Public API Document v1.31

Introduction

This document outlines only the Public API endpoints.

We try to and will continue to maintain these endpoints in terms of uptime, availability and data accuracy. In the event of any problems with these API endpoints please connect with us on support@nestex.one or via our discord at https://discord.gg/CB3SaR7WuM

For more information, please refer to the following section:

Section A: Spot Exchange API
- A.0: Informational
- A.1: Tickers
- A.2: Order Book
- A.3: Trade Book
- A.4: Fees
- A.5: Liquidity
- A.6: Voting

All data is in JSON format. Rate limits for public API are kept in the range of 10 hits per minute, and short-term (30 second) caching is activated to reduce the load on our database.

Note for numeric values: Several users consuming this API via Javascript (nodejs or suchlike) have complained about the values appearing distorted. We will henceforth put all numeric values in double quotes i.e. treat as a string, which prevents the automatic conversion that Javascript does.

Please consume these API mindfully.

A. Spot Exchange API

Endpoints Overview

No.	Endpoint	Description
A.0.	/	Information about the API including endpoints, uptime
A.1.	/tickers	Market related statistics for all markets for the last 24 hours.
A.2.	/orderbook	Order book depth of any given trading pair, split into two different arrays for bid and ask orders.
A.3.	/tradebook	Trade book of any given trading pair, paginated 100 entries per page, maximum of 100 pages (roughly a months worth of data based on our current volumes).
A.4.	/fees	Fees for all assets listed on the exchange denominated in the asset and in USDT/USD equivalent
A.5.	V1/liquidity	Liquidity information about the given asset. Also provides a brief leaderboard.
A.6	V1/voteinfo	List of coins that can be voted on and number of votes received

Endpoint 0 - https://api.nestex.one/ (Information about API)

Output of the form

```
{
  "success": true, "data": "ready",          // status across all balanced servers (ready / degraded)
  "uptime": 43549, "hits": 1965,             // hits served & uptime
  "public_endpoints": {                      // all public endpoints
    "tickers": "/cg/tickers", "orderbook": "/cg/orderbook/NEX", "tradebook": "/cg/tradebook/NEX",
    "fees": "/cg/fees", "liquidity": "/v1/liquidity/NEX", "voting": "/v1/voteinfo", "coinstatus": "/v1/coins"
  }
}
```

Endpoint 1 - /tickers (Market Info)

URL - https://api.nestex.one/cg/tickers to get ALL tickers
OR - https://api.nestex.one/cg/tickers/TICKER_ID to get a single ticker

The /tickers endpoint provides 24-hour pricing and volume information on each market pair available on an exchange.

ALL TICKERS OUTPUT FORMAT /api/cg/tickers	**SINGLE TICKER OUTPUT FORMAT /api/cg/tickers/BTC_USDT**

[{
"ticker_id": "BTC_USDT",
"base_currency": "BTC",
"target_currency": "USDT",
"last_price": “82989.9724137”,
"base_volume": “0.00215083”,
"target_volume": “178.497322”,
"bid": “82052.66134236”,
"ask": “84794.92222999”,
"high": “84886.00914459”,
"low": “80010.86105018”,
"oldltp24h": “70010.86105018”
}, …]
	{
	"ticker_id": "BTC_USDT",
	"base_currency": "BTC",
	"target_currency": "USDT",
	"last_price": “82989.9724137”,
	"base_volume": “0.00215083”,
	"target_volume": “178.497322”,
	"bid": “82052.66134236”,
	"ask": “84794.92222999”,
	"high": “84886.00914459”,
	"low": “80010.86105018”,
	"oldltp24h": “70010.86105018”
	}

(NOTE ALL TICKERS GIVES AN ARRAY, SINGLE TICKER DOES NOT)

/tickers endpoint response description:

Name	**Data Type**	**Description**
ticker_id	string	Identifier of a ticker with delimiter to separate base/target, eg. BTC_ETH
base_currency	string	Symbol/Currency code of a the base cryptoasset, eg. BTC
target_currency	string	Symbol/Currency code of the target cryptoasset, eg. ETH
last_price	decimal	Last transacted price of base currency based on given target currency
base_volume	decimal	24 hour trading volume for the pair (unit in base)
target_volume	decimal	24 hour trading volume for the pair (unit in target)
bid	decimal	Current highest bid price

ask	decimal	Current lowest ask price
high	decimal	Rolling 24-hours highest transaction price
low	decimal	Rolling 24-hours lowest transaction price
oldltp24h	decimal	Rolling 24-hours old price
decimals	integer	Number of decimals to honor for this asset
liquidity	decimal	Liquidity on record in USDT

Endpoint 2 - /orderbook (Order book depth details)

URL - https://api.nestex.one/cg/orderbook

The /orderbook/ticker_id endpoint is to provide order book information with at least depth = 100 (50 each side) returned for a given market pair/ticker.

For example https://api.nestex.one/cg/orderbook/BTC_USDT provides the BTC_USDT pair order book

Endpoint parameters:

Name	Type	Description
ticker_id	string	A ticker such as "BTC_ETH", with delimiter between different cryptoassets
depth	integer	Orders depth quantity: [0, 100, 200, 500...]. 0 returns full depth. Depth = 100 means 50 for each bid/ask side. Note that for more liquid or closely priced pairs, the lack of order depth may result in miscalculation of depth/spread.

Example query:
https://api.nestex.one/cg/orderbook/BTC_USDT?depth=10

Example output:


```
{
  "ticker_id": "BTC_USDT",
  "timestamp": 1742007300000,
  "bids": {
    "83041.57253037": 0.000001,
    "82957.94556911": 0.000002,
    "82874.31860785": 0.000002,
    "82790.69164659": 0.000002,
    "82707.06468533": 0.000002,
    "82623.43772407": 0.000002,
    "82539.81076281": 0.000001,
    "82456.18380155": 0.000002,
    "82372.55684029": 0.000002,
    "82288.92987903": 0.000002
  },
  "asks": {
    "84794.92222999": 0.000001,
    "84714.11175555": 0.000003,
    "84630.48479429": 0.000002,
```

```
      "84546.85783303": 0.000002,
      "84463.23087177": 0.000002,
      "84379.60391051": 0.000001,
      "84295.97694925": 0.000002,
      "84212.34998799": 0.000001,
      "84128.72302674": 0.000001,
      "84045.09606548": 0.000001
    }
}
```


Order book response descriptions:

Name	Data Type	Description
ticker_id	string	A pair such as "BTC_ETH", with delimiter between different cryptoassets
timestamp	timestamp	Unix timestamp in milliseconds for when the last updated time occurred.
bids	decimal	An array containing 2 elements. The offer price and quantity for each bid order
asks	decimal	An array containing 2 elements. The ask price and quantity for each ask order

Endpoint 3 - /tradebook (Trade book details)

URL - https://api.nestex.one/cg/tradebook

The /tradebook/ticker_id endpoint is to provide trade information with page length = 100 (max 100 pages) returned for a given market pair/ticker.

For example https://api.nestex.one/cg/tradebook/BTC_USDT provides the BTC_USDT pair trade book

Endpoint parameters:

Name	Type	Description
ticker_id	string	A ticker such as "BTC_ETH", with delimiter between different cryptoassets
page	integer	Pagination. Default page id = 1, Max page id = 100

Example query:
https://api.nestex.one/cg/tradebook/AEGS_USDT?page=10

Example output:


```
{
   "ticker_id": "AEGS_USDT",
   "page": 10,
   "data": [
      {
         "trade_id": 8005701,
         "side": "BUY",
         "price": 0.00035464,
         "quantity": 28.197535,
         "timestamp": 1754803891280
      },
       .... snip ....
      {
         "trade_id": 7997798,
         "side": "SELL",
         "price": 0.00029931,
         "quantity": 218.847658,
         "timestamp": 1754782983000
      }
   ]
}
```

Order book response descriptions:

Name	Data Type	Description
ticker_id	string	A pair such as "BTC_ETH", with delimiter between different cryptoassets
page	integer	Pagination. Expected to be the same as input unless the input was missing or invalid
timestamp	timestamp	Unix timestamp in milliseconds for when the last updated time occurred.
trade_id	bigint	Unique id of the trade
side	string	BUY or SELL
price	decimal	Price at which this trade was fulfilled
quantity	decimal	Quantity of asset bought/sold in this trade

Endpoint 4 - /Fees (Fees for each asset)

URL - https://api.nestex.one/cg/fees

This provides the fees for each asset, denominated in both the asset itself and in USDT

There are no input parameters required. The output is a simple JSON array

Example query:
https://api.nestex.one/cg/fees

Example output:


```
[
  {
    "coin": "ALGO",
    "name": "Algorand",
    "minfee": "3",
    "maxfee": "3",
    "minfeeusdt": "0.267938",
    "maxfeeusdt": "0.267938"
  },
… ]
```


Fees response descriptions:

Name	Data Type	Description
coin name	string string	The asset in question, ticker & the name.
minfee minfeeusdt	numeric	The minimum fee for this asset across all of it's chains minfee is in the asset itself, while minfeeusdt is it's USDT value. In the example above, $ALGO has a withdrawal fee of 3 $ALGO which is equivalent to 0.267938 USDT
maxfee maxfeeusdt	numeric	The maximum fee for this asset. So in the example above, $ALGO has a withdrawal fee of 3 $ALGO which is equivalent to 0.267938 USDT AND the min/max are equal so there's just one price and just one chain

Endpoint 5 - /Liquidity (liquidity of a given asset & LP leaderboard)

URL - https://api.nestex.one/v1/liquidity/ASSET

This provides the liquidity information for the given asset, and a brief leaderboard of the top 5 liquidity contributions

There are no input parameters required. The output is a simple JSON array
Example output:


```
{
  "success": true,
  "data": {
    "score": "134",
    "total": "1343.10",
    "pooledCoin": "945.1403080",
    "pooledUsdt": "671.553824",
    "growth": "5.2",
    "dump": "false",
    "leaderboard": [
      "663.96", "345.12", "76.98", "27.88", "7.89"
    ]
  }
}
```


Response descriptions:

Name	Data Type	Description
score	numeric	A rough liquidity score assigned by the system
total	numeric	Total liquidity, in USDT
pooledCoin/pooledUsdt	numeric	Pooled coin and USDT
growth	numeric	How much the K for the pool has grown in %
dump	boolean	Dump fee applicability
leaderboard	numeric array	USDT value of the top 5 contributors If someone made an LP with $1 + 50 coins logically their contribution would be approximated to $2 (since the liquidity pool logic ties in that way)

Endpoint 6 - /Voteinfo

URL - https://api.nestex.one/v1/voteinfo

Simple list of assets that can be voted and current number of votes.
There are no input parameters required. The output is a simple JSON array

Example output:


```
{
  "success": true,
  "data": [
    {
      "coin": "INC",
      "name": "Incognito",
      "img": "inc.png",
      "votes": 20
    },
…]
```


Response descriptions:

Name	Data Type	Description
coin name img	string string string	The asset in question, ticker & the name. The image implies https://cdn.nestex.one/img/coins/[image-file-name] however you should store it locally and not hot-link
votes	numeric	Number of votes received