# Technical Analysis of Phemex Market Data Infrastructure: Ingestion Protocols for OHLCV and Symbol Normalization

## 1. Executive Summary and Architectural Context

The reliability of quantitative trading systems and algorithmic market analysis is fundamentally dependent on the integrity of the data ingestion layer. In the context of the Phemex cryptocurrency exchange, developers and data engineers frequently encounter specific architectural idiosyncrasies that distinguish it from other major venues. This report provides an exhaustive technical analysis of the Phemex Application Programming Interface (API), specifically focusing on the retrieval, interpretation, and persistent storage of Open-High-Low-Close-Volume (OHLCV) candlestick data.

The analysis is driven by the necessity to address prevalent integration challenges, notably the corruption of price data due to improper handling of scaled integers, the misalignment of candlestick array indices, and the failure of retrieval routines caused by rigid symbol nomenclature. Phemex employs a high-performance architectural design that favors fixed-point arithmetic over floating-point representations to ensure numerical precision in its matching engine. While this design choice eliminates floating-point determinism errors, it imposes a strict burden of normalization on the client side. Furthermore, the bifurcation of the exchange into Spot and Perpetual (Contract) markets results in distinct symbol taxonomies that must be rigorously adhered to for successful data queries.

This document serves as a definitive reference for engineering robust data pipelines for Phemex. It dissects the endpoint topology, mathematically defines the scaling transformations required for price and value data, elucidates the non-standard schema of the candlestick payloads, and provides architectural patterns for the long-term storage of historical market data.

## 2. API Endpoint Topology and Connectivity Standards

Understanding the physical and logical routing of requests is the first step in establishing a stable connection to Phemex’s market data repositories. The exchange creates a clear separation between public market data distribution and private account operations, as well as distinguishing between standard access and high-frequency (VIP) access tiers.

### 2.1 Base Endpoint Architecture

The Phemex API ecosystem is accessible through several base Uniform Resource Locators (URLs), each serving a distinct operational profile. The selection of the correct base endpoint is critical not only for successful authorization but also for optimizing network latency and adhering to the appropriate rate-limiting buckets.

#### 2.1.1 The Primary Public REST Interface

For the vast majority of market data ingestion tasks—specifically the retrieval of historical kline (candlestick) data—the primary entry point is the standard public API.

* **Endpoint URL:** https://api.phemex.com
* **Operational Scope:** This URL routes traffic to the general-purpose API gateway. It supports all public market data queries (tickers, order books, trades, and candlesticks) as well as authenticated order management for standard accounts.
* **Relevance to OHLCV:** This is the default target for scripts fetching historical price actions. It is subject to standard IP-based rate limits.1

#### 2.1.2 The High-Performance (VIP) Interface

Phemex offers a specialized endpoint infrastructure for institutional clients and high-frequency market makers, designated as the "VAPI" (VIP API).

* **Endpoint URL:** https://vapi.phemex.com
* **Access Requirements:** Access to this endpoint is typically restricted. It often requires the client IP address to be whitelisted by Phemex support or the account to hold a specific VIP status.
* **Performance Characteristics:** The VAPI infrastructure is architected to handle higher throughput and provide lower latency responses. For enterprise-grade data ingestion pipelines that require simultaneous monitoring of hundreds of ticker symbols, utilizing the VAPI endpoint is recommended to avoid the throttling inherent in the public gateway. It mirrors the functionality of the primary endpoint but operates with significantly higher rate limit capacities.1

#### 2.1.3 The Testnet Environment

For the development of robust storage routines, particularly when testing the normalization logic for scaled integers without risking capital or polluting production databases, the Testnet is an essential resource.

* **Endpoint URL:** https://testnet-api.phemex.com
* **Data Fidelity:** The Testnet mimics the production environment's behavior, including the specific array structures and scaling factors (Ep/Ev/Er). However, the market data (prices/volumes) is generated by simulation and does not reflect real-world liquidity. It is the ideal sandbox for validating code that parses the 9-element candle array unique to Phemex.1

### 2.2 Protocol Standards and Response Formats

The Phemex API strictly adheres to RESTful standards for its HTTP interface, using standard HTTP verbs (GET, POST, PUT, DELETE) to define operations. A crucial standardization feature is the transport format: all endpoints, without exception, return data encapsulated in JavaScript Object Notation (JSON) objects.

This uniformity simplifies the parsing logic in client applications. Whether the request is for a spot market ticker or a perpetual contract position, the receiving system can anticipate a JSON structure. However, while the *format* is standard JSON, the *schema*—specifically how numbers are represented—deviates from the floating-point standards used by many other exchanges (e.g., Binance or Coinbase), utilizing a string-based or integer-based representation to preserve precision.5

### 2.3 Rate Limiting and Optimization Strategies

A common point of failure in historical data backfilling is the triggering of HTTP 429 (Too Many Requests) errors. Phemex employs a sophisticated, weight-based rate limiting system rather than a simple request-counter.

* **Weight-Based Consumption:** Not all API calls are equal. A simple check of server time might cost 1 weight unit, whereas a query for historical candlesticks—which requires the matching engine to aggregate thousands of trade records from cold storage—costs significantly more.
* **Kline Endpoint Weight:** The specific endpoint used for OHLCV data, /exchange/public/md/kline (and its V2 counterpart), is assigned a weight of **10** per request.
* **Capacity Limits:** Public API groups typically possess a capacity refill rate (e.g., 500 units per minute). This implies that a naive loop requesting candles for 50 symbols sequentially will rapidly exhaust the capacity (50 requests \* 10 weight = 500), leading to a temporary IP ban.6

**Strategic Implication:** Data ingestion pipelines must implement a "token bucket" or "leaky bucket" algorithm that tracks the cumulative weight of outgoing requests, pausing execution when the threshold is neared, rather than relying solely on the HTTP header X-RateLimit-Remaining responses which arrive only *after* a request is processed.

## 3. Symbol Taxonomy and Asset Classification

One of the most frequent sources of error in Phemex data integration—manifesting as empty datasets or "Invalid Symbol" responses—is the incorrect specification of asset symbols. Unlike exchanges that utilize a unified naming convention (e.g., BTC-USDT for all markets), Phemex enforces a strict taxonomic divergence between Spot and Perpetual markets. This section details the distinct nomenclature required for each.

### 3.1 Spot Market Nomenclature: The 's' Prefix

The Spot market, which facilitates the immediate settlement of assets (e.g., exchanging USDT for BTC), utilizes a naming convention that is unique to Phemex and often overlooked by developers migrating from other platforms.

* **The Syntax:** All spot market symbols must be prefixed with a lowercase **s**.
* **Structure:** s + BaseAsset + QuoteAsset
* **Example:** The symbol for Bitcoin trading against Tether is **sBTCUSDT**.
* **Validation:** Attempting to query the spot endpoint with BTCUSDT (omitting the s) will result in a failure. The API treats BTCUSDT as a potential contract symbol, not a spot symbol. This prefix effectively namespaces the symbol within the exchange's internal routing engine, directing the query to the spot matching engine rather than the derivatives engine.7

**Implementation Note:** When designing a database schema, it is common practice to normalize symbols to a generic format (e.g., BTC/USDT). However, the ingestion adapter must contain a mapping layer that explicitly prepends the s before dispatching requests to Phemex.

### 3.2 Perpetual Contract Nomenclature

The derivatives market is further subdivided into Inverse Contracts (Coin-Margined) and Linear Contracts (USD/USDT-Margined). Each sub-class possesses specific naming rules that relate to how the contract is valued and settled.

#### 3.2.1 Inverse Perpetual Contracts

Inverse contracts are quoted in USD but settled in the base cryptocurrency (e.g., BTC). These are the legacy products of the crypto derivatives space.

* **Syntax:** BaseAsset + USD
* **Example:** **BTCUSD**.
* **Semantic Meaning:** This symbol represents a contract where the value is calculated in USD, but the PnL and margin are held in Bitcoin. Each contract typically represents 1 USD of value.8

#### 3.2.2 Linear Perpetual Contracts (USDT-Margined)

Linear contracts are both quoted and settled in a stablecoin, typically Tether (USDT). These are increasingly popular due to the intuitive calculation of profit and loss.

* **Syntax:** BaseAsset + USDT
* **Example:** **BTCUSDT**.
* **Differentiation:** Note the absence of the s prefix. BTCUSDT is the Linear Perpetual contract. sBTCUSDT is the Spot market pair. Confusing these two will lead to significant data corruption, as the price action, funding rates, and volume characteristics differ drastically between the spot and derivatives markets.

### 3.3 Dynamic Symbol Validation

Given the potential for delistings, rebrandings, or new listings, hardcoding symbols is a fragile practice. The robust approach to symbol management involves querying the Products endpoint during the initialization phase of the application.

* **Endpoint:** GET /public/products
* **Function:** This endpoint returns a comprehensive metadata list of all active tradable assets.
* **Metadata Extraction:** The response contains not only the valid symbol strings (e.g., sBTCUSDT) but also critical metadata required for parsing the OHLCV data, specifically the scaling factors (priceScale, ratioScale, valueScale). Parsing this endpoint allows the application to dynamically build a valid universe of symbols and their associated processing rules.1

## 4. The OHLCV (Kline) Architecture

The retrieval of historical market data is facilitated through the "Kline" (candlestick) endpoints. Phemex has iterated on this interface, and currently, the V2 endpoint is the standard for reliable data access.

### 4.1 Endpoint Specifications

The retrieval of OHLCV data is stateless and does not require authentication headers, making it accessible via standard HTTP clients (e.g., requests in Python, axios in JavaScript).

* **Primary URL:** https://api.phemex.com/exchange/public/md/v2/kline
* **Method:** GET
* **Parameter Encoding:** Query parameters must be URL-encoded.

### 4.2 Resolution: The Integer Standard

A critical deviation in Phemex’s API compared to others (like Binance or CCXT standards) is the definition of the resolution parameter. While many exchanges accept string literals like 1m, 1h, or 1d, Phemex strictly enforces the use of **integer seconds**.

The failure to map these values correctly is a primary cause of 400 Bad Request errors. The following table maps standard timeframes to the required Phemex integer values 1:

| **Standard Timeframe** | **Phemex Resolution (Seconds)** | **Description** |
| --- | --- | --- |
| **1 Minute** | 60 | High-resolution intraday data. |
| **5 Minutes** | 300 | Standard intraday analysis. |
| **15 Minutes** | 900 | Intraday trend analysis. |
| **30 Minutes** | 1800 | Half-hour intervals. |
| **1 Hour** | 3600 | Hourly data; standard for swing trading. |
| **4 Hours** | 14400 | Major trend identification. |
| **1 Day** | 86400 | Daily close data. |
| **1 Week** | 604800 | Weekly aggregation. |
| **1 Month** | 2592000 | Monthly aggregation (approximate 30 days). |
| **3 Months** | 7776000 | Quarterly data (approximate 90 days). |
| **1 Year** | 31104000 | Yearly data (approximate 360 days). |

**Integration Insight:** Data ingestion scripts must implement a translation layer that accepts human-readable strings (e.g., "1h") and converts them to the corresponding integer (e.g., 3600) before constructing the API request.

### 4.3 Pagination and Limits

The retrieval of extensive historical datasets (backfilling) is constrained by the API's response limits.

* **The Hard Limit:** The maximum number of candlestick records returned in a single HTTP response is **1000**. Requesting a limit parameter greater than 1000 (e.g., 2000) is ignored, and the server defaults to returning 1000 records.
* **Pagination Logic:** To retrieve data exceeding 1000 bars, the client must implement a loop using the from and to timestamps.
  + **Algorithm:**
    1. Define end\_time (current time) and start\_time (historical target).
    2. Request data from start\_time.
    3. Identify the timestamp of the *last* candle in the returned array.
    4. Update start\_time to last\_candle\_timestamp + resolution\_in\_seconds.
    5. Repeat until start\_time >= end\_time.
* **Gap Handling:** Note that Phemex kline data may be sparse; if no trading activity occurs during a specific minute, no record is generated. This contrasts with some exchanges that return "flat" candles (Open=Close). The ingestion logic must account for these temporal gaps, potentially filling them forward if the downstream database requires continuous time series.1

## 5. Decoding the Response Payload: Structure and Schema

The most complex aspect of Phemex data ingestion—and the source of the user's explicit trouble with "storing candle data"—lies in the interpretation of the JSON response payload. Phemex utilizes a non-standard array structure and a specialized numerical encoding system that renders standard parsing libraries ineffective.

### 5.1 The JSON Envelope

The HTTP response body is a JSON object containing a status code and a data payload.

JSON

{  
 "code": 0,  
 "msg": "OK",  
 "data": {  
 "total": -1,  
 "rows": [...] // Array of candle arrays  
 ]  
 }  
}

* **code**: A value of 0 indicates a successful query. Any non-zero value represents an error state (e.g., rate limit exceeded, invalid symbol).
* **data.rows**: This array contains the actual market data. Each element within rows is itself an array representing a single time interval.

### 5.2 The 9-Element Array Structure

In most digital asset exchange APIs, a candlestick is represented by an array of 5 or 6 elements: ``. Phemex, however, returns an array of **9 elements** for its V2 kline endpoint. The specific ordering of these elements is critical; misinterpreting the indices results in data corruption (e.g., storing the "Last Close" as the "Open" price).

The Phemex Kline Schema:

``

The following table details each index, its data type, and its semantic meaning:

| **Index** | **Field Name** | **Data Type** | **Description and Normalization Requirement** |
| --- | --- | --- | --- |
| **0** | timestamp | Integer | The start time of the candle. **Unit:** Seconds (Unix Epoch). Note: Verify magnitude; 10 digits = seconds, 13 digits = ms, 19 digits = ns. V2 REST typically returns seconds. |
| **1** | interval | Integer | The resolution in seconds (e.g., 60). Redundant for storage but useful for verification. |
| **2** | lastClose | Integer (Scaled) | **Critical Trap:** The closing price of the *previous* interval. Many parsers mistakenly map this to "Open". It is distinct from the current Open price. |
| **3** | open | Integer (Scaled) | The opening price of the current interval. **Must be descaled.** |
| **4** | high | Integer (Scaled) | The highest price traded during the interval. **Must be descaled.** |
| **5** | low | Integer (Scaled) | The lowest price traded during the interval. **Must be descaled.** |
| **6** | close | Integer (Scaled) | The closing price of the current interval. **Must be descaled.** |
| **7** | volume | Integer (Scaled) | The volume traded. Unit depends on the market (Base Currency for Spot/Linear, Contracts for Inverse). |
| **8** | turnover | Integer (Scaled) | The quote currency volume (Value). Often used for calculating VWAP. |

**Parsing Strategy:** To correctly map this to a standard OHLCV object, the parser must strictly access indices 0, 3, 4, 5, 6, 7. Indices 1 and 2 should generally be discarded for standard charting purposes, though lastClose (Index 2) can be useful for gap analysis.2

## 6. Numerical Precision: The Scaled Integer System

The second major hurdle in data storage is Phemex’s implementation of **Fixed-Point Arithmetic**. To avoid the floating-point errors inherent in computing (where 0.1 + 0.2!= 0.3), Phemex represents all monetary values as integers. This is known as the "Internal Representation" vs. "Display Representation."

### 6.1 The Mechanics of Scaling (Ep, Ev, Er)

The API documentation refers to these scaling factors with specific suffixes:

* **Ep (Price Scale):** Used for price fields (Open, High, Low, Close, LastClose).
* **Ev (Value Scale):** Used for value fields (Turnover, and sometimes Volume).
* **Er (Ratio Scale):** Used for ratios (Funding Rates, Interest Rates).

The server returns a raw integer which is the product of the real value and a power of 10.

$$\text{RawInteger} = \text{RealValue} \times 10^{\text{ScaleFactor}}$$

### 6.2 Normalization Logic

To store this data in a standard database (SQL DECIMAL or FLOAT columns), the application must reverse this operation upon receipt.

#### 6.2.1 Price Normalization

For the vast majority of Phemex Spot symbols (e.g., sBTCUSDT) and Linear Perpetuals (BTCUSDT), the Price Scale (priceScale) is **8**.

* **Example Response:** 2500000000000
* **Transformation:**  
  $$2500000000000 \div 10^8 = 25000.00$$

*Note:* Legacy Inverse Contracts (BTCUSD) may utilize a scale of **4** ($10^4$). It is imperative to fetch the priceScale attribute from the /public/products endpoint for each symbol dynamically, rather than hardcoding 100,000,000.

#### 6.2.2 Volume and Turnover Normalization

* **Volume (Index 7):**
  + For **Spot** (sBTCUSDT), volume is typically returned in the base unit scaled by the asset's precision.
  + For **Inverse Perps** (BTCUSD), volume represents the number of contracts (where 1 contract = 1 USD). This is typically an integer and may not require descaling, or uses a scale of 0.
  + For **Linear Perps** (BTCUSDT), volume is the amount of base currency (BTC).
* **Turnover (Index 8):** This represents the *value* of the trade in the Quote currency (e.g., USDT). It is scaled by the valueScale (typically 8 for USDT pairs).
  + **Calculation:** TurnoverRaw / 10^8 = TurnoverUSDT.

### 6.3 Risks of Improper Handling

Failure to apply this division results in price data that is orders of magnitude incorrect (e.g., Bitcoin price appearing as 2.5 quadrillion instead of 25,000). This will not only break charting applications but will also cause overflows in standard database columns defined with DECIMAL(18,2) precision.2

## 7. Implementation Guide: Spot vs. Perpetual

While the underlying mechanics of the API are consistent, the interpretation of the data varies significantly between Spot and Perpetual markets.

### 7.1 Spot Market Specifics

* **Symbol Prefix:** Mandatory s (sETHUSDT).
* **Volume Interpretation:** Volume is strictly the quantity of the **Base Asset** moved.
* **Price Scaling:** Universally 10^8 for modern pairs.
* **Usage:** Spot data is essential for calculating the underlying index price but does not include funding rates or open interest data.

### 7.2 Perpetual Contract Specifics

* **Symbol Suffix:** USD (Inverse) or USDT (Linear).
* **Volume Interpretation:**
  + **Inverse:** Volume = Nominal Value in USD (Contracts).
  + **Linear:** Volume = Quantity of Base Asset (e.g., BTC).
* **Additional Data:** While the Kline endpoint provides OHLCV, perpetual traders often require Funding Rate history. This is *not* included in the standard OHLCV payload and must be queried via a separate endpoint (/md/funding/history), though it shares the same scaled integer logic.

## 8. Common Troubleshooting and Error Handling

### 8.1 Error 400: Bad Request

* **Symptom:** The server rejects the request immediately.
* **Root Cause:** Most often caused by sending the resolution as a string "1m" instead of the integer 60, or using an invalid symbol format (missing s prefix).

### 8.2 Error 39999 or "Invalid Symbol"

* **Symptom:** Response contains a code indicating the symbol does not exist.
* **Root Cause:** Confusion between Spot and Linear identifiers. Querying sBTCUSDT on a contract-specific endpoint, or BTCUSDT on a spot endpoint.

### 8.3 Empty Data Rows

* **Symptom:** data.rows is an empty list ``.
* **Root Cause:**
  1. **Timestamp Precision Mismatch:** Sending milliseconds (13 digits) to an endpoint expecting seconds (10 digits). The server interprets the millisecond timestamp as a date far in the future, finding no data.
  2. **No Trading Activity:** Phemex does not generate candles for intervals with zero volume. The ingestion script must handle gaps logically.12

### 8.4 CCXT Library Specifics

Many developers use the CCXT library for abstraction. However, users have reported issues where CCXT's fetch\_ohlcv implementation for Phemex may inadvertently default to public endpoints even when private keys are supplied, or fail to handle the V2 authentication correctly in older versions.

* **Workaround:** Ensure the library is updated to the latest version. If issues persist, it is often more reliable to implement a direct REST wrapper for the md/v2/kline endpoint using the native requests library in Python, as this grants full control over the parameter types and scaling logic.14

## 9. Conclusion

The successful storage of Phemex market data requires a departure from generic API integration assumptions. The "trouble" users face is rarely a system failure but rather a misalignment between standard expectations (floating-point JSON, 6-element arrays) and Phemex's optimized architecture (scaled integers, 9-element arrays).

By implementing a normalization layer that strictly adheres to the **9-element array index map** (specifically targeting indices 0, 3, 4, 5, 6, 7), rigorously applies the **Symbol Prefix** rules (s for spot), and mathematically reverses the **Scaled Integer** representation using dynamically fetched metadata, engineers can construct a high-fidelity historical database. This rigorous approach ensures that the stored data accurately reflects market reality, providing a solid foundation for quantitative analysis and algorithmic trading strategies.

#### Works cited

1. phemex-api-docs/Public-Hedged-Perpetual-API.md at master - GitHub, accessed December 22, 2025, <https://github.com/phemex/phemex-api-docs/blob/master/Public-Hedged-Perpetual-API.md>
2. phemex-api-docs/Public-Spot-API-en.md at master - GitHub, accessed December 22, 2025, <https://github.com/phemex/phemex-api-docs/blob/master/Public-Spot-API-en.md>
3. Phemex Updates API Entry Endpoints, accessed December 22, 2025, <https://phemex.com/announcements/phemex-updates-api-entry-endpoints>
4. Phemex API |Take your business to next level, accessed December 22, 2025, <https://phemex.com/home-api>
5. API Overview - Phemex, accessed December 22, 2025, <https://phemex.com/help-center/api-overview>
6. Phemex API Reference: Overview, accessed December 22, 2025, <https://phemex-docs.github.io/>
7. phemexboy - PyPI, accessed December 22, 2025, <https://pypi.org/project/phemexboy/>
8. Cryptocurrency | Perpetual Contract | Phemex - SignalStack, accessed December 22, 2025, <https://help.signalstack.com/kb/phemex/phemex-crypto-perpetual>
9. Perpetual Futures Introduction | Intro to Crypto Futures | Phemex, accessed December 22, 2025, <https://phemex.com/contract/introduction>
10. Phemex: Limit amount of candles/bars moving averages · Issue #16919 · ccxt/ccxt - GitHub, accessed December 22, 2025, <https://github.com/ccxt/ccxt/issues/16919>
11. Cannot place LIMIT order, getting error "TE\_PRICE\_TOO\_SMALL" #3 - GitHub, accessed December 22, 2025, <https://github.com/phemex/phemex-python-api/issues/3>
12. Overview | MktCap Historical OHLCV All Assets Hour | CoinDesk Cryptocurrency Data API, accessed December 22, 2025, <https://developers.coindesk.com/documentation/data-api/overview_v1_historical_marketcap_all_assets_hours>
13. fetch\_ohlcv returns empty list for 1min timeframe when previous day it worked fine. - GitHub, accessed December 22, 2025, <https://github.com/ccxt/ccxt/issues/11984>
14. [Phemex] fetch\_ohlcv is hardcoded to public API and never uses authentication · Issue #26156 - GitHub, accessed December 22, 2025, <https://github.com/ccxt/ccxt/issues/26156>