Onboarding FAQs

Intro

1. Can you give an overview of the structure of the network and the role full nodes/validators play in constructing the orderbook and participating in block building, and how trades are placed?
2.
    - dYdX Chain (or "v4") is composed of full nodes and each maintains an in-memory order book. Anyone can use the open source software to run a full node. Traders can submit order placements and cancellations to full nodes, which gossip the transactions amongst themselves.
3.
    - Full nodes with enough delegated layer 1 governance tokens participate in block building as validators. Validators on dYdX Chain take turns proposing blocks of trades every ~1 second. The validator whose turn it is to propose a block at a given height is called the proposer. The proposer uses its mempool orderbook to propose a block of matches, which validators either accept or reject according to CometBFT (Tendermint) consensus.
4.
    - All full nodes have visibility into the consensus process and the transactions in the mempool. Another component of dYdX Chain is the indexer software, an application that reads data from full nodes and exposes it via REST / WebSocket APIs for convenience.
5. What is the difference between a full node and a validator?
6.
    - A full node does not participate in consensus. It receives data from other full nodes and validators in the network via the gossip protocol. A validator participates in consensus by broadcasting votes signed by each validator's private keys.
7. What are the benefits of running a full node as a market maker?
8.
    - Running a full node will eliminate the latency between placing an order and when the actual order is gossipped throughout the network. Without your own node, your order will need to first be relayed to the nearest geographic node, which will then propagate it throughout the network for you. With your own node, your order will directly be gossiped.
9.
    - Instructions on how to set up a full node can be foundhere(opens in a new tab)
10.
    - .
11. What is the current block time?
12.
    - The current block time is ~1 second on average
13. What is an indexer?
14.
    - The indexer is a read-only service that consumes real time data from dYdX Chain to a database for visibility to users. The indexer consumes data from dYdX Chain via a connection to a full node. The full node contains a copy of the blockchain and an in-memory order book. When the full node updates its copy of the blockchain and in-memory order book due to processing transactions, it will also stream these updates to the indexer. The indexer keeps the data in its database synced with the full-node using these updates. This data is made available to users querying through HTTPS REST APIs and streaming via websockets. More info can be foundhere(opens in a new tab)
15.
    - .

Trading on an Exchange Run on dYdX Chain

1. What are the different order types in dYdX Chain?
2.
    - There are two order types: Short-Term orders and stateful orders.* Short-Term orders are meant for programmatic, low-latency traders that want to place orders with shorter expirations.
3.
    -
        - Stateful orders are meant for retail that wants to place orders with longer expirations. These orders exist on chain.
4. How does the orderbook work in dYdX Chain for short-term orders?
5.
    - Each validator runs their own in-memory orderbook (also known as mempool), and the set of orders each validator knows about is what order placement transactions are in their mempool.
6.
    - User places a trade on a decentralized front end (e.g., website) or via the typescript or python client that places orders directly to a full node or validatorAPI

7.
    - The consensus process picks one validator to be the block proposer. The selected validator will propose their view of the matches in the next proposed block.
8.
    - If the matches are valid (orders cross, subaccounts well-collateralized, etc.) and accepted by ⅔+ of validator stake weight (consensus), then the block is committed and those matches are written to state as valid matches.
9.
    - After the block is committed, the updated on-chain (and off-chain) data is streamed from full nodes to Indexers. The Indexer then makes this data available via API and websockets back to the front end and/or any other outside services querying for this data.
10.
    - Note: the block proposer's matches are the canonical matches for the next block assuming their block is accepted by consensus.* Other validators maintain a list of matches and those matches might differ from the block proposer's matches, but if they're not the block proposer those matches will not be proposed in the next block.
11.
    -
        - Similarly, the indexer is not the block proposer so its list of matches might be different from the block proposer's matches, until the network reaches finality.
12. Why should market makers only use short-term orders?
13.
    - Short-Term orders are placed and can be immediately matched after they're added to the mempool, while stateful orders can only be placed and matched after they're added to a block.* Short-Term orders should always have superior time priority to stateful orders.
14.
    -
        - Stateful orders have worse time priority since they can only be matched after they are included on the block, short-term orders can be matched in the same block
15.
    - Short-Term orders have less restrictive rate limits than stateful order rate limits. See rate limits later on in this section
16.
    - Short-Term orders can be replaced, and stateful orders currently don't support replacement.
17.
    - Short-Term orders can be canceled immediately after they're placed, while stateful orders can only be canceled after they've been included in a block.
18.
    - Short-Term orders can be received by validators in any order, while stateful orders have an ordering requirement and will fail to be placed if they're received out of order.* This is because stateful orders use a "sequence number", which is equivalent to a nonce on Ethereum. Short-Term orders don't use this.
19. How can I understand how finality works on dYdX Chain?
20.
    - When your order fills, a block proposer will propose a block containing the fill (visible to the whole network), and then the block will go through consensus. If the block is valid it will be finalized a couple seconds later (in Cosmos-speak this happens at the "commit" stage of consensus after all validators have voted). At that point, an indexer service will communicate the fill to you.
21.
    - It is recommended to post orders with a "Good-Til-Block" of the current block height, and adjusting prices once per block. If the block is published without a match to your order, you know that it is no longer active and did not fill.
22. How can I place a short-term order?
23.
    - Please use the latest dYdX Chaintypescript client(opens in a new tab)
24.
    - to place orders
25.
    - Please refer to theorder.proto(opens in a new tab)
26.
    - for parameter and field definitions
27.
    - For more advanced order placements, please refer to the validator clientv4-proto-js(opens in a new tab)
28.
    - orv4-proto-py(opens in a new tab)
29.
    - .
30. How can I tell if the block proposer has placed my short-term order?
31.
    - The block proposer has proposed and filled the order in the block.

32.
    - The block proposer has the order in their mempool.
33. How can I tell if my short-term order is canceled?
34.
    - Short-term order placements and cancellations are best-effort, and therefore their cancels and placement can't be considered actually canceled and unfillable until expiry
35.
    - A FOK or IOC order can also be seen as canceled if a fill does not occur after expiry
36.
    - The indexer does not send a websocket notification if a short-term order has been canceled
37.
    - However, since the goodTilBlock for the cancel needs to be set when placing the cancel, the block height can be checked to ensure that the cancel was successful
38.
    - This is lower latency than what a websocket notification could provide
39. How can I replace an order?
40.
    - Replacing an order reuses the short-term order placement function with the same order ID(opens in a new tab)
41.
    - and an equal-to-or-greater good til block
42.
    - Note: when replacing partially-filled orders, the previous fill amount is counted towards your current order.* Example: Buy 1 BTC order @ 20k is filled for 0.5 BTC. After replacing that order with a Buy 2 BTC order @ 25k, that order can only be filled for a maximum of 1.5 BTC. This is because the previously replaced order was already filled for 0.5 BTC.
43. Are fills computed/updates steamed only when a block is finalized? How about order placements?
44.
    - Fills are computed only when a block is finalized
45.
    - Short term order place / cancel (including IOC / FOK orders being canceled due to not filling / being on the book or POST-ONLY orders crossing) are streamed when the full node the Indexer deployment is listening to receives the order / cancel and not only when the block is finalized* This is why the status “BEST_EFFORT_OPENED” or “BEST_EFFORT_CANCELED” since the Indexer only knows that a full-node received the order / cancel, and it’s not guaranteed to be true across the whole network
46.
    - For the orderbook updates, these are sent when the full-node the Indexer is listening to receives orders / cancels and not just when the block is finalized* For example, when the full-node receives a short term order it will be approximate how much is filled and how much would go on the orderbook. This is what the Indexer uses to stream orderbook updates. However, there is no guarantee that the orderbook looks the same in other nodes in the network
47.
    - Note that you can now stream the orderbook directly through your full node for the orderbook. Read more about thathere(opens in a new tab)
48.
    - .
49. Do orders get matched and removed from the book in between blocks?
50.
    - For removal of short term orders, yes they can be removed in between blocks, however this is on a node-by-node basis and not across the whole network* E.g. a short-term order could be removed on one node, but still be present on another
51.
    -
        - When a short-term order expires (current block height > goodtilBlock), then it is guaranteed to be removed from all nodes
52.
    - For removal of stateful orders, they can be removed from the book in-between blocks. This is on a node-by-node basis* If the node removing the stateful orders is the block proposer, these stateful order removals will also be propagated to all other nodes, an be entirely removed from the network
53.
    - For all orders, regarding matching* For matching, each node on the network will attempt to match the orders as they are received in-between blocks
54.
    -
        - Per block, only 1 node (the block proposer) will propagate the matches it’s done during the block to all other nodes in the network. Validator nodes take turns being the block proposer based on their stake weight
55.
    -

        - If a set of validators with ⅔+ of the stake weight of the network see the matches propagated as valid, then those matches are included in the block when finalized
56.
    -
        - The only matches that occur on the network are the ones in the block
57. Do certain order types have priority? Are cancels prioritized?
58.
    - Short term orders when received by a node will be matched against it’s in-memory orderbook* Cancels of short-term orders are also processed by a node when received
59.
    - Stateful orders (long-term / conditional) are matched at the end of the block when they are received* E.g. Stateful orders have at least a 1 block-time delay (it’s possible the order does not get included in the block) between a node receiving the order, and it being matched
60.
    -
        - Stateful order placement will be processed AFTER short-term order placements and cancellations for a block
61.
    - Stateful order cancellations are also done at the end of the block they are received* The stateful order cancellations are also processed AFTER short-term placements and cancellations for a block
62.
    - As mentioned above, only the matches from the block proposer will be included in the block (if a set of validators with ⅔+ of the stake weight of the network see the matches as valid)
63. How does the order cancellation mechanism work? Is it necessary for the cancel instruction and the original order to be on the same validator for a successful cancel? What's the best practice to ensure order cancellation succeeds?
64.
    - Short-term:* When validators receive a cancellation, if they don't already see a match for the order, they will remove the order from their order book
65.
    -
        - Only once every validator receives the cancellation is when the order will no longer be able to be matched
66.
    -
        - The other way an order would no longer be matchable is if the block height is past the good til block
67.
    - Long-term:* Once a stateful order cancellation is included into a block, the order will be canceled and no longer matchable. This could take 1s+ for a cancelation to be included in a block
68. Why is it slower to cancel orders than place orders?
69.
    - An order placement only needs to be on a single validator to have a match happen, but the cancellation has to have arrived at the block proposer, but since the BP rotates, to be completely sure that the order wont be matched, it has to arrive at all the validators who will be block proposer before the order expires. This is why cancelations seem to be guaranteed slower than placing / matching orders.
70. How do order statuses transition for the Indexer, for short-term and long-term orders?
71.
    - Short-term* Once the order is placed and seen by the Indexer's full-node, the order has status BEST_EFFORT_OPENED
72.
    -
        - If the order is matched and has a fill in a block, the order has status OPEN
73.
    -
        - If the order is fully-filled, the order has status FILLED
74.
    -
        - If the order is canceled by a cancel order message, the order will be status BEST_EFFORT_CANCELED , the order may still have fills if other validator nodes haven't received the cancel message yet
75.
    -
        - If the order expires due to the block height exceeding the good til block of the order, the order status will be CANCELED, the order no longer can be filled
76.
    - Long-term* Once the order is placed and included in a block, the order has status OPEN
77.
    -
        - If the order is fully filled, the order has status FILLED
78.
    -

    - If the cancelation of the order is included in a block, the order has status CANCELED the order can no longer be filled
79. How do subaccounts work on dYdX Chain?
80.
    - Each address's subaccounts all fall under a single address, but they are labeled subaccount0, subaccount1, etc. This is unlike v3, where each subaccount was a secondary address.
81.
    - To begin trading, you need to make sure your funds are in your subaccount. You can do this two ways:* Frontend: Simply leave your frontend open and it will automatically sweep.
82.
    -
        - Backend: Simply transfer USDC to it like irthis example(opens in a new tab)
83.
    -
        - .
84. When I transfer funds to create a new subaccount (the only way to create new subaccount btw), do I need gas?
85.
    - Yes, you will need gas. Fortunately, both USDC and cosmos native DYDX can be used to pay for gas fees. This USDC must be in the main wallet and not another subaccount to pay for fees.
86.
    - To ensure this, the frontend leaves a small amount of USDC in your wallet when sweeping into your subaccount, to ensure there's enough to pay for gas.
87. What impact do subaccounts have on rate limits?
88.
    - Rate limits are per account, and not per subaccount.
89. How do we compete for liquidation orders?
90.
    - If you run a full-node, there is a liquidations daemon that has metrics on what accounts are up for liquidation orders, and they could try and compete for liquidations that way.
91.
    - However, this is not at all documented so you'll have to work it out by reading code.

Trading as it Relates to Validators

1. How much throughput and latency can be expected from a self-hosted full node? Would having multiple full nodes in different regions improve speed?
2.
    - Throughput of up to 1500 orders / second from our load-testing. Latency depends on which validator is the proposer. Having multiple full-nodes in different regions where there are validators (so maybe 1 in Europe + 1 in Tokyo) would lead to improved latency.
3. 0 Do validators communicate through a public P2P network, or is there an internal network?
4.
    - It's a public P2P network.
5. What is the expected order-to-trade latency under normal conditions?
6.
    - Expected order → trade latency would be:* Time for order to get from the node it was submitted to, to the proposer, so location dependent.
7.
    -
        - Order match -> trade, probably at least 1 block so ~1.1s, could be more than 1 block.
8. Compared to a full node, is it faster to submit transactions directly to the validator, or is it faster to broadcast to the next validator with block-producing rights?
9.
    - It would be faster to submit a transaction directly to the block proposer, the difference between a full-node / validator is negligible unless that validator was the proposer.
10. Do you have some faster validators that we can send orders to?
11.
    - Validators usually don't expose the RPC endpoints for orders to be submitted to, and we also don't have a list of validator IPs.
12. How do other teams improve their speed?
13.
    - Some teams are trying to get data about the order book / order updates from a full-node they are running to improve the latency to receiving data, as there is additional latency to getting order updates due to the Indexer systems having additional latency. We currently do not have documentation around this, but are working on it.

Indexer

1. How does the indexer reconstruct the orderbook when it started/initial snapshot of the book?
2.
    - A full node is run alongside the Indexer and sends messages to the indexer when it receives orders either from the RPC or gossiped from other nodes, as well as any updates from:* node pruning the order when it expires
3.
    -
        - another order that matches an order that the node received earlier
4.
    -
        - node removing order due to receiving a cancel from RPC or gossip
5.
    - The indexer also updates the order book whenever it receives these order messages.
6. How does the indexer know what orders are in the book on start up?
7.
    - On a cold-start, a full-node would still have all the stateful orders and would send them to the indexer. For short-term orders, the full-node would not know them, nor would the indexer. Since short-term orders only are valid for 20 blocks, within 20 blocks the indexer would have an accurate view of the order book, but for the first 20 blocks it would not.

MEV

1. How will dYdX Chain handle MEV?
2.
    - Unlike general purpose smart contract environments, the Cosmos infrastructure enables unique MEV solutions to be built that align a validator's incentives with a user's incentives. dYdX Chain has a framework where MEV is measured via adashboard(opens in a new tab)
3.
    - created by Skip Protocol. The first step would be to punish validators with slashing. Further proposed solutions are still being considered, and will be announced once finalized.
4. When do I have finality related to fills?
5.
    - When your order fills, a block proposer proposes a block containing the fill (visible to the whole network), and then the block undergoes consensus. If the block is valid, it finalizes shortly thereafter (in Cosmos-speak this happens at the "commit" stage of consensus, after all validators have voted). In Cosmos, every block is final (no reorgs or forks).
6.
    - If you're connected via full node, you'll see each step of this process. If you're connected via the indexer service, you'll see order updates over webSocket as soon as each block is confirmed.
7. Would deliberately taking already-canceled orders be considered an attack against makers? How can such attacks be mitigated?
8.
    - Nodes should respect cancels as soon as they receive them, if they don't then we see that as MEV and the aforementioned dashboard / metrics tracking MEV will track that.

Pricing

1. How is the oracle price is computed? *The oracle price has four parts:* pricefeed daemon: side car that pulls price data from external sources and caches them for the node to uselink(opens in a new tab)
2.
    -
        - consensus: proposes new priceslink(opens in a new tab)
3.
    -
        - and validates the new prices are acceptablelink(opens in a new tab)
4.
    -
        - module: updates the state based on the new price, also has logic for validation and etclink(opens in a new tab)
5.
    -
        - params: determines the external sources and sensitivitylink(opens in a new tab)
6.
    -
        - , these are configured per network (testnet genesis example), but should query the network config for these dydxprotocold query prices list-market-param

Rewards

1. How will trading rewards work on dYdX Chain?
2.
    - Trading rewards are not controlled by dYdX. dYdX recommends that trading rewards could be calculated primarily based on total taker fees paid, along with a few other variables. The full proposed formula can be foundhere(opens in a new tab)
3.
    - . These rewards could be distributed on a block by block basis (1-2 seconds).
4. Will liquidity provider rewards exist in v4?
5.
    - Liquidity provider rewards in v4 are not controlled by dYdX. dYdX recommends that liquidity provider rewards should cease to exist in v4. Makers could be rewarded with a maker rebate ranging from 0.5bps to 1.1bps, based on their nominal volume and volume share. The full proposed fee schedule can be foundhere(opens in a new tab)
6.
    - .