

Replacements

We recommend using replacement instructions over cancelling and placing new orders.

Replacing prevents accidental double-fills that can occur with a ‘place order A, cancel order A, place order B’ approach, where both A and B might fill simultaneously unless the chain height has already passed A’s GTB.

For example, after the following messages are sent:

1. Place A: Sell 1 @ 100, client id = 123
2. Cancel A
3. Place B: Sell 1 @ 101, client id = 456

If a proposer sees messages 1 and 3, but not 2, it sees both orders A and B as open. If it also sees marketable bids for qty >= 2, both could fill simultaneously.

Replacement Instruction Fields

To replace an order, send a placement with the same order IDand a larger GTB value .

Orders have the same ID if these client-specified fields match (OrderId proto definition(opens in a new tab)):

- Subaccount ID(opens in a new tab)
- (owner: signing address, number: 0 unless different subaccount)
- Client ID
- Order flags (0 for short-term orders)
- CLOB pair ID