F L O F M A T R I X

Fractal Liquidity & Order Flow Trading System

COMPREHENSIVE ENGINEERING SPECIFICATION

Module Architecture | Function Signatures | Data Flow | Configuration

The Complete Build Reference for NautilusTrader + DataBento Implementation

Version:	**1.0 (Initial Release)**
Date:	**February 2025**
Target Stack:	**Python 3.11+ / NautilusTrader / DataBento**
Classification:	**CONFIDENTIAL**

1. Executive Summary

The FLOF Matrix (Fractal Liquidity & Order Flow) is an institutional-grade algorithmic trading system designed to trade ES futures and cryptocurrency using a multi-timeframe Smart Money Concepts (SMC) framework combined with deep Order Flow analysis. The system is executed via NautilusTrader (event-driven backtesting and live trading engine) with DataBento providing market data for ES and free exchange WebSockets for crypto.

The core thesis is that price moves from liquidity pool to liquidity pool, and that institutional order flow leaves detectable footprints on the tape (Time & Sales) and in the order book (Level 2) before price moves. The system identifies these footprints at pre-mapped structural levels and executes with defined risk using a confluence-graded entry system.

1.1 Core Design Principles

PRINCIPLE	IMPLEMENTATION
Event-Driven Efficiency	The Predator State Machine minimizes compute by only subscribing to expensive data feeds (T&S, L2) when price is near a point of interest. Three states (Scouting, Stalking, Kill Mode) scale data consumption to market proximity.
Multi-Timeframe Fractal Structure	Monthly/Weekly charts establish macro bias. Daily/4H charts determine directional trend. 15m/1H charts map Points of Interest (POIs). 1m chart is the execution timeframe for CHOCH confirmation. 2m chart runs Velez moving average calculations.
Order Flow as Primary Trigger	The system does not enter on structure alone. Every entry requires Order Flow confirmation: CVD divergence, footprint imbalances, absorption detection, and whale block trade filtering. Structure tells WHERE to look; Order Flow tells WHEN to pull the trigger.
Confluence-Graded Sizing	A 14-point rubric scores every trade across 10 criteria (6 core SMC+OF, 4 Velez momentum layers). The score determines position size: A+ (full risk), A (standard), B (half), C (no trade). This ensures capital is concentrated on the highest-probability setups.
Defense-in-Depth Risk Management	Four layers of protection: Exchange-native OCO brackets (survives bot death), RiskOverlord circuit breakers (rate limiting, position limits, drawdown caps), Sudden Move Classifier (news shield, cascade engagement, infrastructure shutdown), and Volume Profile stop placement (HVN/LVN Moat system).
Feature Toggle Architecture	30 independently toggleable features (T01–T30) with dependency enforcement and safety locks. Every non-trivial feature can be isolated for backtesting to measure its contribution to system edge.

1.2 System at a Glance

ATTRIBUTE	SPECIFICATION
Markets	ES (E-mini S&P 500 Futures) and Crypto (BTC/ETH on Binance/Bybit)
Trading Style	Intraday scalps and swings within NY Killzones (ES: 9:30–11:30 AM, 1:30–3:30 PM EST) and London/NY overlap (crypto)

Execution Engine	NautilusTrader (Python, event-driven, supports backtesting and live execution with the same codebase)
Market Data (ES)	DataBento — schema shifting between ohlcv-1m, trades, tbbo, and mbp-10 based on Predator State
Market Data (Crypto)	Free exchange WebSocket feeds (Binance/Bybit). Dual-exchange redundancy for failover.
Risk Per Trade	0.5% to 2.0% of account depending on confluence grade. Max 3 consecutive losses or -2% daily drawdown triggers Nuclear Flatten.
Confluence Rubric	14-point scale: 10 core SMC+OF points + 4 Velez momentum points. A+ ≥ 12, A = 11, B = 9–10, C < 9 (no trade). Tier 1 gate: core must score ≥ 7.
Feature Toggles	30 toggles (T01–T30) across 5 layers: Structure, Execution, Velez, Risk, Safety. Config via TOML file. Safety toggles locked in live mode.
Configuration Format	TOML with sections for toggles and constants. Hot-reloadable in paper trading. Requires restart in live.

2. Module Architecture Overview

The system is organized into 12 modules, each with a single responsibility. Modules communicate through NautilusTrader's event bus (publish/subscribe pattern). No module directly calls another module's internal methods — all inter-module communication is through events and shared state objects.

#	MODULE	LAYER	PRIMARY TOGGLES	RESPONSIBILITY
M01	**DataIngestor**	Data	T28	Manages DataBento subscriptions, exchange WebSockets, and the Ring Buffer. Handles schema shifting based on Predator State.
M02	**InfraHealth**	Data	T28	Monitors DataBento latency, broker API health, exchange status, and heartbeat. Publishes health events.
M03	**HTFStructureMapper**	Structure	T01, T02, T03	Maps macro structure (Daily/4H BOS), calculates directional bias, runs Regime Filter, generates Synthetic MA POIs.
M04	**POIMapper**	Structure	T04, T05	Maps 15m/1H Points of Interest: Order Blocks, FVGs, session highs/lows, PDH/PDL. Tracks freshness (mitigated vs. unmitigated).
M05	**EventCalendar**	Structure	T29	Maintains scheduled economic events. Publishes pre-event warnings at T-5 minutes. Feeds the MoveClassifier.
M06	**PredatorStateMachine**	State	T06, T10, T11	Three-state machine (Scouting → Stalking → Kill). Controls data subscription levels. Manages Proximity Halo and Tape Velocity triggers.
M07	**OrderFlowEngine**	Execution	T07, T08, T09	Processes T&S and L2 data during Kill Mode. Calculates CVD, footprint imbalances, absorption detection, whale block filtering.
M08	**VelezMAModule**	Scoring	T12–T16	Calculates 2m 20 SMA and 200 SMA. Evaluates 4 Velez confluence criteria: 20 SMA Halt, Flat 200 SMA, Elephant Bar, Micro-Trend. Also provides RBI/GBI Hold Filter for runners.
M09	**ConfluenceScorer**	Scoring	All T01–T16	Aggregates all criteria into a single score. Applies Tier 1 gate check. Assigns grade (A+/A/B/C). Determines position size. Applies cascade override.
M10	**ExecutionManager**	Execution	T17, T24	Executes entries (limit or market based on tape speed). Places HVN/LVN stops. Sends OCO brackets to exchange. Manages order lifecycle.
M11	**TradeManager**	Management	T18–T23	Manages open positions through 3 phases: Fixed Partial, Structural Node Trail (with RBI/GBI + 20 SMA health check), and Dynamic Climax Exit (with 200 SMA watch zone).

M12	**RiskOverlord**	Safety	T24–T30	Circuit breakers, rate limiting, position limits, drawdown caps, stale data detection, Sudden Move Classifier, and Nuclear Flatten sequence. Wraps around all other modules.

3. Data Layer (M01: DataIngestor, M02: InfraHealth)

The data layer is the foundation of the entire system. Every analytical decision depends on the quality, timeliness, and cost-efficiency of the data flowing into the system. The data layer is designed to be adaptive — it subscribes to exactly the data it needs at each moment, nothing more.

3.1 Module M01: DataIngestor

The DataIngestor manages all external data connections and provides a unified internal data interface. It receives commands from the PredatorStateMachine about which data schemas to subscribe to, and it maintains the Ring Buffer for instant Order Flow context.

3.1.1 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
subscribe_schema()	instrument: InstrumentId schema: str (ohlcv-1m | trades | tbbo | mbp-10)	Subscription handle or raises DataError	Subscribes to a DataBento schema for the given instrument. Called by PredatorStateMachine when transitioning states. Scouting uses ohlcv-1m (cheapest). Stalking adds trades + tbbo. Kill Mode adds mbp-10. Each upgrade is additive — previous schemas remain active.	T28
unsubscribe_schema()	instrument: InstrumentId schema: str	bool (success)	Drops a DataBento schema subscription. Called when downgrading state (Kill → Stalking) or during time-based severing (ES disconnect 4PM–8:30AM). Reduces ongoing DataBento costs.	
connect_exchange_ws()	exchange: str (binance | bybit) symbols: list[str]	WebSocket connection handle	Opens a free exchange WebSocket for crypto data. Maintains dual-exchange connections for redundancy. If primary fails, secondary is already connected. Never uses DataBento for crypto (cost optimization).	
push_to_ring_buffer()	tick: TradeEvent | QuoteEvent	None	Pushes incoming tick data into the 60-second circular queue. Oldest tick is overwritten when buffer is full. Always running during active Killzones. Provides instant Order Flow context when price suddenly wicks into a POI.	
get_ring_buffer()	lookback_seconds: float = 60.0	list[TradeEvent]	Returns all ticks in the Ring Buffer within the specified	

			lookback window. Called by OrderFlowEngine when Kill Mode activates. Returns a snapshot (copy), not a reference, to prevent race conditions.	
sever_connection()	instrument: InstrumentId	None	Completely disconnects from DataBento for the given instrument. Used for time-based severing: ES disconnects 4PM–8:30AM EST to save 16+ hours of data costs daily. Reconnects at Killzone open.	
get_ohlcv()	instrument: InstrumentId timeframe: str count: int	list[Bar]	Returns the most recent N OHLCV bars for the given timeframe. Used by all analytical modules. Cached internally to avoid repeated DataBento queries.	

3.1.2 Ring Buffer Architecture

The Ring Buffer is a fixed-size circular queue that stores the last 60 seconds of raw tick data. It is the key to the system's ability to react instantly to violent price moves. When price wicks into a POI without warning, the Ring Buffer already contains 60 seconds of Order Flow history, allowing Kill Mode to evaluate tape conditions immediately without waiting to accumulate data.

PROPERTY	VALUE
Data Structure	collections.deque with maxlen (Python) or equivalent fixed-size circular buffer. When full, oldest element is automatically evicted on new push.
Capacity	60 seconds of ticks. For ES during active hours, this is approximately 500–2000 ticks. For crypto, 100–500 ticks. Memory footprint: < 5MB.
CPU Cost	Near-zero. One deque.append() per tick. No sorting, no searching, no aggregation on insert.
Always Active	YES during Killzones. The Ring Buffer receives ticks regardless of Predator State. Even in Scouting Mode, the trades feed is maintained just for the Ring Buffer (via the heartbeat subscription).
Thread Safety	The Ring Buffer is written to by the DataIngestor's event handler and read by the OrderFlowEngine. Use a threading.Lock or NautilusTrader's built-in message passing to prevent concurrent access.

3.2 Module M02: InfraHealth

The InfraHealth module continuously monitors the health of all data connections and broker APIs. It is the primary trigger for Type C (Infrastructure Degradation) events in the Sudden Move Classifier. If any health metric degrades beyond its threshold, InfraHealth publishes a health alarm that RiskOverlord can act on.

3.2.1 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
check_databento_latency()	exchange_ts: datetime local_ts: datetime	latency_ms: float is_stale: bool	Compares DataBento's exchange-origin timestamp to the local clock. If latency > 500ms, data is stale. Publishes STALE_DATA event. If latency persists > 10 seconds, RiskOverlord initiates Nuclear Flatten.	T28
check_broker_api()	None (polls broker)	latency_ms: float status_code: int	Pings the broker API endpoint. If latency > 400ms or any 5xx error returned, publishes BROKER_DEGRADED event. The bot can still receive data but cannot reliably place orders.	T28
check_exchange_status()	exchange: str	status: str (normal | degraded | halted)	Queries the exchange's system status API. CME publishes circuit breaker states. Binance/Bybit publish system status. Any non-normal status triggers a Type C classification.	T28
check_heartbeat()	last_tick_ts: datetime	seconds_since: float is_dead: bool	Monitors time since last received tick. If > 5 seconds during ES RTH or > 3 seconds on crypto WebSocket, the feed is considered dead. Publishes HEARTBEAT_TIMEOUT event.	T28
publish_health_report()	None	HealthReport object	Aggregates all health checks into a single report object published every 1 second. Contains: databento_latency_ms, broker_latency_ms, exchange_status, seconds_since_tick, all_green: bool. Consumed by RiskOverlord and MoveClassifier.	T28

4. Structure Layer (M03–M05)

The Structure Layer establishes the market context for every trading session. It answers three questions: What is the macro directional bias? Where are the key structural levels? Are there any scheduled events that could create sudden moves?

4.1 Module M03: HTFStructureMapper

The HTFStructureMapper runs on a scheduled cycle (Daily/4H evaluation once per session open, Weekly evaluation Monday pre-session, Monthly evaluation first trading day of month). It is the most computationally expensive module during its evaluation cycle, but it runs infrequently and caches results.

4.1.1 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
evaluate_macro_bias()	daily_bars: list[Bar] h4_bars: list[Bar]	MacroBias (BULLISH | BEARISH | NEUTRAL) list[SwingPoint]	Identifies BOS (Break of Structure) on Daily and 4H charts. A bullish BOS (higher high breaking previous swing high) sets bias to BULLISH. Evaluates swing highs and lows using fractal detection (3-bar or 5-bar pivot). The bias is the foundation for the +2 Trend Alignment criterion.	T01
calculate_regime()	current_price: float weekly_200sma: float monthly_200sma: float macro_bias: MacroBias	RegimeState trend_alignment_weight: int (1 or 2)	The Macro Regime Filter. Compares current price to Weekly and Monthly 200 SMAs. If price is above both and bias is bullish, returns full weight (+2). If price is below Weekly 200 SMA but bias is bullish, returns reduced weight (+1). This modification directly adjusts the Confluence Rubric's most important criterion.	T02
generate_synthetic_poi()	weekly_200sma: float monthly_200sma: float existing_pois: list[POI] daily_atr: float	list[SyntheticPOI] or empty	Checks whether each HTF MA falls within any existing POI. If not, creates a Synthetic MA POI centered on the MA value with width = MA ± (1.5 × Daily ATR). Synthetic POIs are	T03

			capped at B grade max and require the 20 SMA Halt confluence for entry. This catches the “drifting equilibrium” scenario where a universally watched MA sits in empty structural space.	
compute_weekly_200sma()	weekly_closes: list[float] (200+)	sma_value: float	Simple arithmetic: sum of last 200 weekly closes / 200. Cached until next Monday. The Weekly 200 SMA is the single most important level for institutional portfolio allocation decisions.	T02
compute_monthly_200sma()	monthly_closes: list[float] (200+)	sma_value: float	Same calculation on monthly data. Updated first trading day of month. Represents ~16 years of price history. Rarely tested but creates massive reactions when approached.	T02
identify_htf_order_blocks()	bars: list[Bar] timeframe: str	list[OrderBlock]	Identifies institutional Order Blocks on the given timeframe: the last opposite-color candle before a displacement move. An OB is valid if it precedes a BOS and has not been fully mitigated. Returns OB with high, low, and midpoint.	T01
identify_htf_fvgs()	bars: list[Bar] timeframe: str	list[FVG]	Identifies Fair Value Gaps: three-candle formations where candle 1’s high is below candle 3’s low (bearish FVG) or candle 1’s low is above candle 3’s high (bullish FVG). These represent institutional imbalances.	T01

4.2 Module M04: POIMapper

The POIMapper operates on the 15-minute and 1-hour timeframes, mapping the specific levels where the bot will look for trade entries. It runs continuously during active sessions, updating as new structure forms.

4.2.1 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
map_order_blocks()	bars_15m: list[Bar]	list[POI] typed ORDER_BLOCK	Same algorithm as HTF OB identification but on the 15m chart. These are the primary entry zones. An OB is the candle body (open to close) of the last opposite-color candle before a displacement move.	T01
map_fvgs()	bars_15m: list[Bar]	list[POI] typed FVG	15-minute FVG detection. FVGs represent price imbalances where orders were not fully matched. Price tends to return to fill these gaps, making them valid entry zones.	T01
map_session_levels()	bars: list[Bar]	dict with PDH, PDL, session_high, session_low, asian_high, asian_low	Calculates Previous Day High/Low, current session high/low, Asian session range, and Initial Balance (first 30 min of RTH for ES). These are the major liquidity pools that the system watches for sweeps.	T05
detect_liquidity_sweep()	current_price: float session_levels: dict recent_bars: list[Bar]	SweepEvent or None	Detects when price has just taken out a major liquidity level and reversed. Example: price spikes above PDH, triggers stops above, then closes back below. This sweep creates the setup condition for the +2 Major Liquidity Sweep criterion.	T05
track_freshness()	poi: POI current_price: float	bool (is_fresh)	Marks POIs as mitigated when price enters the zone. First-touch POIs earn the +1 Fresh POI bonus. Once mitigated, a POI keeps its boundaries but loses the freshness bonus on any subsequent visit.	T04
calculate_proximity_halo()	poi: POI atr_1m: float multiplier: float = 1.5	halo_high: float halo_low: float	Calculates the distance-based trigger zone around each POI. When price enters this zone, the PredatorStateMachine transitions from Scouting to Stalking. The halo expands and contracts with volatility (ATR-based), providing an adaptive early warning system.	

4.3 Module M05: EventCalendar

A lightweight module that maintains a static JSON file of scheduled economic events and publishes pre-event warnings.

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
load_events()	filepath: str	list[ScheduledEvent]	Parses the events JSON file at session start. Each event has: name, datetime, impact_level (high/medium/low), and affected_instruments. Updated weekly by manual edit.	T29
check_upcoming()	current_time: datetime window_minutes: float = 5.0	ScheduledEvent or None	Checks if any high-impact event is within the specified window. If yes, publishes PRE_EVENT warning. The MoveClassifier uses this to determine Type A classification.	T29
is_in_post_event()	current_time: datetime cooldown_minutes: float = 3.0	bool	Returns true if currently within the post-event cooldown window. During this period, new entries are blocked while the system re-evaluates HTF bias and recalculates POIs that may have been invalidated by the news move.	T29

5. State Machine Layer (M06: PredatorStateMachine)

The PredatorStateMachine is the central controller that determines the bot's operational intensity. It is named "Predator" because it mimics a hunting cycle: the bot conserves energy (data costs, compute) when there is nothing to do, and focuses maximum resources only when a kill opportunity is imminent.

5.1 State Definitions

STATE	DATA SUBSCRIPTIONS	COMPUTE LEVEL	WHAT THE BOT IS DOING
SCOUTING	15m/1H OHLCV only	**LOW**	Mapping POIs, calculating Proximity Halos, monitoring session levels. Waiting for price to approach a POI. Data costs: pennies/day.
STALKING	+ 1m OHLCV	**MEDIUM**	Price is inside a Proximity Halo. Watching for 1m CHOCH/BOS formation inside the HTF POI. Also triggered by Tape Velocity spike (300% above 1hr average).
KILL MODE	+ trades (T&S) + mbp-10 (L2)	**MAXIMUM**	Price has tapped the POI and 1m CHOCH is beginning. Full Order Flow analysis: CVD calculation, footprint imbalances, absorption detection, whale block filtering. This is the most expensive state.

5.2 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
evaluate_state()	current_price: float active_pois: list[POI] tape_velocity: float atr_1m: float	State (SCOUTING | STALKING | KILL)	Core state transition logic. Checks: (1) Is price inside any Proximity Halo? If yes → STALKING. (2) Has Tape Velocity spiked 300%+ above 1hr average? If yes → STALKING (cascade detection). (3) Has price tapped a POI AND 1m CHOCH begun? If yes → KILL. (4) Otherwise → SCOUTING.	T06
transition_to()	new_state: State	None (publishes StateChange event)	Handles state transitions. On SCOUTING→STALKING: instructs DataIngestor to add 1m OHLCV. On STALKING→KILL: instructs DataIngestor to add trades + mbp-10. On downgrade: instructs DataIngestor to drop schemas. Logs every transition with timestamp.	

calculate_tape_velocity()	ring_buffer: list[TradeEvent] window_seconds: float = 5.0	trades_per_second: float	Counts tick frequency over the specified window from the Ring Buffer. Compared against the rolling 1-hour average. If current velocity > 300% of average for 5+ consecutive seconds, returns spike signal. This detects liquidity cascades before they reach the POI.	T11
detect_choch()	bars_1m: list[Bar] poi: POI	CHOCHEvent or None with displacement_ratio: float	Detects 1-minute Change of Character within a POI zone. CHOCH = price makes a higher high (bullish) or lower low (bearish) on the 1m chart after tapping the POI. Displacement is measured as CHOCH candle body / 1m ATR. Must be > 1.5 to qualify. Also checks for FVG creation.	T06
check_killzone()	current_time: datetime instrument: str	bool (in_killzone)	Returns whether current time is within an active Killzone. ES: NY AM 9:30–11:30 or NY PM 1:30–3:30 EST. Crypto: London/NY overlap. Used for the +1 Killzone Timing criterion and for time-based data severing.	T10

6. Order Flow Engine (M07: OrderFlowEngine)

The OrderFlowEngine is the analytical core of the system. It processes raw tick data (Time & Sales and Level 2) during Kill Mode to determine whether institutional money is confirming the structural setup. This is the module that separates the FLOF Matrix from a basic SMC system: structure tells you WHERE to look, but Order Flow tells you WHETHER institutions are actually defending the level.

6.1 Order Flow Concepts

CONCEPT	DEFINITION & SIGNIFICANCE
Cumulative Volume Delta (CVD)	The running total of buy volume minus sell volume. A “buy” is a trade that crosses the ask (aggressive buyer). A “sell” crosses the bid (aggressive seller). CVD divergence at a POI (e.g., price makes a lower low but CVD makes a higher low) signals that selling pressure is exhausting and institutions are accumulating. This is the primary Order Flow confirmation.
Footprint Imbalance	Within each price level of a candle, the ratio of buy volume to sell volume. A 3:1 or greater imbalance in the trade direction, stacked across 3+ consecutive price levels, indicates one-sided institutional flow. Stacked imbalances at a POI are a strong confirmation signal.
Absorption	When massive market sell orders (visible on T&S) hit the bid, but price does not drop (or barely drops). This means institutional limit buy orders are “absorbing” the selling. Detected by comparing T&S volume (market orders) against L2 price movement. This is the ultimate confirmation: someone with very deep pockets is defending the level.
Tape Speed (Pace of Tape)	The frequency of trades per second on the T&S. A sudden spike in tape speed at a POI, followed by abrupt slowdown (“sudden death”), often precedes a reversal. The burst = liquidation cascade hitting the level. The slowdown = selling has exhausted. Combined with CVD shift, this is a high-probability reversal signal.
Whale Block Trades	Trades > 50 contracts (ES) or > 5 BTC on the T&S. These are institutional-scale orders that cannot be disguised. A cluster of whale blocks appearing at or near a POI, especially on the bid (buying), is a strong signal that the level will hold. Filtered from the T&S stream by size threshold.

6.2 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
calculate_cvd()	trades: list[TradeEvent] window_seconds: float	cvd_value: float cvd_series: list[float]	Classifies each trade as buy (price ≥ ask) or sell (price ≤ bid) and calculates the running cumulative delta. Returns both the final CVD value and the time series for divergence detection. The window should match the Ring Buffer lookback (60 seconds default, 30 seconds during cascades).	T07

detect_cvd_divergence()	price_series: list[float] cvd_series: list[float] trade_direction: str	bool (divergence_present) strength: float	Compares price lows/highs against CVD lows/highs. Bullish divergence: price lower low + CVD higher low. Strength is measured as the angle difference between the two series. Divergence alone scores +1 in the Order Flow criterion.	T07
calculate_footprint()	trades: list[TradeEvent] price_levels: list[float] tick_size: float	dict[price_level → {buy_vol, sell_vol, ratio}]	Groups trades by price level (rounded to tick size) and calculates buy/sell volume and ratio at each level. Used to detect stacked imbalances (3:1+ ratio across 3+ levels). The footprint is the standard institutional analysis tool.	T07
detect_stacked_imbalance()	footprint: dict trade_direction: str min_ratio: float = 3.0 min_levels: int = 3	bool imbalance_count: int	Scans the footprint for consecutive price levels where the buy/sell ratio exceeds min_ratio in the trade direction. Three or more stacked levels = stacked imbalance. Combined with CVD divergence, this scores the full +2 in the Order Flow criterion.	T07
detect_absorption()	trades: list[TradeEvent] book_snapshots: list[L2Snapshot] price_change: float	bool (absorption_detected) volume_absorbed: float	The ultimate confirmation. Compares the volume of aggressive market sells (from T&S) against the price displacement (from L2 snapshots). If sell volume > 200% of 1-minute average AND price displacement < 0.25 × expected displacement based on that volume, absorption is occurring. This means institutional limit buyers are silently accumulating at the POI.	T08
filter_whale_blocks()	trades: list[TradeEvent] threshold_es: int = 50 threshold_crypto: float = 5.0	list[TradeEvent] (whale trades only)	Filters the T&S stream for block-size trades. For ES: ≥ 50 contracts. For crypto: ≥ 5 BTC. Clusters of whale blocks at a POI, especially on the defending side (buys at a bullish POI), provide strong A+ grade confirmation.	T09

evaluate_order_flow()	ring_buffer_data: list[TradeEvent] book_data: list[L2Snapshot] poi: POI trade_direction: str	OFScore (0, 1, or 2) OFReport	Master function that orchestrates all OF sub-analyses. Calls CVD, footprint, absorption, and whale detection. Returns a score (0 = no OF confirmation, 1 = CVD divergence only, 2 = full CVD + stacked imbalance confirmation) and a detailed report for logging.	T07

7. Velez Momentum Module (M08: VelezMAModule)

The VelezMAModule is a lightweight analytical module that calculates moving averages on the 2-minute chart and evaluates four additional confluence criteria inspired by Oliver Velez's trading methodology. It also provides the RBI/GBI hold filter for the runner management phase. The entire module adds less than 1 millisecond of compute per evaluation.

7.1 Moving Average Maintenance

The module maintains two simple moving averages on the 2-minute OHLCV data: a 20-period SMA (short-term momentum) and a 200-period SMA (long-term trend). These are updated on every new 2-minute bar close. The 2-minute timeframe was chosen because it is Velez's preferred chart and provides a good balance between noise filtering and responsiveness for intraday trading.

7.2 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
update_smas()	bar_2m: Bar	sma_20: float sma_200: float sma_20_slope: float sma_200_slope: float	Called on every 2-minute bar close. Updates both SMAs and their slopes. Slope is calculated as (current SMA − SMA from N bars ago) / N. SMA 20 slope uses 5-bar lookback. SMA 200 slope uses 50-bar lookback.	T16
check_20sma_halt()	poi: POI sma_20: float atr_1m: float	bool	Returns true if abs(POI midpoint − 2m 20 SMA) ≤ (0.5 × 1m ATR). This means the 20 SMA dynamic support/resistance coincides with the structural POI — two independent reasons to expect a reaction at the same price. Awards +1 confluence point.	T12
check_flat_200sma()	poi: POI sma_200: float sma_200_slope: float atr_1m: float	bool	Returns true if (1) abs(POI midpoint − 2m 200 SMA) ≤ (1.0 × 1m ATR) AND (2) abs(sma_200_slope) < 0.02 points/bar. A flat 200 SMA at a POI is one of the strongest confluence signals because it represents true equilibrium with massive institutional volume. Awards +1 point.	T13
check_elephant_bar()	recent_bars: list[Bar] (last 2) avg_range_10: float avg_volume_10: float trade_direction: str	bool	Checks if either of the last 2 candles qualifies as an Elephant Bar: body ≥ 70% of range, range ≥ 1.3× avg range of prior 10 bars, volume > avg volume of prior 10 bars, and direction matches trade. Provides candle-structure	T14

			confirmation that pairs with OF absorption. Awards +1 point.	
check_20sma_trend()	sma_20_slope: float current_price: float sma_20: float trade_direction: str	bool	For longs: slope > 0 AND price > SMA 20. For shorts: slope < 0 AND price < SMA 20. Confirms that short-term momentum is aligned with the structural entry. Awards +1 point.	T15
check_rbi_gbi()	bar_prev: Bar bar_current: Bar trade_direction: str	bool (pattern_detected)	Red Bar Ignored (long): red candle appears, next bar breaks above red bar's high. Green Bar Ignored (short): green candle, next bar breaks below green bar's low. Used by TradeManager during runner phase to prevent premature exit on single-candle noise.	T20
evaluate_velez_tier()	poi: POI sma_20: float sma_200: float sma_20_slope: float sma_200_slope: float recent_bars: list[Bar] atr_1m: float trade_direction: str	VelezScore (0–4) VelezReport	Master function calling all 4 checks. Returns total Tier 2 score and detailed report. Only called if Tier 1 score ≥ 7 (gate check passed).	T16

8. Confluence Scorer (M09: ConfluenceScorer)

The ConfluenceScorer is the decision-making hub. It aggregates all analytical outputs from M03–M08 into a single trade score, applies the Tier 1 gate check, assigns a grade, and determines position sizing.

8.1 Scoring Flow

The scorer follows this exact sequence, which must be implemented as written:

Step 1: Evaluate all 6 Tier 1 criteria and sum the score (max 10). **Step 2:** GATE CHECK — if Tier 1 < 7, return Grade C immediately (no trade). Do NOT evaluate Tier 2. **Step 3:** If gate passed, evaluate all 4 Tier 2 Velez criteria (max 4). **Step 4:** Total = Tier 1 + Tier 2 (max 14). **Step 5:** Assign grade: A+ (12–14), A (11), B (9–10), C (<9). **Step 6:** Apply Synthetic POI cap (max B) if applicable. **Step 7:** Apply Type B cascade override (50% of graded risk) if MoveClassifier reports cascade. **Step 8:** Return grade, position size in contracts, and full score breakdown.

8.2 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
score_trade()	setup: TradeSetup (contains all analytical outputs)	TradeGrade object: { tier1_score, tier2_score, total, grade, position_size_pct, position_size_contracts, score_breakdown: dict, is_synthetic_poi: bool, is_cascade: bool }	The master scoring function. Called once per Kill Mode evaluation. Produces the final go/no-go decision and position size.	All
calculate_position_size()	grade: str account_equity: float stop_distance_ticks: float tick_value: float is_cascade: bool	size_contracts: int risk_pct: float	Converts the grade into a concrete number of contracts. A+ = 1.5–2.0% risk. A = 1.0–1.5%. B = 0.5–1.0%. Contracts = (equity × risk_pct) / (stop_distance × tick_value). Rounds down to nearest integer. Cascade override halves the result.	

9. Execution Manager (M10: ExecutionManager)

The ExecutionManager handles the physical act of placing orders on the exchange. It makes the entry mode decision (limit vs. market), calculates the HVN/LVN stop level, and constructs the OCO bracket that protects the position even if the bot crashes.

9.1 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
decide_entry_mode()	tape_velocity: float avg_velocity_1hr: float absorption_detected: bool delta_spike: bool	EntryMode (LIMIT | MARKET)	Normal tape speed (velocity < 200% avg): place limit order at 1m FVG for better R:R. Wait for candle close. Extreme tape speed (velocity > 300% avg) with absorption + delta spike: fire market order immediately (Fast Move Switch). The rationale: in a V-shape reversal, waiting for a limit fill means missing the trade entirely.	T11
calculate_stop()	poi: POI recent_bars: list[Bar] atr_1m: float	stop_price: float hvn_level: float lvn_level: float	Builds a micro Volume Profile of the POI leg. Identifies the HVN (where institutions accumulated — the densest volume node) and the LVN (the vacuum below/above it). Stop = bottom edge of LVN − (0.5 × 1m ATR). The LVN acts as a moat: flash wicks are unlikely to penetrate through empty space. The ATR buffer prevents single-tick stop hunts. Fallback (T17 OFF): stop = entry ∓ (2 × 1m ATR).	T17
submit_oco_bracket()	instrument: InstrumentId entry_order: Order stop_price: float tp_price: float	OCOBracketId	Submits the entry order with an exchange-native OCO bracket. CME uses 'Stop with Protection' and 'Market with Protection' order types for better fill quality. The OCO lives on the exchange server, not in the bot. If the bot process dies, the exchange still holds the stop and target. This is the foundation of the entire risk management system.	T24
calculate_tp()	entry_price: float stop_price: float r_multiple: float = 2.0	tp_price: float	Calculates the initial take-profit level for the OCO bracket. Default is 2R (twice the stop distance). The actual first partial target may be at internal liquidity rather than exactly 2R, but the OCO	

			bracket uses 2R as the guaranteed minimum.	

10. Trade Manager (M11: TradeManager)

The TradeManager takes over after the ExecutionManager has opened a position. It manages the trade through three distinct phases, each with its own logic and exit criteria. The TradeManager is the most complex module because it must handle multiple simultaneous state variables: position phase, trailing stop level, RBI/GBI status, 20 SMA health check status, and climax detection.

10.1 Phase Overview

PHASE	NAME	TARGET	ACTION
1	**Fixed Partial**	Nearest internal liquidity at ~2–2.5R	Exit 50% of position. Move stop to breakeven + 1 tick. Position is now risk-free. If T23 is OFF, skip this phase entirely (full position rides to macro target).
2	**Structural Node Trail**	Macro Draw on Liquidity	Remaining 50% trails behind 5m LVN Moats. After each 5m BOS, run Volume Profile on new leg → new HVN/LVN → move trail. RBI/GBI filter prevents false exits. 20 SMA health check tightens Tape Failure sensitivity when momentum weakens.
3	**Dynamic Climax Exit**	Macro target zone	Once price enters macro target zone, bot re-enters Kill Mode for exit. Watches for Buying Climax (tape speed vertical + delta positive but price stalls) or Tape Reversal (massive block sell). 200 SMA Watch Zone lowers the exhaustion threshold when the 200 SMA converges with the target.

10.2 Function Specifications

FUNCTION	INPUTS	OUTPUTS	DESCRIPTION & RATIONALE	TOGGLE
manage_phase1()	position: Position current_price: float internal_liquidity: float	Phase1Action (HOLD | PARTIAL_EXIT)	Monitors whether price has reached the internal liquidity target (~2–2.5R). When triggered: submits market order to close 50% of position, modifies OCO stop to breakeven + 1 tick, transitions to Phase 2. Logs the partial profit.	T23
manage_phase2()	position: Position bars_5m: list[Bar] of_data: OFReport velez_data: VelezReport	Phase2Action (HOLD | TRAIL_UPDATE | TAPE_FAILURE_EXIT)	The runner management phase. On each 5m bar: (1) Check for BOS. If new BOS, run Volume Profile → find new LVN → move trail. (2) If counter-trend candle appears, check	T19, T20, T21

			RBI/GBI first. If RBI/GBI confirmed, HOLD without further analysis. (3) If no RBI/GBI, evaluate CHOCH + OF (tape speed + delta). Fake CHOCH (low tape, declining delta) = hold. Real CHOCH = exit. (4) 20 SMA health check: if price closes below 2m 20 SMA for 3 consecutive bars, tighten Tape Failure threshold from 80% to 65%.	
check_tape_failure()	current_price: float hvn: HVN sell_delta_pct: float tape_velocity: float health_check_active: bool	bool (should_exit)	The Conditional Tape Failure exit. If price is in lower quadrant of HVN AND sell delta > threshold (80% normal, 65% when health check active) AND tape velocity is elevated → institutional buyers have pulled their orders → market-exit immediately. This turns a full 1R loss into a ~0.3R loss. Requires T18 ON.	T18
manage_phase3()	position: Position macro_target: float of_data: OFReport sma_200_near_target: bool	Phase3Action (HOLD | CLIMAX_EXIT)	Once price enters the macro target zone: re-subscribe to Kill Mode data. Watch for buying climax: tape speed > 400% avg + CVD heavily positive + price displacement near zero = sellers absorbing. OR tape reversal: whale block sell hits bid. If 200 SMA Watch Zone is active (sma_200_near_target = true), exit on absorption alone without requiring delta stall.	T22
calculate_structural_trail()	bars_5m: list[Bar] current_trail: float trade_direction: str	new_trail: float or None	After detecting a 5m BOS, runs a micro Volume Profile on the new structural leg. Identifies the LVN within that leg. New trail = bottom of LVN − (0.5 × 5m ATR) for longs (or top + buffer for shorts). Returns None if no BOS	T19

			detected (trail unchanged).	

11. Risk Overlord (M12: RiskOverlord)

The RiskOverlord is the outermost safety wrapper. It monitors all bot activity and has the authority to cancel orders, flatten positions, and terminate the process. It operates independently of all other modules and cannot be overridden by any analytical decision. Every order the bot attempts to place must pass through the RiskOverlord's validation before reaching the exchange.

11.1 Four Pillars

PILLAR	THRESHOLD	ACTION ON BREACH
Anti-Spam Rate Limiter	Max 3 executed orders per 60s. Max 10 per hour.	Block the order. Flatten all positions. Execute os._exit(1). Requires manual restart. This prevents a bug from generating thousands of orders.
Fat Finger Position Limit	Hardcoded MAX_POSITION_SIZE (e.g., 5 ES contracts).	Destroy the order. Kill the bot process. This prevents a single error from risking the entire account.
Hard Capital Thresholds	Max 3 consecutive losses OR -2% daily drawdown.	Flatten all positions. Terminate process. This ensures the bot cannot spiral into catastrophic loss.
Zombie Data Feed Monitor	DataBento exchange timestamp vs. local clock: if lag > 500ms, data is stale.	If stale > 10 seconds: flatten and shutdown. The bot cannot make valid decisions on stale data.

11.2 Sudden Move Classifier (Chameleon Protocol)

The Sudden Move Classifier detects and classifies three types of abnormal market conditions, each with a specific response protocol:

TYPE	TRIGGER	RESPONSE
A	Scheduled Event (CPI, FOMC, NFP, token unlock). Detected by EventCalendar within ±5 min window.	NEWS SHIELD: Block new entries. Tighten open stops to nearest 1m LVN. 3-minute post-event cooldown. Re-evaluate HTF bias and POIs after cooldown.
B	Organic Cascade (flash crash, liquidation waterfall). Detected by velocity/spread thresholds: tick velocity > 400% of 1hr avg for 5+ sec AND spread > 3 ticks ES.	CASCADE ENGAGEMENT: 50% position size override. Require 30s Ring Buffer minimum. Must see absorption + delta flip within 5 seconds. Miss is acceptable — if POI tap-and-reject < 2 sec without confirmation, skip. 5-minute cooldown after velocity normalizes.
C	Infrastructure Degradation. DataBento latency > 500ms, broker API errors, exchange status abnormal, heartbeat timeout.	FULL SHUTDOWN: Cancel all working orders (except exchange OCO). Freeze all logic. Rely on OCO brackets only for open positions. Crypto: attempt hedge on secondary exchange. Recovery: all health metrics green for 60 consecutive seconds.

11.3 Nuclear Flatten Sequence

The Nuclear Flatten is the most extreme safety measure. When triggered, the bot performs these steps in exact order: (1) Cancel ALL working orders. (2) Market-exit ALL positions to achieve Net_Position = 0. (3)

Execute os._exit(1) to physically terminate the Python process. (4) The bot remains offline until a human manually restarts it. There is no automatic recovery from Nuclear Flatten.

ENGINEERING REQUIREMENT

The Nuclear Flatten function must be reachable from EVERY code path. It must not depend on any module being in a healthy state. It must work even if the DataIngestor is dead, the OrderFlowEngine has crashed, or the TradeManager is in an inconsistent state. It communicates directly with the broker API using the simplest possible code path. No abstractions, no dependencies, no exception handling that could prevent execution.

12. Configuration & Deployment

12.1 Configuration File (TOML)

All 30 feature toggles and all configurable constants are stored in a single TOML file loaded at startup. The TOML format supports comments (critical for documenting rationale), sections (maps to the 5-layer toggle structure), and is human-readable. See the HTF MA Integration & Feature Toggle System document for the complete TOML structure and all toggle IDs (T01–T30).

12.2 Safety Lock

When the [system] section has live_mode = true, the configuration loader must refuse to set any toggle in [toggles.safety] (T24–T28) to false. Any attempt to disable a safety toggle in live mode should be logged as a CRITICAL alert and the toggle value should remain true. This is a hard engineering constraint that prevents accidental removal of safety measures during live trading.

12.3 Deployment Checklist

#	STEP	DETAILS
1	**Backtest all profiles (Baseline + 7 research profiles)**	Minimum 1000 trades per profile. Compare win rate, expectancy, profit factor, max drawdown, Sharpe ratio, trade frequency.
2	**Tune constants based on backtest results**	Adjust ATR multipliers, delta thresholds, grade boundaries. Re-run Baseline after each change. Never tune more than one variable at a time.
3	**Paper trade for 30 days minimum**	Full system with all toggles ON. Verify: order routing, OCO brackets, state transitions, Nuclear Flatten (trigger deliberately), data severing, Ring Buffer.
4	**Paper trade with live data for 14 days**	Connect to real DataBento and exchange feeds. Verify latency, fill simulation accuracy, and cost tracking.
5	**Live deployment with minimum capital**	Start with 1 contract maximum. All safety toggles ON. Monitor for 30 trading days. Compare live results to backtest baseline.
6	**Scale up gradually**	After 30 profitable live days, increase MAX_POSITION_SIZE by 1 contract. Continue monitoring. Never scale faster than performance justifies.

E N D O F D O C U M E N T

FLOF Matrix — Comprehensive Engineering Specification v1.0

This document should be read in conjunction with: Confluence Grading Rubric v2.0, Sudden Move Policy (Chameleon Protocol), and HTF MA Integration & Feature Toggle System. Together, these four documents define the complete system.