WebSocket Market Data API
(Real-time + Historical)

Technical Specifications

20 Feb 2025

V2.6

Table of Contents:

1. Change History

Version	Date	Notes
0.9	28 Jan 2020	First version (Beta)
1.0	02 Feb 2020	Master file can also be downloaded
1.1	08 Feb 2020	ATP, Bid Ask, Added in RT feed
1.2	20 Feb 2020	Symbol Subscription plus Touchline updates
1.3	05 Mar 2020	Bid Ask Upgrades, Historical Data Updates
1.4	07 Apr 2020	Added field "O","H","L",""to trade header
1.5	06 May 2020	• Added subscription type in login response • Added Tick sequence number field in tick data to track tick drops at client end • Rearranged the sequence of the fields to enable data flow with and without Bid / Ask • Added touchline fields previous close and previous day open interest close • Replaced Total Traded volume in place of Turnover in tick streaming • Added 15min/30min history • Add Symbol ID, Symbol name & Internval in History response to identify & match the request with the response. • Added 1min streaming Subscription type • Deprecated "mysymbols" method with new "touchline" method
1.6	02 Aug 2020	• Added Heartbeat messages for Real time stream • Histories now available for Tick / 1/2/3/5/10/15/30/60 mins along with EOD • Added last n bars retrieval with gethistorylastnbars method • Added Market Messages • Search for a symbol in the masters using a keyword • Sandbox page additions • Python Library - Tick Sequence number

		• Python Library – Tick Type Added • Python Library – Last n bars • Python Library – Heartbeat control
1.7 1.8	18 Dec 2020	• Tick + 1 min simultaneous feed enabled • Touchline / Add symbol format updated • Touchline / Add symbols – fields added • Trade (Real Time Tick) – fields added • Option chains - Symbol list download • Heartbeat timestamp & format updated • Market messages format updated • Get Market Status – Open / Closed • Get EOD Data availability status • Get Bhavcopy download • Force logout enabled • Search for a particular symbol • Historical Data- Rate limits set • Historical data – Idle disconnect
1.9	01 Mar 2020	• REST History Service enabled • (WebSocket History Service Deprecated) – Would be completed disabled from 30 Apr 2021
2.1	21 Mar 21	• Get Bhavcopy Status, get Bhavcopy, Gainers / Loser added.
2.2	02 Apr 21	• Error codes updated
2.3	09 Sep 21	• Document updated
2.4	06 July 23	• BSE Bid-Ask Level 2 message updated
2.5	06 Feb 24	• Greeks feed
2.6	20 Feb 25	• Document Update

2. About TrueData

A. **TrueData Financial Information Pvt. Ltd**. is a leading provider of real-time market data APIs for Indian stock and commodity exchanges. As an official real-time data vendor for **NSE, BSE, and MCX**, we offer **ultra-low latency, high-accuracy data** solutions designed for traders, investors, algo traders, and financial technology platforms

B. Since our inception in 2007 with the launch of True Data Financial Information Pvt Ltd., we have been at the forefront of market data innovation. Our proprietary APIs, along with our **flagship product Velocity**, provide seamless real-time and historical data access through advanced plugins and direct exchange feeds.

C. Our APIs offer extensive market coverage, including **corporate action-adjusted market data, corporate announcements, option Greeks, financial reports, and shareholding patterns**. To simplify integration for trading platforms and fintech developers, we provide **ready-made** API modules, **such as symbol master, option chain, top gainers/losers, company logos, 52-week high/low**, and **much more**.

D. Driven by our motto, **"Positioning You for Profits!"**, TrueData empowers traders with **accurate real-time data, advanced analytics, and seamless API integration**, ensuring **better decision-making and execution efficiency**.

3. The TrueData API Advantage

TrueData continuously innovates to provide **best-in-class APIs and data integration solutions** that meet the evolving needs of **professional traders, fintech platforms, and algo trading systems**.

Key Features & API Offerings

1. Ultra-Low Latency Real-Time Market Data Feeds

- Direct connectivity to NSE, BSE, and MCX ensures lightning-fast data delivery.
- Optimized for high-frequency trading (HFT), algo trading, and automated systems.
- Institutional-grade data feeds with minimal latency, ensuring accurate trade execution.

2. Advanced Data Infrastructure for High Performance

- Our backend is optimized for handling massive volumes of tick data efficiently.
- 99.9% uptime ensures reliable, stable, and uninterrupted market data access.
- High-speed delivery with minimal packet loss, crucial for professional trading strategies.

3. Seamless REST API Access

- Easily integrate real-time and historical data into trading platforms, research tools, and mobile apps.
- REST APIs provide access to: real-time price updates, corporate actions, option chain data, financial reports, and more.
- Flexible data formats (JSON, CSV, Binary) for easy customization based on user needs.

4. Developer-Friendly Libraries & SDKs

- We offer plug-and-play libraries to make integration effortless:
- Python: TrueData Python Library – Ideal for algo trading, backtesting, and research.
- Node.js: TrueData Node.js Library – Perfect for real-time streaming and web-based trading applications.
- C# .NET: TrueData .NET Library – Designed for Windows-based trading systems and enterprise applications.

5. Ready-Made API Modules for Easy Integration

- TrueData provides pre-built API modules for seamless integration with trading platforms:
- Symbol Master API – A complete database of all tradable instruments on NSE, BSE, and MCX.
- Option Chain API – Live option chain data with bid-ask spreads, open interest, IVs, and Greeks.

- Top Gainers/Losers API – Quickly fetch top-performing and worst-performing stocks based on real-time metrics.
- Company Logos API – Retrieve official stock symbols with logos for a visually enhanced user experience.
- 52-Week High/Low API – Identify stocks hitting their yearly highs or lows for trend analysis.
- More API Modules Continuously Added for advanced market insights.

6. Scalable & Flexible Data Streaming – No Symbol Limitations

- TrueData provides unlimited real-time market data streaming with two powerful modes:
- Limited Feed (Pub/Sub Model) – Best for retail traders and small-scale algo traders. Users subscribe to specific symbols and receive real-time price updates, optimizing bandwidth and efficiency.
- Full Market Streaming – Built for HFTs, institutions, and fintech platforms. As soon as a user connects, all trades for the subscribed segment stream automatically—tick by tick—without any symbol limitations. This mode delivers the full depth of the market in real time, making it perfect for market-making, algo trading, and deep market analysis.
- No restrictions. No delays. Just pure, real-time market action.

7. Streaming Candle Data API

- Real-time candle updates for 1-minute and 5-minute bars for live trend analysis.
- Perfect for intraday traders, scalpers, and automated strategies.
-

8. REST API Access to Historical Market Data

- Access to tick-by-tick, intraday, and EOD data via REST APIs.
- Data is fully adjusted for corporate actions (splits, bonuses, dividends, rights).
- Note: Historical market data access is available via our APIs, subject to compliance with exchange regulations.

9. Corporate Announcements & Financial Reports APIs

- Get real-time earnings reports, shareholding patterns, and key disclosures.
- Integrate fundamental analysis into trading strategies.

10. Option Greeks API

- Real-time Delta, Gamma, Theta, Vega, and Rho for precise option trading strategies.

11. Continuous & Contract Futures API

- Access both continuous futures and specific contract futures simultaneously (e.g., NIFTY-I and NIFTY20MARFUT).
- Essential for backtesting, technical analysis, and automated trading.

12. Premium Adjusted Continuous Futures API

- Seamless long-term analysis with adjusted continuous futures at rollover (NIFTY-IP).

13. Extensive Market Coverage & Real-Time Data Access

- Tick-by-tick data for 60,000+ symbols across NSE, BSE, and MCX.
- Instant access to intraday and EOD data, covering all market segments

14. Unmatched API Support & Customer Service

We provide dedicated API support through multiple channels.

- Live Chat & Phone Support – Get real-time help from our experts.
- Email & Ticketing System – Submit queries and receive fast resolution.
- Remote Setup via TeamViewer – Hassle-free API integration assistance.
- Comprehensive Knowledge Base & FAQs – Self-service troubleshooting for smooth implementation.

Why Choose TrueData?

1. Official Data Vendor for NSE, BSE & MCX
2. Ultra-Low Latency & High-Performance APIs
3. Seamless REST API & Streaming Access
4. Scalable, Enterprise-Ready Solutions
5. Comprehensive Market Data Coverage
6. Dedicated Technical & Developer Support

4. API Specifications

a) Overview

i) The TrueData API provides Real Time streaming data via WebSocket while the historical data is provided via REST. A client would need to connect to the respective WebSocket port for which they wish to receive the real time data. In case historical data is required, then the client would need to login to the REST service using the same login details and obtain the authentication token (bearer token) which would be needed for each subsequent request.

ii) The API is provided under two environments. One is the sandbox and the other being the production environment. All trials and access to real time data during the development period / initial integration period would be provided on the sandbox environment. Once you complete your integration, you would be shifted to the Production ports. Please change the codes in your codes accordingly.

iii) The only difference between the two live data ports would be as follows: -

a)	Sandbox Environment Port(s)	–	Real Time – 8086
b)	Production Environment Port(s)	–	Real Time – 8084

iv) Please use this sandbox webpage to test the feed during the development phase and to keep a track of / compare your own coding results

a) https://wstest.truedata.in/

b) We have added a few buttons on this page to help you to start testing your code and seeing the data flow immediately.

c) Please use this page to test the raw requests and responses in case your code has some issues. This is recommended as the first step before you raise a ticket for your issue.

b) Streaming WebSocket (Real Time)

i) Streaming WebSocket URL (Real Time)

Streaming WebSocket URL	wss://push.truedata.in:**real_time_port**?user=your_user_id&password=your_password

ii) Authentication Response

a) A successful login response returns the details as shown below along with the details of segments subscribed and the type of streaming subscription enabled – which is tick

/ 1 min/ 5 min. (Note that if you are subscribed for tick, you would not receive 1 min or 5min streaming bars and vice versa.)

b) If the client is connected elsewhere, the client would not be able to connect. You would first need to logout from your previous session and then attempt a login.

Response Success (JSON)	{
	“success”: true,
	“message” :“TrueData Real Time Data Service”,
	“segments” : [“nseeq”, “nsefo”, “mcx”, “cds”],
	“maxsymbols” : 250,
	“subscription” : “tick/1min/tick+1min/tick+5min/tick+1min+5min”,
	“validity” , “31/12/2020T23:30:00”
	}

Response Binary

Field	**Size**	**Value**	**Location**
Msg Code (char)	1 Byte	A	0
Status (boolean)	1 Byte	0 or 1	1
Message (str)	31 Byte	TrueData Real Time Data Service	2
Segment (str)	60 Byte	“NSEEQ”,”NSEFO”,…	33
Max Symbols (int)	4 Byte	900	93
Subscription(str)	20 Bytes	Tick+1min	97
Validity (str)	10 Bytes	Yyyy-MM-dd	117
Total Bytes	**128**		

Response Failure	{
	“success”: false,
	“message” :“invalid login credentials / user subscription expired / invalid request”,
	“segments” : null,
	"maxsymbols":0,
	“subscription” : null,
	"validity":"0001-01-01T00:00:00"}
	}

Response Duplicate Login	{
	"success":false,
	"message":"User Already Connected",
	"segments":null,
	"maxsymbols":0,
	“subscription” null,
	"validity":"0001-01-01T00:00:00"}
	}

iii) Heartbeat Messages

a) Heartbeat messages have been added to the Real time stream (Ports 8084/8086). A heartbeat shall be sent at every 5-6 seconds interval.

b) As soon as the client is connected to the Real time port, you would start receiving heartbeat messages along with the time stamp, as shown below.

c) Please note that Heartbeat messages are not sent over REST for the Historical data port. These are only sent over the Real time streaming ports.

Heartbeat JSON	{"success":true, "message": "HeartBeat", "success": true, "timestamp": "YYYY-MM-DDTHH:MM:SS.FFF"}
Heartbeat Binary	

Field Name	**Size**	**Value**	**Location**
Msg Code (char)	1 Byte	H	0
Status (boolean)	1 Byte	0 or 1	1
Unix timestamp (long)	8 Bytes	1730828522088	2
Total Bytes	**10**		

iv) Market Messages

a) Market messages indicating the start and stop of the various market sessions have been added for NSE. The following messages are pushed automatically in the real time WebSocket streams: -

- NSE EQ Pre Open Start
- NSE EQ Pre Open End
- NSE EQ Normal Market Start
- NSE FO Normal Market End
- NSE EQ Post Close Start
- NSE EQ Post Close End
- NSE FO Normal Market Start
- NSE FO Normal Market End
- MCX Market Status: 5 Session No: 1
- MCX Market Status: 6 Session No: 1
- MCX Market Status: 1 Session No: 2
- MCX Market Status: 2 Session No: 2

Market Messages JSON	{“success” : “true” , "message":"marketstatus","data":"NSE EQ Normal Market Start"}
	
	{“success” : “true” ,"message":"marketstatus","data":"NSE FO Normal Market Start"}

	{"success":true,"message":"marketstatus","data":"MCX Market Status: 1 Session No: 2"}
Market Status Message Binary	

Field	**Size**	**Value**	**Location**
Msg Code (char)	1 Byte	M	0
Status (boolean)	1 Byte	0 or 1	1
data (string)	60 Bytes	NSE EQ Normal Market Start	2
Total Bytes	**62**		

b) Other than the market messages which are pushed automatically, you can also check at any time if a particular market is open or closed.

Request	{
	"method”: "getmarketstatus"
	}

Response	{
	"success":true,
	"NSE_EQ":"CLOSED",
	"NSE_FO":"CLOSED",
	"NSE_CDS":"OPEN",
	"MCX":"OPEN"
	}

v) Symbol Subscribe

a) The symbols once subscribed are referred to by their symbol ids in the real time feed.

b) When you subscribe for a symbol with the request as shown below, a successful response would return the touchline, which would have the symbol id. This symbol id would need to be mapped to that symbol. This would help you to identify the data in the stream to that symbol.

c) As mentioned above, the symbol subscribed return includes a snap shot (touchline) information about the symbol till that very moment. This is also helpful post market hours to get the snapshot of the market at close of trade for reference till the next trading day open. The touchline is explained in detail at Sl. No. vii below

d) **Search for a symbol / symbol keyword in the masters -** Another way to get the symbol id list would be to search for the symbols (recommended) or request for the entire masters (not recommended) from the following mentioned URL to get the master files in json format: -

https://api.truedata.in/getAllSymbols?segment=eq&user=your_user_id&password=your_password&search=RELIANCE

e) **Download entire Master files -** You can also download the entire Master files if needed. This however, may not be needed as Symbol ID is already returned when you first subscribe to the symbol for the streaming feed.

https://api.truedata.in/getAllSymbols?segment=fut&user=your_user_id&password=your_password&csv=true&allexpiry=false

Please change the segment in this URL as desired by you: -

✓	segment=eq	–	Equity Master
✓	segment=fo	–	Futures & Options Master
✓	segment=in	-	Indices Master
✓	segment=fut	–	Futures (Only) Master
✓	segment=mcx	–	MCX Master
✓	segment=all	–	Full master (all segments)
✓	segment=bseeq	-	BSE Equity Master
✓	segment=bsefo	--	BSE F&O Master

**Please note that it is not necessary to download the master files as the symbol ID is returned once a symbol subscription request is sent (as shown below).*
**The instrument list API returns large amounts of data. It's best to request it once a day (ideally at around 08:30 AM) and store in a database at your end.*
**JSON format – removing &csv=True would give you the response in JSON format. Note that this would increase the download size and could take longer to download*
**&allexpiry=true would download all the symbols (including all expired symbols) in the TrueData database. This could be a huge list and it is recommended to use the &search command instead as explained above.*

f) **Get Option Chain Symbols -** If you need to get all the symbols of an Option chain, you can get those by calling the getOption Chain command. You need to specify the symbol and the expiry date in yyyymmdd format. Default format for the option chain is json. If you need the option in csv format add “&csv=true” in the end.

https://api.truedata.in/getOptionChain?user=your_user_id&password=your_password&symbol=NIFTY&expiry=20200917&csv=true

g) Add symbol request and response is now updated to an array. Many additional fields have been added like timestamp, Bid, Ask. Except for the message > symbols added, the format of the message is same as the touchline message. For detailed format of the message see the touchline fields as explained in Sl. No. vii. An example of the return when symbols are added is as below: -

Request	{ "method" : "addsymbol", "symbols" : ["NIFTY BANK", "NIFTY-I", "RELIANCE20DECFUT", "CRUDEOIL-I"] }
Response Success	{ "success":true,

	"message":"symbols added", "symbolsadded":4, "symbollist": [["NIFTY BANK","200000004","2020-12-17T08:40:29","30698.4","0","0","0","30920.35","30932.25","30587.1","30698.4","0","0","0","0","0","0"], ["NIFTY-I","900000596","2020-12-17T09:17:52","13701.8","300","13695.74","206850",13710.15","13715.2","13686.9","13699.45","12538050","2832963189","13701.25","450","13701.9","375"], ["RELIANCE20DECFUT","300173228","2020-12-17T09:17:52","1996","12625","1991.2","448945","1996","1996.2","1985.3","1984.6","35465140","0","1995.55","1010","1996.55","1010"], ["CRUDEOIL-I","950000072","2020-12-17T09:17:52","3554","0","3554.48","209","3555","3558","3551","3520","1057","74288600","3553","40","3554","19"]], "totalsymbolsubscribed":4 }

Response Failure	{
	"success": false,
	"message" : "symbol limit reached / invalid request"
	}

vi) Symbol Unsubscribe

Request	{
	"method": "removesymbol",
	"symbols": ["MINDTREE"]
	}

Response Success	{
	“success”: true,
	“message” :“symbols removed”,
	“symbolsremoved” : 1,
	}

Response Failure	{
	“success”: false,
	“message” :“invalid symbol/invalid request”
	}

vii) Touchline/Snap-quote Message

a) Touchline messages comes automatically before market begins, at MCX Pre-open Start (Normally 8:45AM) followed by NSE Pre Open Start (Normally 9:00 AM) and also at 30 seconds after NSE Pre Open Close (Normally around 09:08 am). The aim to push these messages is mainly to update the previous Close & the Prev OI close prior to market open.

b) The touchline message is similar to the symbols added message, which is now formatted as an array in an array. The touchline message is pushed into the real time WebSocket streams at the times mentioned above and also when symbols are added:-

Symbol, SymbolID, LastUpdateTime, LTP, TickVolume, ATP, TotalVolume, Open, High, Low , Previous Close , Today's OI , Previous Open Interest Close, Turnover, Bid , BidQty , Ask , AskQty

c) Note that the requesting for touchline method ("method": "touchline"), has been deprecated. Touchline is now pushed automatically at certain times as listed above without the requirement of a request. Also, this is sent whenever a symbol is subscribed as explained above. Please find the message format below:-

Touchline	{ "success":true, "message":"touchline", "symbolsadded":4, "symbollist": [["NIFTY BANK","200000004","2020-12-17T08:40:29","30698.4","0","0", "0","30920.35","30932.25","30587.1","30698.4","0","0","0","0","0","0"], ["NIFTY-I","900000596","2020-12-17T09:17:52","13701.8","300","13695.74", "206850",13710.15","13715.2","13686.9","13699.45","12538050", "2832963189","13701.25","450","13701.9","375"], ["RELIANCE20DECFUT","300173228","2020-12-17T09:17:52","1996", "12625","1991.2","448945","1996","1996.2","1985.3","1984.6","35465140","0","1995.55","1010","1996.55","1010"], ["CRUDEOIL-I","950000072","2020-12-17T09:17:52","3554","0","3554.48", "209","3555","3558","3551","3520","1057","74288600","3553","40","3554","19"]], "totalsymbolsubscribed":4 }

viii) Tick Streaming Data Response (Real Time)

a) Tick Streaming Data needs to be allowed from the True Data backend. This is allowed by default (Data subscription is tick).
b) The stream contains two main messages type header – 'trade' and 'bidask'
c) The 'bidask' header needs to be also allowed from the backend. This is allowed by default in case of Tick Streaming data is enabled. However, this could be disabled in case it is not required by a client to reduce the flow / clutter in the feed.
d) The 'trade' header also contains a Special tag character which contains **"O", "H","L" or ""** as a field, signifying if the tick is an open tick for the day, day' s high or day' low. If

it’s none of these it is demarcated with nothing. An **“OHL”** Tag has also been added in the ases where we have sent the first tick (Open) itself along with a new High & also a New Low.
e) Tick Sequence number provided helps you verify that no Ticks are lost or packets dropped and that you were able to handle all the packets received.
f) Implementing an asynchronous WebSocket client may be a complex task. We recommend using our pre-built client libraries in Python. If not done properly this could fill up the receive buffer at your end and the send buffer at the server end leading to packet drops.
g) In case **both Tick Streaming & Bid / Ask are activated (Allowed by default)**, then the following response would be sent to the user: -

i) The response with the “**trade**” header is sent whenever there is a trade in the subscribed symbol, and it includes the following fields: -

Symbol ID, Date Time (Timestamp), LTP, LTQ, ATP (Average Traded Price – for the day), TTQ (Total Traded Qty – Volume for the day), Open, High, Low, Prev Close, OI (Open Interest), Prev Open Int Close, Day’s Turnover, Special Tag (”O/H/L” or “”), Tick Sequence No, Bid, Bid Qty, Ask, Ask Qty

Response JSON (with bid/ask)	{"trade":["100000995","2020-12-16T14:02:32","1472.8","635","1475.83","680949","1475.05","1484","1463","1468.35","0","0","1004964962.67","","4775","1472.8","429","1473.3","34"]}
Trade Binary	

Field	**Size**	**Value**	**Location**
Msg Code (char)	1 Byte	T	0
Symbol Id (int)	4 Byte	100001262	1
Time stamp (int)	4 Byte	**1730829992**	5
LTP (float)	4 Byte	**1472.8**	9
Volume(int)	4 Byte	**635**	13
ATP(float)	4 Byte	**1475.83**	17
Tot Volume(long)	8 Byte	**680949**	21
Open(float)	4 Byte	**1475.05**	29
High(float)	4 Byte	**1484**	33
Low(float)	4 Byte	**1463**	37
Prev Close(float)	4 Byte	**1468.35**	41
OI(long)	8 Byte	**0**	45
Prev OI(long)	8 Byte	**0**	53
Turnover(double)	8 Byte	1004964962.75	61
OHL(1 character)	1 Byte	O	69
Sequence No(int)	4 Byte	12345	70
Bid(float)	4 Byte	1472.8	74
Bid Qty(int)	4 Byte	429	78
Ask(float)	4 Byte	1473.3	82
Ask Qty	4 Byte	34	86
Total Bytes	**90**		

ii) The response with the “**bidask**” header is sent whenever there is only a Bid / Bid Qty / Ask / Ask Qty update without any trade having been executed in the subscribed symbol and it includes the following fields: -

Symbol ID, Date Time (Timestamp), Bid, Bid Qty, Ask, Ask Qty

Response JSON	{"bidask":["950000606","2/18/2020 3:43:45 PM","3698","34","3700","54"]}

h) In case **only Tick Streaming is activated & Bid /Ask are deactivated** on request, then the following response would be sent to the user: -

i) The response with the "**trade**" header is sent whenever there is a trade in the subscribed symbol, and it includes the following fields: -

Symbol ID, Date Time (Timestamp), LTP, LTQ, ATP (Average Traded Price – for the day), TTQ (Total Traded Qty – Volume for the day), Open, High, Low, OI (Open Interest),PrevDay Open Int, Special Tag ("O/H/L" or ""), Tick Sequence No

Response JSON (without bid/ask)	{"trade":["100000995","2020-12-16T14:02:32","1472.8","635","1475.83","680949","1475.05","1484","1463","1468.35","0","0","1004964962.67","","4775"]}

ii) There would be **no "bidask"** header in this response. This needs to be disabled from **our backend** as it is enabled by default.

iii) In case of BSE bid, response with the "**bidaskL2**" header is sent whenever there is 5 "**No of Bids / Bid / Bid Qty/ No of Asks / Ask / Ask Qty** " update without any trade having been executed in the subscribed symbol and it includes the following fields: -

Symbol ID, Date Time (Timestamp),
#no of bids1, #bidprice1 , #bidqty1 , #no of ask1, #askprice1, #askqty1 ..
…
(**5 times**),
…
#total bid qty , # total ask qty

Response JSON	{"bidaskL2": ["490000010","2023-07-05T09:15:44", "0","65280","10","1","65540.75","10", "1","65120.75","10","1","65980","50", "1","65050","50","1","65989","50", "1","65010","50","1","0","0", "0","0","0","0","0","0","0", "120" , "110"]}

ix) Realtime Greeks Response (on request)

a) Greeks in websocket needs to be enabled from backend
b) Each message contains an array which has following things symbolid, timestamp , delta , gamma, theta, vega, rho, IV
c) Greeks would come up to 4 digits after decimal places

Response JSON	{"greeks":["301680343","2024-02-14T09:42:02","0.2015","0.0331","-6.0417","0.0005","0.8335","0.0198"]}

x) 1Min Streaming Bar Data Response (Real Time)

a) Bid / Ask Streaming does not form a part of 1 min streaming subscription.
b) 1 Min streaming can be enabled upon request
c) The number of symbols streamed would depend upon your subscription
d) Format shall be {“bar1min”, [symbolid, timestamp, open, high, low, close, volume, oi]}

Response JSON (Compressed GZIP)	{"bar1min": ["950000114", "2020-05-06T09:06:00", "45691", "45698", "45687", "45693", "45", "7602"]}

xi) Logout/Disconnect WebSocket (for Real Time & Historical)

a) Logging out from the Real time WebSocket connection **would now not force a logout from the historical WebSocket** connection or vice versa. Meaning that you would need to logout of the real time & historical ports separately.
b) However, note that, in case of no request on the Historical Data Port, the **History Data WebSocket** would get disconnected after an **idle time of 90 minutes**. For a subsequent history request in that case you would need to re-login to the Historical data WebSocket port.

Request	{ "method": "logout" }
Response Success	{ “success”: true }

xii) Forced Logout

a) There may be occasions when you are logged in / connected via WebSocket elsewhere and have forgotten about it and are then trying to reconnect again at a different location with your same id or password.
b) In this case you would get a “User Already Connected” message.
c) This could also happen if you have had a dirty disconnect from your previous session or that your coding environment has not cleanly disconnected your previous session.
d) In such a case you can use the force logout command.

i) Make sure that all your sessions are closed. This is especially important in case automated reconnect to the WebSocket has been enabled by you.

ii) Request you to fire the below-given URL(s) to force logout from all other places that you might have been logged in.

iii) Wait for 1 min after firing the URL and then try to log in and you should not get this error message.

iv) **URL for Real-Time Connection:** https://api.truedata.in/logoutRequest?user=xxx&password=yyy&port=8082
(note port as per your concerned port)

v) For the historical port a re-login after 3600 seconds would provide you with a fresh authorization token.

c) Historical Data over REST

i) Historical Data API & URL

a) History URL for login / authentication > https://auth.truedata.in/token

b) History URL for subsequent requests > https://history.truedata.in/...

c) History tokens have a validity of 3600 seconds. A new token needs to be obtained in time in case further requests are required to be made.

d) Request rates are currently limited for tick history as follows. (This includes all REST calls): -

Tick Data >> Per Second - 5, Per Minute - 300, Per Hour – 18000
Minutes Bar Data >> Per Second - 10, Per Minute - 600, Per Hour – 18000

e) The access token is renewed once every day at around 4 am. A fresh token needs to be obtained at that time to continue using the service without any interruptions.

f) In case there is no data available for the date time range, given in the request, then you will get a message like so: -

"No Data exists for <Symbol>"

g) You can download the entire postman collection for quicker development from this link here: TrueData_History_REST_API_v1.1.postman_collection

h) You can pull the response in csv or json formats. You need to declare the response format needed in your request. However, a csv response is recommended as it lowers the amount of data transfer / bandwidth consumption and thereby the data lands up faster due to its comparably smaller size.

i) **The recommended and default response format is csv.** Json responses are also possible. We recommend choosing the csv response format as it ensures comparatively smaller bandwidth usage for the same historical data thereby enabling faster delivery over the internet.

j) Sample csv format is as follows:-

```
timestamp,open,high,low,close,volume,oi
```

```
2021-02-01T09:07,1859.4,1859.4,1859.4,1859.4,1,0
2021-02-01T09:15,1859.4,1862.3,1853.05,1856.95,578586,0
2021-02-01T09:16,1857.8,1861.6,1856.15,1860.4,240393,0
……
```

k) Sample json format is as follows:-

```
{
    "status": "Success",
    "Records": [
        [
            "2021-03-22T15:29:40",
            14756.55,
            150,
            8712225,
            14756.4,
            150,
            14757.0,
            375
        ],
        [
            "2021-03-22T15:29:42",
            14757.0,
            975,
            8712225,
            14756.5,
            300,
            14757.0,
            225
        ]]}
```

l) For the rest of the document we have depicted all responses in the recommended format which is csv.

m) To hasten your development process, the Postman collection for the Historical REST API is available from this link. Please unzip the file and import it into Postman (https://www.postman.com/) to work with it :-

TrueData_History_REST_API_v1.1.postman_collection

ii) REST Authentication

a) The first step in the authentication process is to obtain the token which can be used for subsequent requests. This is a POST request like so: -

Service URL	https://auth.truedata.in/token
Request Method	**POST**
Format	x-www-from-urlencoded

Request Parameters	username <user-id> password <user-pwd> grant_type **passoword**

b) The above request typically sends JSON response as follows: -

Response	JSON

Sample Response	{
	"access_token": "og6zhJqQZuLAT54CpwPFNI8yOhC
	RasyMiO6eMS62YF19A9tAMkgqJNP4LreC2vt_muHKpcMOisi
	wPZgHZNmxwvfplKvU3SVZJ2QMtET3oeAMT8smgAP9HIxLTYn
	QJIMJ_QB0xYWGe5efwGO4Kwun16nZ9GvNdQyPhPNed5t-oR-
	ajdg_bSGdzb8XusmMbctC5aiCE2w2dcknuWKm6v8follpLD6
	VvYBhEIHWxguePUuTfNztsZyVDyLn4p_-
	vlwmquxDtt5BHqQfSksq_zy9hA",
	"token_type": "bearer",
	"expires_in": 13961,
	"userName": "<your_username>",
	".expires": "Sun, 21 Mar 2021 22:30:00 GMT",
	".issued": "Sun, 21 Mar 2021 18:37:18 GMT"
	}

Response parameters	access_token : will be used in subsequent request to retireve data expires_in : token validity in seconds Token_type: bearer (type of token)

Error	{
	"error": "invalid_grant",
	"error_description": "The user name or password is incorrect."
	}

	{
	"error": "invalid_grant",
	"error_description": "User subscription expired."
	}

iii) Tick Data History

a) Tick Data can be retrieved using the following request. This is a GET request like so: -

Service URL	**https**://history.truedata.in/getticks? **symbol**=NIFTY-I&**bidask**=0&**from**=210224T09:00:00& **to**=210225T15:30:00&**response**=csv
Request Method	**GET**

Headers	*bearer* **your-token – supply token obtained from auth service in header**
GET Query string Parameters	**symbol**=symbol-name e.g. NIFTY-I or RELIANCE or CURDEOIL-I **bidask**=0/1 , set 1 to get bid/ask in response otherwise 0 **from**=yymmddTHH:mm:ss – from date **to**=yymmddTHH:mm:ss – to date **response**=csv/json – response type required

b) Above request typically sends CSV or JSON response as follows: -

Response	CSV/JSON
Sample Response Csv >	`timestamp,ltp,volume,oi,bid,bidqty,ask,askqty` 2021-02-24T09:15:00,14755.15,1,7390425,14730,225,14745.45,375 2021-02-24T09:15:00,14755.35,1,7390425,14730,225,14745.45,375 2021-02-24T09:15:00,14725,1,7390425,14730,225,14745.45,375 ….
Response parameters	time,ltp, tickvol, openinterest,bid,bidqty,ask,askqty
Error Response1	API calls quota exceeded! maximum admitted 1 per Second.
Error Response2	Segment not subscribed
Error Response3	Symbol does not exist
Error Response4	Authorization has been denied for this request.
Error Response5	No data exists for <symbol>

iv) Bar Data History

a) Bar Data History can be retrieved using the following request. This is a GET request like so: -

Service URL	**https**:// history.truedata.in/**getbars**?**symbol**=RELIANCE&**from**=210201T09:00:00&**to**=210224T15:30:00&**response**=csv&**interval**=1min
Request Method	GET
Headers	*bearer* **your-token – supply token obtained from auth service in header**
GET Query string Parameters	**symbol**=symbol-name e.g. NIFTY-I or RELIANCE or CURDEOIL-I **from**=yymmddTHH:mm:ss – from date **to**=yymmddTHH:mm:ss – to date **response**=csv/json – response type required **interval**=1min/2min/3min/5min/10min/15min/30min/60min

b) Above request typically sends CSV or JSON response as follows: -

Response	CSV/JSON
Sample Response	`timestamp,open,high,low,close,volume,oi`

	2021-02-01T09:07,1859.4,1859.4,1859.4,1859.4,1,0 2021-02-01T09:15,1859.4,1862.3,1853.05,1856.95,578586,0 2021-02-01T09:16,1857.8,1861.6,1856.15,1860.4,240393,0 ……
Response parameters	time,open,high,low,close,volume,openinterest
Error Response1	API calls quota exceeded! maximum admitted 1 per Second.
Error Response2	Segment not subscribed
Error Response3	Symbol does not exist
Error Response4	Authorization has been denied for this request.
Error Response5	No data exists for <symbol>

v) Last N Bars History

a) Last N Bars History is currently enabled for **1 min & EOD bars**. More bar sizes will be added in the future.
b) The Data of the last 'n' bars can be retrieved using the following request. This is a GET request like so: -

Service URL	**https**:// history.truedata.in/**getlastnbars**? **symbol**=NIFTY-I &**response**=csv&**nbars**=200&interval=1min&bidask=0
Request Method	GET
Headers	*bearer* **your-token – supply token obtained from auth service in header**
GET Query string Parameters	**symbol**=symbol-name e.g. NIFTY-I or RELIANCE or CURDEOIL-I **nbars**=1 to 200 (max 200) **response**=csv/json – response type **required** **bidask=0 (must be 0)** **interval**=1min/2min/3min/5min/15min/30min/60min/eod

c) Above request typically sends CSV or JSON response as follows: -

Response	CSV/JSON
Sample Response	`timestamp,open,high,low,close,volume,oi` 2021-02-01T09:07,1859.4,1859.4,1859.4,1859.4,1,0 2021-02-01T09:15,1859.4,1862.3,1853.05,1856.95,578586,0 2021-02-01T09:16,1857.8,1861.6,1856.15,1860.4,240393,0 ……
Response parameters	time,open,high,low,close,volume,openinterest
Error Response1	API calls quota exceeded! maximum admitted 1 per Second.
Error Response2	Segment not subscribed
Error Response3	Symbol does not exist
Error Response4	Authorization has been denied for this request.
Error Response5	No data exists for <symbol>

vi) Last n Ticks History

a) The Data of the last 'n' ticks can be retrieved using the following request. This is a GET request like so: -

Service URL	**https**:// history.truedata.in/**getlastnticks**? **symbol**=ACC& **bidask**=1& **response**=csv& **nticks**=2000&**interval**=tick
Request Method	GET
Headers	*bearer* **your-token – supply token obtained from auth service in header**
GET Query string Parameters	**symbol**=symbol-name e.g. NIFTY-I or RELIANCE or CURDEOIL-I **nticks**=1 to 200 (max 200) **response**=csv/json – response type **required** **interval**=tick – must be **tick**

b) Above request typically sends CSV or JSON response as follows: -

Response	CSV/JSON
Sample Response	`timestamp,ltp,volume,oi,bid,bidqty,ask,askqty` 2021-02-24T09:15:00,14755.15,1,7390425,14730,225,14745.45,375 2021-02-24T09:15:00,14755.35,1,7390425,14730,225,14745.45,375 2021-02-24T09:15:00,14725,1,7390425,14730,225,14745.45,375 ….
Response parameters	time,ltp, tickvol, openinterest,bid,bidqty,ask,askqty
Error Response1	API calls quota exceeded! maximum admitted 1 per Second.
Error Response2	Segment not subscribed
Error Response3	Symbol does not exist
Error Response4	Authorization has been denied for this request.
Error Response5	No data exists for <symbol>

Note: bid, bid_qty, ask, ask_qty only comes if requested & enabled

vii) Retrieve the Bhavcopy

a) The **Bhavcopy** of any particular date, can be retrieved. The token obtained in authentication would need to be used in this too, like in any other request. This is a GET request like so: -

Service URL	**https**:// history.truedata.in/ **getbhavcopy**?**segment**=MCX&**date**=2020-02-18&**response**=csv
Request Method	GET

Headers	*bearer* **your-token – supply token obtained from auth service in header**
GET Query string Parameters	**segment**=eq/fo/mcx/cds **date**=yyyy-MM-dd **response**=csv/json – response type **required**

b) Above request typically sends CSV or JSON response as follows: -

Response	CSV/JSON
Sample Response	`symbolid,symbol,open,high,low,close,volume,oi` 800000372,MCXCOMPDEX,11059.52,11133.83,10984.32,11110.59,0,0 800000373,MCXBULLDEX,14944.58,14956.36,14771.68,14877.5,0,0 …
Response parameters	symbolid,symbol,dopen,dhigh,dlow,dclose,volume,oi
Error Response1	Authorization has been denied for this request.
Error Response2	No data exists for <date>

viii) Retrieve the Bhavcopy Availability Status for the day

a) Before you can request for the Bhavcopy, you need to know if the Bhavcopy for the day has arrived or not.

b) You can retrieve the **Bhavcopy availability status** of today's bhavcopy or for that matter, for any particular date. The token obtained in authentication would need to be used in this too, like in any other request. This is a GET request like so: -

Service URL	**https**:// history.truedata.in/ /**getbhavcopystatus**?**segment**=FO&**date**=2020-02-20&**response**=csv
Request Method	GET
Headers	*bearer* **your-token – supply token obtained from auth service in header**
GET Query string Parameters	**segment**=eq/fo/mcx/cds **date**=yyyy-MM-dd **response**=csv/json – response type **required**

c) Above request typically sends CSV or JSON response as follows: -

Response	CSV/JSON

Sample Response	`segment,timestamp`
	`FO,3/2/2021 7:54:00 PM`

Response parameters	Segment, timestamp
Error Response1	Authorization has been denied for this request.
Error Response2	No data exists for <date>

ix) Retrieve the LTP of any symbol

a) The **Last traded price (LTP)** of any particular date, can be retrieved. The token obtained in authentication would need to be used in this too, like in any other request. This is a GET request like so: -

Service URL	**https**:// history.truedata.in/**getlastnticks**? **symbol**=RELIANCE& **bidask**=1&response=csv& **nticks**=1&**interval**=tick
Request Method	GET
Headers	*bearer* **your-token – supply token obtained from auth service in header**
GET Query string Parameters	**symbol**=symbol-name e.g. NIFTY-I or RELIANCE or CURDEOIL-I **nticks**=1 **response**=csv/json – response type **required** **interval**=tick

b) Above request typically sends CSV or JSON response as follows: -

Response	CSV/JSON

Sample Response	`timestamp,ltp,volume,oi,bid,bidqty,ask,askqty`
	`2021-03-03T15:30:00,15300,900,10911825,15299.95,150,15303,75`

Response parameters	timestamp,ltp,volume,oi,bid,bidqty,ask,askqty
Error Response1	API calls quota exceeded! maximum admitted 1 per Second.
Error Response2	Segment not subscribed
Error Response3	Symbol does not exist
Error Response4	Authorization has been denied for this request.
Error Response5	No data exists for <symbol>

5. Support Contact Details

Data Vendor Name	TRUEDATA FINANCIAL INFORMATION PVT. LTD
GSTIN	24AAECT8911G1ZD
Permanent Account Number	AAECT8911G
Address	309-310, Ugati Corporate Park, Opposite ICICI Bank, PIN - 382421
City	Gandhinagar
State	Gujarat
Email address of contact person	support@truedata.in
Client Support Number	+91-7304-22-44-66

Appendix I

6. Pre-built Client Library Details (Python)

a) You can run 'pip install truedata' to get the latest version of the library.
b) Refer this link for latest updates >> https://pypi.org/project/truedata/

7. Pre-built Client Library Details (nodejs)

s
c) You can run 'npm install truedata-nodejs' to get the latest version of the library.
d) Refer this link for latest updates >> https://www.npmjs.com/package/truedata-nodejs

8. Pre-built Client Library Details (.Net)

e) You can run 'dotnet add package TrueData-DotNet --version 1.4.6' to get the latest version of the library.
f) Refer this link for latest updates >> https://www.nuget.org/packages/TrueData-DotNet