Coinbase Tradebot

⚓ Naval Command Edition — Captain, Skipper, Commodore, Navigator, and Crew Coordination

Overview

Naval Command Edition presents the most stable and feature-rich release of the Coinbase Tradebot. This version refines operational hierarchy — each trading subsystem now carries a naval rank, reflecting its strategic importance and role in coordinated market maneuvers. Together, these officers ensure balanced, adaptive, and profitable operations under all sea conditions.

Edition: Naval Command Series
Version: v1.1.3 (October 2025)
Mode: Live & Dry-run supported (WS or Local aggregation)

TradeBot is a modular, self-regulating trading system for Coinbase Advanced. It operates as a disciplined fleet: each module represents a crew role within a naval hierarchy, ensuring strategy, risk, and maintenance remain in balance.

Core Architecture

1. Load configuration and credentials from APIkeys.env and config.py.
2. Initialize Candle Engine: WS mode for feed or Local mode for ticker aggregation (150 ms settle delay).
3. Seed indicators (EMA, RSI, MACD) with 200 backfill candles.
4. Initialize Advisors and Quartermaster modules.
5. Start WebSocket client and monitor ticker/candles.
6. Run AutoTune to determine regime (uptrend, downtrend, choppy, blend).
7. Monitor Captain (EMA) for crossovers and confirmed signals.
8. Evaluate Quartermaster exits for take-profit or stagnation.
9. Persist all trades to trades.csv, portfolio.json, and fills.json.

Crew Hierarchy and Responsibilities

- **Captain** (EMA): Determines overall market direction using short and long exponential moving averages.
- **Navigator** (Autotune): Adjusts tactical parameters in response to changing seas (market regimes).
- **Commodore** (MACD): Oversees momentum and signal crossovers, confirming the fleet's directional intent.
- **Skipper** (RSI): Monitors overbought/oversold waters to identify favorable entry or exit zones.
- **Quartermaster** (Take-Profit & Stagnation Officer): Manages exits, profit captures, and stale position clearance.
- **Deckhand** (24 h, ±2%): Handles daily maintenance — pruning, log management, and light telemetry duties.
- **Swab**: Keeps the decks clean by pruning processed fills and maintaining state efficiency.
- **Watchdog**: provides vigilance to ensure API connectivity.

Navigator (Autotune) Enhancements

The Navigator steers the fleet based on changing market currents. It fine-tunes each officer's parameters to ensure the Captain, Skipper, and Commodore remain coordinated during volatile or choppy conditions. Key enhancements:

- Adaptive alpha curve for smoother yet more decisive BLEND transitions.
- Per-knob learning rates for nuanced tuning across all officers.
- Quantized, weighted, and bounded adjustments (0.5 bps steps, 2 bps max per vote).
- BLEND gently shifts settings toward the winning regime, while SNAP enforces full preset alignment.
- Optional drift telemetry showing deviation from the golden CHOPPY preset baseline.
- Preserves the CHOPPY regime as the calm-sea anchor baseline.

Navigator Modes of Operation

The Navigator determines operational mode based on regime vote share:

- SNAP: ≥ 70% vote share — immediately adopts the winning regime preset.
- BLEND: 55–69% vote share — gradually aligns tactical knobs toward the winner using smoothing and quantization.
- CHOPPY: < 55% vote share — maintains the golden CHOPPY preset without deviation.

Parameters Adjusted by the Navigator (Autotune)

The Navigator dynamically adjusts the following operational knobs:

- ema_deadband_bps — Captain's sensitivity to short/long EMA convergence.
- rsi_buy_max / rsi_sell_min — Skipper's buy/sell confirmation thresholds.
- macd_buy_min / macd_sell_max — Commodore's trend validation parameters.
- confirm_candles — Number of confirmations before acting on signals.
- per_product_cooldown_s — Time buffer before the next maneuver in a given pair.

EMA periods (Captain's short_ema=40, long_ema=120) remain fixed unless optional tuning is enabled (autotune_tune_ema_periods=True in config.py).

Reconcile Cadence

Reconcile synchronizes past fills, positions, and P&L. At startup, a full reconcile runs before Autotune for accurate telemetry. During the session, automatic sweep reconcilers run every 60 minutes by default (configurable). When enabled, SELL operations perform a quick validation reconcile just before placing the order.

Key Settings (config.py)

Setting	Default (example)	Purpose
dry_run	True	Simulation mode (no live orders).
autotune_enabled	True	Enable Autotune at startup.
autotune_lookback_hours	18	Candle window hours for regime voting.
autotune_vote_interval	15m	Dedicated timeframe for regime voting.
autotune_vote_min_candles	72	Minimum number of vote candles (~18h at 15m).
lookback_hours	48	Reconcile horizon for fills and KPI sync.
mid_reconcile_enabled	True	Enables periodic mid-session reconcile sweeps.
mid_reconcile_interval_minutes	90	Time interval for mid-session reconcile (minutes).
autotune_preview_only	True	Preview tuning suggestions without applying changes.
autotune_elapsed_refresh_hours	4	Hours between optional elapsed Autotune refreshes.

Telemetry and Logbook Entries

The Commodore's telemetry records each decision in real-time — including regime votes, parameter changes, and fleet readiness. An optional drift summary shows how far the ship's settings have deviated from the golden CHOPPY baseline:

[AUTOTUNE DRIFT] ema_deadband_bps:-0.50, rsi_buy_max:+1.00, ...

Quartermaster Module (Take-Profit & Stagnation Officer)

The Quartermaster safeguards profits and eliminates idle trades.

• Take-Profit (10%): Automatically executes a market SELL when unrealized profit reaches 10% (≈ 1000 bps) or higher. This ensures strong moves are captured without relying solely on EMA crossovers.

• Stagnation (36 h, ±2%): Detects positions held for more than 36 hours with less than ±2% movement and a flat MACD histogram. When triggered, the bot performs a stagnation exit to recycle capital into more active opportunities.

Quartermaster logic runs before EMA-based signals each candle close. It will not interfere with active EMA trades and respects cooldown timers to prevent rapid re-entry.

All Quartermaster actions are logged to trades.csv with the exit_reason field set to 'take_profit' or 'stagnation'. In dry_run=True, no trades are written — only simulated.

Quartermaster Settings (config.py)

Setting	Default (example)	Purpose
enable_quartermaster	True	Enables take-profit and stagnation logic.
take_profit_bps	1000	Trigger threshold (10%) for Quartermaster take-profit exits.
max_hold_hours	36	Hours before stagnation check activates.
stagnation_close_bps	200	±2% band for stagnant positions.
flat_macd_abs_max	0.40	Defines 'flat' MACD for stagnation detection.

Technical Parameters

Parameter	Default	Description
short_ema	40	Short-term moving average
long_ema	120	Long-term moving average
confirm_candles	3	Required consecutive confirmations
ema_deadband_bps	6.0	Prevents flapping between cross states
take_profit_bps	800	Quartermaster profit exit threshold (~8%)
daily_spend_cap	120.0	Daily BUY limit to prevent overtrading
local_close_settle_ms	120	Local mode candle close delay

Operational Summary

Module	Role	Key Function
EMA	Captain	Primary trade signal (crossover logic)
Autotune	Navigator	Regime detection & adaptive tuning
MACD	Commodore	Confirms or vetoes Captain's decision
RSI	Skipper	Overbought/oversold protection
Quartermaster	Officer	Take-Profit & stagnation exits
Deckhand	Support	Capital rotation & cleanup
Swab	Maintenance	File hygiene & log prunning
Watchdog	Connectivity Officer	Monitors connection, pings, reconnects, keeps comms alive

Threads in This Program

TradeBot runs a compact, multi-threaded **command loop** designed for stability and low-latency market reaction. Each thread serves a specific operational rank within the fleet, ensuring resilience and coordination across all subsystems:

MainThread — Oversees the startup sequence *(Reconcile → AutoTune → WebSocket)* and manages session-wide state, acting as the ship's **command bridge**.
Responsible for initializing configuration, spawning all other threads, and ensuring orderly startup and shutdown.

WebSocket Thread — Handles the **Coinbase Advanced Trade data streams**.
Continuously processes ticker and candle updates, executes signal evaluations, manages Quartermaster exits, and performs inline connectivity checks.
Feeds real-time data to the **Captain (EMA)**, **Skipper (RSI)**, and **Commodore (MACD)**.

Watchdog Thread (ws-watchdog) — Operates independently of the WebSocket thread as the fleet's **safety officer**.
Continuously monitors connection health, sends pings, reissues subscriptions, and initiates **auto-reconnects or LOCAL mode fallbacks** if the WebSocket becomes stale.

Mid-Session Reconcile Threads — A two-tier system that keeps the portfolio and realized P&L synchronized with Coinbase:

- Scheduler Thread (mid_reconcile) — Runs continuously in the background, scheduling reconciliation sweeps (typically every 60 minutes). Acts as the Quartermaster's clock, ensuring timely audits.
- Worker Threads (reconcile-<epoch>) — Spawned by the scheduler to perform each reconciliation sweep. Fetch recent fills, update portfolio state, and write trade records, then gracefully exit once complete.

Elapsed AutoTune Thread (autotune-elapsed) — Executes a **one-shot regime analysis** after a defined runtime window *(default ≈ 4 hours)*.
Re-evaluates market conditions and adjusts indicator confirmation thresholds *(e.g., RSI/MACD weights)* to maintain optimal responsiveness.
Once completed, it concludes its voyage until the next bot restart.

Fleet Stability Notes

Optimizations to minimize I/O turbulence, avoid redundant pruning operations, and improve the Swab's cleaning routines. ProcessedFills management has been refactored to ensure smooth sailing, and synchronization routines run efficiently without clogging the bilge (disk).

Run Settings

Example configurations for running Tradebot in different modes (these are set in config.py):

```
# Dry run with AutoTune preview:
dry_run = True
autotune_enabled = True
autotune_preview_only = True

# Live run with active AutoTune:
dry_run = False
autotune_enabled = True
autotune_preview_only = False

# Static (no AutoTune. Uses Golden Preset Chop settings):
autotune_enabled = False
```

You can now also override some settings directly in the CLI:

```
# Paper trade without live orders

python main.py --dry-run=true



# Disable Quartermaster exits

python main.py --enable-quartermaster=false



# Adjust per-order size and daily cap

python main.py --usd-per-order=30 --max-spend-cap=300
```

Risk Controls & Safety Gates

- Daily BUY Cap: Stops BUYs after cap is reached.

- Quartermaster Exits: Enforced via MARKET_ONLY for deterministic execution.

- Live balance fallback: Prevents portfolio desync on restart.

- Unified state locks: Ensure CSV and portfolio writes are atomic.

- Hard-stop protection: Emergency SELL if price drops >3% from cost basis.

Technical Summary

- **Execution path:** ✅ EMA captain → advisor veto → maker orders by default; Quartermaster runs **before** EMA and sells **at market** for take-profit/stagnation (with cooldown + dust guard).
- **Loss minimization:** ✅ Optional hard_stop_bps; advisors veto bad crosses; per-product cooldown; daily buy cap; maker pricing with offsets to avoid taker fees on routine trades.
- **Spam loops:** ✅ Quartermaster throttled, clamped to held size, and won't fire again within a short window.
- **State safety:** ✅ Fill reconciliation before AutoTune; immediate fill handling under a lock; CSV logging gated off in dry-run.