IBKR TWS API data architecture for micro futures trading

Interactive Brokers provides **serviceable but limited** data infrastructure for algorithmic futures trading. The TWS API delivers historical bars up to 5 years back for 1-minute+ data, real-time tick-by-tick streaming, and 5-10 levels of market depth for CME products—but critically lacks historical Level 2 data, aggressor flags on trades, and millisecond timestamp precision. For micro futures like MNQ and MGC, data quality matches their E-mini counterparts with no micro-specific limitations beyond thinner order books.

The most significant architectural constraint is that **IB is fundamentally a broker, not a data vendor**. (Interactive Brokers) Traders building order flow strategies will find the absence of aggressor-side indicators and historical depth data forces reliance on third-party feeds like IQFeed or direct CME market data for advanced analysis.

Historical data delivers years of bars but only seconds of precision

The TWS API supports requesting historical bar data via (reqHistoricalData) with duration strings ranging from "60 S" up to "1 Y". For **bars of 1 minute or larger, approximately 5 years of data** is accessible. (Tradingsoftwarelab) However, IB enforces a tiered system: bars smaller than 30 seconds are limited to **6 months maximum lookback**, (github) and sub-second bars cap at roughly 2 days.

Step sizes limit how much data each request returns. A single request for 1-minute bars cannot exceed 1 day of duration, while daily bars can request up to 1 year. (github) The practical implication: bulk historical downloads require iterative requests with careful pacing.

Historical tick-by-tick data via (reqHistoricalTicks) returns a **maximum of 1,000 ticks per request** (Interactive Brokers) with types "TRADES", "BID_ASK", or "MIDPOINT". (github +2) Lookback extends approximately 2 years for expired futures, though tick timestamps resolve only to the nearest second—not milliseconds. This limitation matters for latency-sensitive strategies.

For expired futures contracts, IB retains **2 years of historical data** from the expiration date. (QuantConnect +2) Access requires setting (includeExpired=True) in the contract definition. (Interactive Brokers) Expired options and futures options data is entirely unavailable. (Interactive Brokers) (github)

Continuous futures ((CONTFUT) security type) automatically stitch successive front-month contracts for seamless historical analysis spanning rollovers. (Interactive Brokers) (Interactive Brokers) A critical caveat: **TWS v10.30+ requires leaving (endDateTime) empty** when requesting continuous futures data— (Interactive Brokers) a breaking change that catches many developers.

Pacing violations require disciplined request management

IB enforces strict rate limiting on historical data that cannot be bypassed:

- **Identical requests within 15 seconds** trigger violations

- **6+ requests for the same contract/exchange/type within 2 seconds** trigger violations
- **More than 60 requests in any 10-minute period** (for bars ≤30 seconds) trigger violations
- **BID_ASK requests count double** toward limits Interactive Brokers

Three pacing violations terminate the API session. Interactive Brokers The safest approach: space requests **15+ seconds apart** for the same contract and limit simultaneous open requests to 50. github Weekend/off-hours downloads typically experience faster response times due to reduced server load.

For bars of 1 minute or larger, IB has officially "lifted" hard pacing limits but maintains a soft throttle for load balancing. Interactive Brokers The effective maximum request rate equals MarketDataLines ÷ 2 per second— Interactive Brokers with 100 default market data lines, this means **50 requests/second** ceiling. Interactive Brokers interactivebrokers

Market depth provides limited real-time view without history

The reqMktDepth function streams real-time Level 2 data for futures, but **historical depth data is completely unavailable**. This fundamental limitation prevents backtesting order flow or market microstructure strategies using IB data alone.

Available depth levels depend on exchange subscriptions. CME Group futures (including MNQ and MGC) typically provide **5-10 levels** through standard depth subscriptions—not full book access. Interactive Brokers The function signature accepts a numRows parameter to request desired depth up to what's available:

```python
self.reqMktDepth(reqId=1001, contract=mnq_contract, numRows=10, isSmartDepth=False, mktDepthOptions=[])
```

Subscription limits follow the formula: MarketDataLines ÷ 3, Interactive Brokers meaning 100 market data lines allow approximately **33 simultaneous depth subscriptions**, capping at 60 maximum. github The isSmartDepth=True parameter (TWS v974+) aggregates DOM quotes across subscribed L1/L2 feeds IBKR Interactive Brokers but requires routing away from direct exchange specification.

Updates arrive via updateMktDepth and updateMktDepthL2 callbacks Interactive Brokers with operation codes (0=INSERT, 1=UPDATE, 2=REMOVE) and side indicators. Interactive Brokers Unlike reqMktData, depth data streams **without sampling or filtering**— Interactive Brokers every book change triggers a callback. Interactive Brokers

The reqMktDepthExchanges function returns exchanges where the user holds appropriate depth subscriptions, Interactive Brokers useful for programmatically discovering available markets. Interactive Brokers

Real-time streaming uses TCP sockets with aggregated snapshots by default

The TWS API uses **traditional TCP sockets, not WebSockets**. Interactive Brokers All data flows through TWS or IB Gateway as an intermediary: Exchange → IB Servers → TWS/Gateway → TCP Socket → EReader →

Message Queue → EWrapper callbacks. This architecture mandates a minimum two-thread design in client applications. github Interactive Brokers

Default ports: TWS live trading uses 7496, paper trading uses 7497. Interactive Brokers github IB Gateway uses 4001 (live) and 4002 (paper). github Maximum **32 simultaneous client connections** per TWS instance, each requiring a unique clientId. Interactive Brokers +3

A critical distinction exists between the two real-time data methods:

reqMktData sends aggregated snapshots approximately **every 250 milliseconds**—not true tick-by-tick data. AlgoTrading101 Worse, it does not include trade timestamps, making precise timing analysis impossible. AlgoTrading101 This is the source of significant confusion for developers expecting granular data.

reqTickByTickData provides actual tick-by-tick data matching TWS Time & Sales, Interactive Brokers with tick types "Last", "AllLast", "BidAsk", and "MidPoint". Interactive Brokers github Timestamps are included but resolve only to seconds. Simultaneous subscriptions follow the same Lines ÷ 3 formula as depth, and only one request per instrument is allowed within 15 seconds. Interactive Brokers

Connection persistence requires manual handling. Error code 1100 signals connectivity loss; 1102 signals restoration. Farm status messages (2104 for real-time, 2106 for historical) are informational, not errors. IB Gateway notably doesn't connect to market data farms until the first request—the yellow "inactive" indicator is expected behavior until then. Interactive Brokers github

Critical limitations traders often discover too late

No aggressor flag: TWS API does not indicate which side initiated a trade. The TickAttribLast structure contains pastLimit and unreported flags but no buy/sell aggressor indicator. This fundamentally limits order flow analysis compared to direct CME feeds.

No trade conditions via API: While TWS Time & Sales displays trade conditions, this information is explicitly unavailable through the API—IB documents this limitation directly. Interactive Brokers

Filtered volume: IB's historical data excludes combo trades, block trades, derivatives, and average price trades. Interactive Brokers Reported volume runs lower than unfiltered exchange feeds, affecting volume-based analysis accuracy. Interactive Brokers

Daily bar settlement quirks: Futures daily bar close prices may represent settlement price rather than last trade. Official settlement often arrives hours after session close—Friday settlements may not appear until Saturday. Interactive Brokers

Options tick data real-time gap: Historical tick-by-tick data is available for options, but **real-time tick-by-tick for options is not supported**— Interactive Brokers Interactive Brokers only the aggregated reqMktData works for live options quotes.

For micro futures specifically, MNQ historical data begins around **October 31, 2019** (approximately 5 months after the May 2019 launch). No fundamental data differences exist between micro and E-mini contracts in API

access—both require the same CME market data subscription and use identical GLOBEX routing.

Python libraries have consolidated around ib_async

The original ib_insync library was archived in March 2024 following creator Ewald de Wit's passing. github The actively maintained successor is **ib_async** Medium (github.com/ib-api-reloaded/ib_async), which requires Python 3.10+ and implements the full IBKR API binary protocol internally GitHub with production-ready reconnection logic. github

```python
from ib_async import IB, Future, ContFuture

ib = IB()
ib.connect('127.0.0.1', 7497, clientId=1)

# Specific expiry contract
mnq = Future('MNQ', '202403', 'CME')
ib.qualifyContracts(mnq)  # Required: validates and retrieves conId

# Historical backfill with live streaming continuation
bars = ib.reqHistoricalData(
    mnq,
    endDateTime='',
    durationStr='2 D',
    barSizeSetting='1 min',
    whatToShow='TRADES',
    useRTH=False,
    keepUpToDate=True  # Transitions to live updates automatically
)

# Event-driven updates
bars.updateEvent += lambda bars, hasNewBar: print(f"Bar: {bars[-1]}")
```

The official ibapi package remains available but requires implementing the callback-heavy EWrapper/EClient pattern with manual threading—significantly more complex for most use cases.

For **Node.js/TypeScript**, @stoqey/ib (npm: @stoqey/ib) provides the primary option with full TypeScript support, based on IB Java Client v10.32.01. It uses EventEmitter patterns and includes an experimental RxJS-based IBApiNext interface.

Rolling window architectures require gap-aware design

The recommended pattern combines historical backfill with live streaming using keepUpToDate=True:

```python
bars = ib.reqHistoricalData(
    contract,
    endDateTime='',      # Current time
    durationStr='2 D',   # Backfill window
    barSizeSetting='1 min',
    whatToShow='TRADES',
    useRTH=False,
    keepUpToDate=True    # Enable live bar updates
)

def onBarUpdate(bars, hasNewBar):
    if hasNewBar:
        process_complete_bar(bars[-2])  # Previous bar is now final
    else:
        update_current_bar(bars[-1])    # In-progress bar modified

bars.updateEvent += onBarUpdate
```

Gap detection requires comparing consecutive bar timestamps against expected intervals—IB does not guarantee data during brief exchange outages. Gap filling should respect 15-second pacing rules between requests for the same contract.

Timestamp synchronization matters: TWS returns timestamps in the timezone selected at login. Interactive Brokers Use ib.reqCurrentTime() to synchronize with server time. Historical data date formats vary by bar size: daily bars return yyyyMMdd only, while intraday includes time components.

Conclusion

The TWS API provides adequate infrastructure for retail algorithmic trading on micro futures, with **5 years of 1-minute+ bar history, real-time 10-level depth, and true tick-by-tick streaming** when using reqTickByTickData. The architecture's primary weaknesses—no historical L2, no aggressor flags, second-only timestamp resolution, and aggregated default tick data—reflect IB's core identity as a broker rather than data vendor.

For strategies requiring order flow analysis, historical depth reconstruction, or microsecond timing, supplement IB data with dedicated feeds. For swing trading, trend following, or strategies operating on 1-minute+ timeframes, the TWS API delivers sufficient data quality at compelling cost—CME market data subscriptions run approximately $10/month for non-professionals with commission waivers available.

The shift from ib_insync to ib_async represents the current state of the Python ecosystem; interactivebrokers both libraries abstract away the complexity of IB's callback architecture into intuitive, async-friendly interfaces ib_insync that dramatically reduce implementation time for futures trading systems.