CoW SDK

Docs website

Test coverage

Statements Branches Functions Lines

Getting started

Usage examples:VanillaJS ,Create React App ,NodeJS

Installation

yarn add @cowprotocol/cow-sdk

Content

- OrderBookApi
-
    - provides the ability to retrieve orders and trades from the CoW Protocol order-book, as well as add and cancel them
- OrderSigningUtils
-
    - serves to sign orders and cancel them usingEIP-712
- SubgraphApi
-
    - provides statistics data about CoW protocol fromSubgraph
- , such as trading volume, trade count and others

import

{ OrderBookApi , OrderSigningUtils , SubgraphApi }

from

'@cowprotocol/cow-sdk'

const chainId =

100

// Gnosis chain

const orderBookApi =

new

OrderBookApi ({ chainId }) const subgraphApi =

new

SubgraphApi ({ chainId }) const orderSigningUtils =

new

OrderSigningUtils ( )

Quick start

Sign, fetch, post and cancel order

For clarity, let's look at the use of the API with a practical example: Exchanging0.4 GNO toWETH onGnosis chain network.

We will do the following operations:

1. Get a quote
2. Sign the order
3. Send the order to the order-book
4. Get the data of the created order
5. Get trades of the order
6. Cancel the order (signing + sending)

[You also can check this code in the CRA example]

```
import

{ OrderBookApi , OrderSigningUtils , SupportedChainId }

from

'@cowprotocol/cow-sdk' import

{ Web3Provider }

from

'@ethersproject/providers'

const account =

'YOUR_WALLET_ADDRESS' const chainId =

100

// Gnosis chain const provider =

new

Web3Provider ( window . ethereum ) const signer = provider . getSigner ( )

const quoteRequest =

{ sellToken :

'0x6a023ccd1ff6f2045c3309768ead9e68f978f6e1' ,

// WETH gnosis chain buyToken :

'0x9c58bacc331c9aa871afd802db6379a98e80cedb' ,

// GNO gnosis chain from : account , receiver : account , sellAmountBeforeFee :

( 0.4

*

10

**

18 ) . toString ( ) ,

// 0.4 WETH kind : OrderQuoteSide . kind . SELL , }

const orderBookApi =

new

OrderBookApi ( { chainId : SupportedChainId . GNOSIS_CHAIN

} )

async

function

main ( )

{ const
```

{ quote }

=

await orderBookApi . getQuote (quoteRequest)

const orderSigningResult =

await OrderSigningUtils . signOrder (quote , chainId , signer)

const orderId =

await orderBookApi . sendOrder ({

... quote ,

... orderSigningResult })

const order =

await orderBookApi . getOrder (orderId)

const trades =

await orderBookApi . getTrades ({ orderId })

const orderCancellationSigningResult =

await OrderSigningUtils . signOrderCancellations ([orderId] , chainId , signer)

const cancellationResult =

await orderBookApi . sendSignedOrderCancellations ({ ... orderCancellationSigningResult , orderUids :

[orderId]

})

console . log ('Results: ' ,

{ orderId , order , trades , orderCancellationSigningResult , cancellationResult }) }

OrderBookApi

OrderBookApi - is a main tool for working withCoW Protocol API . Since the API supports different networks and environments, there are some options to configure it.

Environment configuration

chainId - can be one ofSupportedChainId.MAINNET ,SupportedChainId.GNOSIS_CHAIN , orSupportedChainId.SEPOLIA

env - this parameter affects which environment will be used:

- https://api.cow.fi
- forprod
- (default)
- https://barn.api.cow.fi
- forstaging

import

{ OrderBookApi }

from

'@cowprotocol/cow-sdk'

const orderBookApi =

new

OrderBookApi ({ chainId : SupportedChainId . GNOSIS_CHAIN , env :

'staging'

// <----- })

API urls configuration

In case you need to use custom endpoints (e.g. you use a proxy), you can do it this way:

import

{ OrderBookApi }

from

'@cowprotocol/cow-sdk'

const orderBookApi =

new

OrderBookApi ({ chainId : SupportedChainId . GNOSIS_CHAIN , baseUrls :

{

// <----- [SupportedChainId . MAINNET] :

'https://YOUR_ENDPOINT/mainnet' , [SupportedChainId . GNOSIS_CHAIN] :

'https://YOUR_ENDPOINT/gnosis-chain' , [SupportedChainId . SEPOLIA] :

'https://YOUR_ENDPOINT/sepolia' , } }) TheCoW Protocol API has restrictions on the backend side to protect against DDOS and other issues.

The main restriction is request rate limit of:5 requests per second for each IP address Theclient's limiter settings can be configured as well:

import

{ OrderBookApi }

from

'@cowprotocol/cow-sdk' import

{ BackoffOptions }

from

'exponential-backoff' import

{ RateLimiterOpts }

from

'limiter'

const limiterOpts : RateLimiterOpts =

{ tokensPerInterval :

5 , interval :

'second' , }

const backOffOpts : BackoffOptions =

{ numOfAttempts :

5 , maxDelay :

Infinity , jitter :

'none' , }

const orderBookApi =

new

OrderBookApi ({ chainId : SupportedChainId . GNOSIS_CHAIN , limiterOpts , backOffOpts } ,)

Querying the CoW Subgraph

TheSubgraph is constantly indexing the protocol, making all the information more accessible. It provides information about trades, users, tokens and settlements. Additionally, it has some data aggregations which provides insights on the hourly/daily/totals USD volumes, trades, users, etc.

The SDK provides an easy way to access all this information.

You can query the CoW Subgraph either by running some common queries exposed by theCowSubgraphApi or by building your own:

import

{ SubgraphApi , SupportedChainId }

from

'@cowprotocol/cow-sdk'

const cowSubgraphApi =

new

SubgraphApi ({ chainId : SupportedChainId . MAINNET

})

// Get CoW Protocol totals const

{ tokens , orders , traders , settlements , volumeUsd , volumeEth , feesUsd , feesEth }

= await cowSubgraphApi . getTotals ( ) console . log ({ tokens , orders , traders , settlements , volumeUsd , volumeEth , feesUsd , feesEth })

// Get last 24 hours volume in usd const

{ hourlyTotals }

=

await cowSubgraphApi . getLastHoursVolume (24) console . log (hourlyTotals)

// Get last week volume in usd const

{ dailyTotals }

=

await cowSubgraphApi . getLastDaysVolume (7) console . log (dailyTotals)

// Get the last 5 batches const query =

query LastBatches(n: Int!) { settlements(orderBy: firstTradeTimestamp, orderDirection: desc, first: n) { txHash firstTradeTimestamp } } const variables =

{ n :

5

} const response =

await cowSubgraphApi . runQuery (query , variables) console . log (response)

Architecture

One way to make the most out of the SDK is to get familiar with its architecture.

SeeSDK Architecture

Development

Install Dependencies

yarn

Build

yarn build

Build in watch mode

yarn start

Unit testing

yarn test

Code generation

Some parts of the SDK are automatically generated. This is the case for the Order Book API and the Subgraph API

Re-create automatically generated code

yarn codegen Previous Supported tokens Next Exports