Indexer Deep Dive

A good way to think about the indexer is as similar to Infura or Alchemy's role in the Ethereum ecosystem. However, unlike Infura/Alchemy, and like everything else in dYdX Chain, the Indexer is completely open source and can be run by anyone!

What is the Indexer?

As part of tooling for the dYdX ecosystem, we want to ensure that clients have access to performant data queries when using exchanges running on dYdX Chain software. Cosmos SDK Full Nodes offer a number of APIs that can be used to request on-chain data. However, these Full Nodes are optimized for committing and executing blocks, not for serving high frequency, low-latency requests from web/mobile clients.

This is why we wrote software for an indexing service. The Indexer is a read-only service that serves off chain data to clients over REST APIs and Websockets. Its purpose is to store and serve data that exists on the dYdX Chain in an easier to use way. In other words, the purpose of an indexer is to index and serve data to clients in a more performant, efficient and web2-friendly way. For example the indexer will serve websockets that provide updates on the state of the orderbook and fills. These clients will include front-end applications (mobile and web), market makers, institutions, and any other parties looking to query dYdX Chain data via a traditional web2 API.

On-chain vs. Off-chain data

The Indexer will run two separate ingestion/storage processes with data from a v4 Full Node: one for on-chain data and one for off-chain data. Currently, throughput of on-chain data state changes is expected to be from 10-50 events/second. On the other hand, the expected throughput of off-chain data state changes is between 500-1,000 events/second. This represents a 10-100x difference in throughput requirements. By handling these data types separately, v4 is built to allow for different services to better scale according to throughput requirements.

On-chain Data

On-chain data is all data that can be reproduced by reading committed transactions on the dYdX Chain. All on-chain data has been validated through consensus. This data includes:

1. Account balances (USDC)
2. Account positions (open interest)
3. Order Fills1. Trades
4.
    1. Liquidations
5.
    1. Deleveraging
6.
    1. Partially and completely filled orders
7. Funding rate payments
8. Trade fees
9. Historical oracle prices (spot prices used to compute funding and process liquidations)
10. Long-term order placement and cancellation
11. Conditional order placement and cancellation

Off-chain Data

Off-chain data is data that is kept in-memory on each v4 node. It is not written to the blockchain or stored in the application state. This data cannot be queried via the gRPC API on v4 nodes, nor can it be derived from data stored in blocks. It is effectively ephemeral data on the v4 node that gets lost on restarts/purging of data from in-memory data stores. This includes:

1. Short-term order placement and cancellations
2. Order book of each perpetual exchange pair
3. Indexed order updates before they hit the chain

Indexer Architecture

The Indexer is made up of a series of services that ingest information from v4 Full Nodes and serve that information to various clients. Kafka topics are used to pass events/data around to the services within the Indexer. The key services that make up Indexer are outlined below.

Ender (On-chain ingestion)

Ender is the Indexer's on-chain data ingestion service. It consumes data from the "to-ender" Kafka topic (which queues all

on-chain events by block) and each payload will include all event data for an entire block. Ender takes all state changes from that block and applies them to a Postgres database for the Indexer storing all on-chain data. Ender will also create and send websocket events via a “to-websocket-?” Kafka topic for any websocket events that need to be emitted.

Vulcan (Off-chain ingestion)

Vulcan is the Indexer’s off-chain data ingestion service. It will consume data from the “to-vulcan” Kafka topic (queues all off-chain events), which will carry payloads that include active order book updates, place order updates, cancel order updates, and optimistic fills. This data will be stored in a Redis cache. Vulcan will update Redis with any new open orders, set the status of canceled orders to cancel pending, and update orderbooks based on the payload received. Vulcan will also update Postgres whenever a partially filled order is canceled to update the state of the order in Postgres. Vulcan will also create and send websocket events via a “to-websocket-?” Kafka topic for any websocket events that need to be emitted.

Comlink (API Server)

Comlink is an API server that will expose REST API endpoints to read both on-chain and off-chain data. For example, a user could request their USDC balance or the size of a particular position through Comlink, and would receive a formatted JSON response.

As an explicit goal set out by the dYdX team, we’re designing v4 APIs to closely match the v3 APIs(opens in a new tab) . We have had time to gather feedback and iterate on these APIs over time with v3, and have confidence that they are reasonable at the product-level.

Roundtable

Roundtable is a periodic job service that provides required exchange aggregation computations. Examples of these computations include: 24h volume per market, open interest, PnL by account, candles, etc.

Socks (Websocket service)

Socks is the Indexer’s websockets service that allows for real-time communication between clients and the Indexer. It will consume data from ender, vulcan, and roundtable, and send websocket messages to connected clients.

Hosting & Deploying the Indexer

In service of creating an end-to-end decentralized product, the Indexer will be open source. This will include comprehensive documentation about all services and systems, as well as infrastructure-as-code for running the Indexer on popular cloud providers.

The specific responsibilities of a third party operator looking to host the Indexer generally include initial deployment and ongoing maintenance.

Initial deployment will involve:

- Setting up AWS infrastructure to utilize the open-source repo.
- Deploying Indexer code to ingest data from a full-node and expost that information through APIs and websockets
- Datadog (provides useful metrics and monitoring for Indexer services), and Bugsnag (real-time alerting on bugs or issues requiring human intervention).

Maintenance of the Indexer will involve:

- Migrating and/or upgrading the Indexer for new open-source releases
- Monitoring Bugsnag and Datadog for any issues and alerting internal team to address
- Debugging and fixing any issues with a run book provided by dYdX

dYdX believes that, at minimum, a DevOps engineer will be required to perform the necessary duties for deployment and maintenance of the Indexer. An operator will need to utilize the services below:

- AWS* ECS - Fargate
-
    - RDS - Postgres Database
-
    - EC2
-
    - Lambda
-
    - ElastiCache Redis
-
    - EC2 ELB - Loadbalancer

-
    - Cloudwatch - Logs
-
    - Secret Manager
- Terraform Cloud - for deploying to the cloud
- Bugsnag - bug awareness
- Datadog - metrics and monitoring
- Pagerduty - alerting

Operators should be able to host the open-sourced Indexer for public access in a highly available (i.e., high uptime) manner. Requirements include owning accounts to the services above and hiring the appropriate personnel to perform deployment and maintenance responsibilities.

Last updated onApril 3, 2024 Architectural Overview Getting Started