CryptoBot Platform

Data System Documentation

Version 1.0 | January 2026

Table of Contents

1. System Overview

The CryptoBot Data System provides a robust foundation for cryptocurrency market data storage, retrieval, and management. It is designed to support backtesting and live trading across multiple assets and timeframes.

Key Capabilities

Capability	Description
Multi-Asset Support	32+ cryptocurrency pairs with 12 core pairs for backtesting
Multi-Timeframe	6 timeframes: 1h, 4h, 12h, 24h, 72h, 168h
Time-Series Optimized	TimescaleDB for efficient OHLCV storage and queries
Format Auto-Detection	Automatic CSV format detection (Kraken, CryptoCompare, generic)
Gap Handling	Intelligent gap detection and forward-fill for missing data
Docker-Based	Containerized infrastructure for consistent deployment

2. Architecture

The data system follows a layered architecture separating concerns for maintainability and flexibility.

Component Stack

Layer	Component	Purpose
Storage	TimescaleDB	Time-series optimized PostgreSQL for OHLCV data
Ingestion	csv_loader.py	Load historical data from CSV files
Access	database.py	Python interface for queries and operations
Infrastructure	Docker Compose	Container orchestration for DB and services
Monitoring	Grafana	Visualization and alerting

Data Flow

1. Raw CSV files (Kraken, CryptoCompare) are ingested via csv_loader.py
2. Data is validated, deduplicated, and stored in TimescaleDB ohlcv table
3. database.py provides query interface for features and backtesting
4. Multi-timeframe loader resamples 1h data to higher timeframes on-the-fly

3. Database Layer (TimescaleDB)

TimescaleDB is a PostgreSQL extension optimized for time-series data. It provides automatic partitioning, compression, and time-based queries.

OHLCV Table Schema

Column	Type	Description
timestamp	TIMESTAMPTZ	Bar timestamp (UTC, primary key with pair)
pair	VARCHAR(20)	Trading pair (e.g., XBTUSD, ETHUSD)
open	NUMERIC	Opening price
high	NUMERIC	Highest price in period
low	NUMERIC	Lowest price in period
close	NUMERIC	Closing price
volume	NUMERIC	Base asset volume
source	VARCHAR(50)	Data source (kraken, cryptocompare)
volume_quote	NUMERIC	Quote asset volume (optional)

Supported Trading Pairs

Core 12 pairs for backtesting: BTC, ETH, LTC, ETC, XMR, ZEC, XRP, XLM, ADA, LINK, SOL, AVAX

Additional pairs available: DOT, ATOM, UNI, AAVE, MATIC, FIL, ALGO, NEAR, FTM, SAND, MANA, AXS, CRV, COMP, DOGE, SHIB, and more.

4. CSV Data Loader

The csv_loader.py module handles ingestion of historical OHLCV data from various sources with automatic format detection and validation.

Format Auto-Detection

Format	Detection Method	Column Mapping
CryptoCompare	'time' + 'volumefrom' columns	time->timestamp, volumefrom->volume
Kraken	Numeric columns (0-6) or 'trades'	Positional: timestamp,O,H,L,C,volume,trades
Generic	Standard OHLCV column names	Direct mapping with auto timestamp detection

Usage Examples

Command Line:

```
python -m cryptobot.datasources.csv_loader --file /path/to/file.csv --pair XBTUSD
```

Python API:

```
from cryptobot.datasources import load_csv_to_db
load_csv_to_db('path/to/file.csv', 'XBTUSD')
```

Loader Features

- Automatic duplicate detection and removal
- Append mode: only inserts new data after existing range
- Replace mode: deletes existing data before insert
- Chunked inserts (10,000 rows per batch) for performance
- Progress reporting during large imports

5. Database Interface

The Database class in database.py provides a clean Python interface for all data operations.

Core Methods

Method	Description
get_ohlcv(pair, start, end)	Retrieve OHLCV data for a pair within date range
get_latest_price(pair)	Get most recent close price
get_available_pairs()	List all pairs in database
get_data_range(pair)	Get date range and row count for a pair
save_features(df, pair)	Save computed features to features table
get_features(pair, start, end)	Retrieve cached features
record_trade(...)	Record a trade in the trades table
save_backtest_result(...)	Save backtest results with config
summary()	Get summary of all data in database

Usage Example

```
from cryptobot.datasources.database import Database

db = Database()
df = db.get_ohlcv('XBTUSD', start='2020-01-01', end='2024-12-31')
print(f'Loaded {len(df):,} rows')
```

6. Multi-Timeframe Data Loading

The multi-timeframe system enables analysis across 6 timeframes simultaneously, supporting both structural (long-term) and tactical (short-term) regime detection.

Supported Timeframes

Timeframe	Bars per Day	Bars per Week	Use Case
1h	24	168	Execution timing, short-term signals
4h	6	42	Intraday trends
12h	2	14	Session analysis
24h	1	7	Daily regime detection
72h	0.33	2.33	Multi-day trends
168h	0.14	1	Weekly structural regime

Resampling Logic

Higher timeframes are created by resampling 1-hour base data:

- **Open:** First value in period
- **High:** Maximum value in period
- **Low:** Minimum value in period
- **Close:** Last value in period
- **Volume:** Sum of all volumes in period

Alignment

All timeframes are aligned to the 1-hour index using forward-fill. This ensures that at any given 1-hour bar, you have access to the most recent values from all higher timeframes without look-ahead bias.

7. Data Quality & Validation

Gap Handling

The system detects and handles gaps in the data:

- **Detection:** Gaps identified when timestamp difference exceeds expected interval
- **Logging:** Gaps are logged with start, end, and duration
- **Treatment:** Forward-fill applied for small gaps (< 24 hours by default)
- **Large Gaps:** Flagged for review, may indicate exchange downtime

Validation Checks

- Timestamp monotonicity (sorted ascending)
- No duplicate timestamps per pair
- OHLC relationship validity (High >= max(Open,Close), Low <= min(Open,Close))
- Volume non-negative
- Price continuity (no extreme jumps beyond configurable threshold)

Data Summary

Use db.summary() to get an overview of all data in the database, including pair coverage, date ranges, and row counts per source.

8. Configuration

Environment Variables

Variable	Default	Description
DATABASE_URL	postgresql://...	PostgreSQL/TimescaleDB connection string

Docker Configuration

The system uses Docker Compose for orchestration. Key services:

- **timescaledb:** TimescaleDB container with persistent volume
- **grafana:** Monitoring and visualization dashboard

See docker-compose.yaml and timescaledb.yaml for full configuration.