Thank you for the wonderful feedback! Your enthusiasm motivates me to maintain this high standard. Let's proceed with the IndicatorEngine - the computational heart of AlgoSpace.

# **Product Requirements Document (PRD): IndicatorEngine Component**

**Document Version:** 1.0  
 **Date:** June 20, 2025  
 **Component Level:** 2 - Data Pipeline  
 **Status:** Master Specification

## **1. Component Identity**

### **1.1 Component Name**

**IndicatorEngine** (Central Feature Calculation Engine)

### **1.2 Primary Role**

The IndicatorEngine is the system's computational powerhouse. It transforms raw OHLCV bars into the rich set of technical indicators, market profile features, and regime detection inputs that drive trading decisions. It serves as the single source of truth for all calculated features.

### **1.3 Single Responsibility**

To calculate all technical indicators and features required by the trading strategy, maintain their current values in a centralized Feature Store, and emit a comprehensive update event when calculations complete.

### **1.4 Critical Design Principles**

* **DIR-DATA-01:** LVN calculations performed on Heiken Ashi data
* **DIR-DATA-02:** Core indicators use default parameters only
* All calculations happen in one place to ensure consistency
* Single INDICATORS\_READY event prevents partial updates

## **2. Inputs & Dependencies**

### **2.1 Configuration Input**

From settings.yaml:

indicators:

# Core indicators with DEFAULT parameters (DIR-DATA-02)

mlmi:

k\_neighbors: 5 # Default - DO NOT CHANGE

trend\_length: 14 # Default - DO NOT CHANGE

nwrqk:

bandwidth: 46 # Default - DO NOT CHANGE

alpha: 8 # Default - DO NOT CHANGE

fvg:

threshold: 0.001 # 0.1% minimum gap size

lvn:

lookback\_periods: 20 # Rolling 20 bars for volume profile

strength\_threshold: 0.7 # 70% below POC = LVN

mmd:

signature\_degree: 3 # For regime detection

### **2.2 Event Inputs**

**Two Input Events:**

1. **NEW\_5MIN\_BAR**
   * Source: BarGenerator
   * Used for: FVG detection
2. **NEW\_30MIN\_BAR**
   * Source: BarGenerator
   * Used for: MLMI, NW-RQK, LVN, MMD

### **2.3 Internal Dependencies**

* Historical price buffers for indicator calculations
* Volume profile accumulator for LVN detection
* Heiken Ashi converter
* Feature Store (internal data structure)

## **3. Processing Logic**

### **3.1 Dual-Path Processing**

The IndicatorEngine maintains two parallel processing paths:

NEW\_5MIN\_BAR → 5-Minute Path → FVG Detection → Update Feature Store

↓

NEW\_30MIN\_BAR → 30-Minute Path → HA Conversion → MLMI, NW-RQK, LVN, MMD → Update Feature Store

↓

Check if all updates complete

↓

Emit INDICATORS\_READY

### **3.2 Processing 5-Minute Bars**

**On NEW\_5MIN\_BAR Event:**

1. **Update Price History**
   * Add new bar to 5-minute history buffer
   * Maintain rolling window (last 100 bars)

**FVG Detection (on Standard Candles)** For Bullish FVG:

- Current bar low > bar[-2] high

- Bar[-1] close > bar[-2] high

- Gap size > threshold (0.1%)

For Bearish FVG:

- Current bar high < bar[-2] low

- Bar[-1] close < bar[-2] low

- Gap size > threshold (0.1%)

1. **FVG Tracking**
   * Add new FVGs to active list
   * Check for mitigation of existing FVGs:
     + Bullish mitigated when price returns to lower boundary
     + Bearish mitigated when price returns to upper boundary
   * Remove mitigated or expired FVGs (>50 bars old)

**Update Feature Store** Features updated:

- fvg\_bullish\_active: bool

- fvg\_bearish\_active: bool

- fvg\_nearest\_level: float (price)

- fvg\_age: int (bars since creation)

- fvg\_mitigation\_signal: bool (just mitigated)

### **3.3 Processing 30-Minute Bars**

**On NEW\_30MIN\_BAR Event:**

**Heiken Ashi Conversion** HA\_Close = (Open + High + Low + Close) / 4

HA\_Open = (Previous\_HA\_Open + Previous\_HA\_Close) / 2

HA\_High = max(High, HA\_Open, HA\_Close)

HA\_Low = min(Low, HA\_Open, HA\_Close)

1. **Update History Buffers**
   * Add HA bar to 30-minute HA history
   * Add standard bar to volume profile buffer
   * Maintain appropriate rolling windows

**Calculate MLMI (on HA data)** Process:

1. Calculate RSI on HA close prices (14 period)

2. Apply k-NN algorithm (k=5) to predict next value

3. Calculate WMA of predictions (5 period)

4. Generate signal: 1 (bullish cross), -1 (bearish cross), 0 (neutral)

**Calculate NW-RQK (on HA data)** Process:

1. Apply Nadaraya-Watson regression with RQ kernel

2. Use bandwidth=46, alpha=8 (defaults)

3. Calculate regression curve value

4. Calculate slope (rate of change)

5. Generate signal: 1 (turning up), -1 (turning down), 0 (flat)

**Update Volume Profile & Calculate LVN (on HA data)** Process:

1. Update rolling 20-bar volume profile

2. Identify Point of Control (POC) - highest volume price

3. Find all price levels with volume < 70% of POC volume

4. Calculate "strength score" for each LVN:

- Number of historical touches

- Average bounce magnitude

- Recency weighting

5. Identify nearest LVN to current price

**Calculate MMD Features (on HA data)** For Regime Detection Engine:

1. Calculate log returns from HA closes

2. Compute path signatures (degree 3)

3. Calculate volatility metrics

4. Package as MMD feature vector

**Update Feature Store** Features updated:

- mlmi\_value: float (0-100)

- mlmi\_signal: int (-1, 0, 1)

- nwrqk\_value: float (price level)

- nwrqk\_slope: float (rate of change)

- nwrqk\_signal: int (-1, 0, 1)

- lvn\_nearest\_price: float

- lvn\_nearest\_strength: float (0-100)

- lvn\_distance\_points: float

- mmd\_features: array (for RDE)

### **3.4 Feature Store Management**

**Structure:**

Feature Store (Dictionary):

{

# 30-minute features

'mlmi\_value': 55.4,

'mlmi\_signal': 1,

'nwrqk\_value': 5150.25,

'nwrqk\_slope': 0.15,

'nwrqk\_signal': 1,

'lvn\_nearest\_price': 5145.00,

'lvn\_nearest\_strength': 85.5,

'lvn\_distance\_points': 5.25,

# 5-minute features

'fvg\_bullish\_active': True,

'fvg\_nearest\_level': 5148.50,

'fvg\_age': 3,

'fvg\_mitigation\_signal': False,

# Regime features

'mmd\_features': [0.012, -0.003, 0.008, ...],

# Metadata

'last\_update\_5min': datetime,

'last\_update\_30min': datetime

}

### **3.5 Event Emission Logic**

**When to Emit INDICATORS\_READY:**

The engine tracks updates from both timeframes. It emits the event when:

1. A 30-minute update completes, OR
2. A 5-minute update completes AND 30-minute features exist

This ensures downstream components always have a complete feature set.

## **4. Outputs & Events**

### **4.1 Primary Output**

**Event Name:** INDICATORS\_READY **Frequency:** Every 5 minutes (when either timeframe updates) **Payload:** Complete copy of Feature Store

### **4.2 Feature Categories in Output**

1. **Entry Signal Features**
   * MLMI value and signal
   * NW-RQK value, slope, and signal
   * FVG status and mitigation
2. **Risk Context Features**
   * LVN locations and strength
   * Distance to nearest LVN
3. **Regime Detection Features**
   * MMD feature vector
   * Volatility metrics
4. **Metadata**
   * Last update timestamps
   * Calculation status flags

## **5. Critical Requirements**

### **5.1 Calculation Requirements**

* **Accuracy:** All indicators must match reference implementations exactly
* **Determinism:** Same input must always produce same output
* **Default Parameters:** Core indicators MUST use defaults (DIR-DATA-02)
* **HA Consistency:** 30-min indicators use HA, 5-min FVG uses standard

### **5.2 Performance Requirements**

* **5-min Calculations:** Complete within 50ms
* **30-min Calculations:** Complete within 100ms
* **Memory Usage:** Bounded by fixed-size history buffers

### **5.3 Data Integrity Requirements**

* **Atomic Updates:** Feature Store updates must be atomic
* **No Partial States:** All features updated before event emission
* **Synchronization:** No race conditions between timeframes

### **5.4 Operational Requirements**

* **Stateless Between Runs:** Rebuild from bar stream each time
* **Single Symbol:** Process one asset only (DIR-SYS-02)
* **Event Ordering:** Maintain chronological order

## **6. Integration Points**

### **6.1 Upstream Integration**

**From BarGenerator:**

* Events: NEW\_5MIN\_BAR, NEW\_30MIN\_BAR
* Data: Complete OHLCV bars
* Timing: Synchronized with market time

### **6.2 Downstream Integration**

**Primary Consumers:**

1. **SynergyDetector**
   * Uses: Signal features to detect valid setups
   * Needs: MLMI, NW-RQK, FVG signals
2. **MatrixAssemblers**
   * Uses: All features to build agent matrices
   * Needs: Complete, consistent feature set
3. **Main MARL Core**
   * Uses: LVN features for risk context
   * Needs: Accurate strength scores

## **7. Calculation Specifications**

### **7.1 Heiken Ashi Formula**

HA\_Close[i] = (Open[i] + High[i] + Low[i] + Close[i]) / 4

HA\_Open[i] = (HA\_Open[i-1] + HA\_Close[i-1]) / 2

HA\_High[i] = max(High[i], HA\_Open[i], HA\_Close[i])

HA\_Low[i] = min(Low[i], HA\_Open[i], HA\_Close[i])

First bar: HA\_Open = (Open + Close) / 2

### **7.2 LVN Strength Score**

Strength = W1 \* TouchCount +

W2 \* AvgBounce +

W3 \* RecencyFactor +

W4 \* VolumeRatio

Where:

- W1-W4: Weights (must sum to 1.0)

- TouchCount: Number of times tested

- AvgBounce: Average move after touch

- RecencyFactor: Recent touches weighted higher

- VolumeRatio: Volume at level vs. average

## **8. Error Handling**

### **8.1 Calculation Errors**

* **Insufficient Data:** Use partial calculations or defaults
* **Invalid Values:** Log warning, use previous value
* **Division by Zero:** Check denominators, use small epsilon

### **8.2 System Errors**

* **Memory Issues:** Log critical, system exit
* **Event Bus Failure:** Log critical, no recovery

## **9. Logging Specification**

### **9.1 Startup**

* "IndicatorEngine initialized with indicators: [list]"
* "Feature Store created with [n] features"

### **9.2 Operational**

* Each calculation: "[INDICATOR] calculated for [timestamp]"
* Each emission: "INDICATORS\_READY emitted with [n] features"
* Warnings for edge cases

### **9.3 Debug Mode**

* Detailed calculation steps
* Intermediate values
* Feature Store contents

## **10. Testing Considerations**

### **10.1 Unit Tests**

* Each indicator calculation in isolation
* HA conversion accuracy
* FVG detection logic
* LVN strength calculations

### **10.2 Integration Tests**

* Full pipeline from bars to features
* Multi-timeframe synchronization
* Feature Store consistency

### **10.3 Validation Tests**

* Compare with TradingView/reference implementations
* Verify default parameters
* Check edge cases

## **11. Implementation Notes**

### **11.1 Indicator Libraries**

* Consider using established libraries where available
* Validate against reference implementations
* Document any deviations

### **11.2 Numerical Stability**

* Use appropriate data types (float64)
* Handle edge cases in calculations
* Avoid accumulating rounding errors

### **11.3 Performance Optimization**

* Pre-allocate arrays
* Use vectorized operations
* Cache repeated calculations

## **12. What This Component Does NOT Do**

* Does NOT make trading decisions
* Does NOT store historical indicators
* Does NOT modify indicator parameters
* Does NOT communicate with external services
* Does NOT persist state between runs
* Does NOT handle multiple symbols
* Does NOT implement custom indicators

This IndicatorEngine PRD completes the data pipeline foundation. It transforms raw market data into the rich feature set that powers the AI decision-making system, while maintaining strict adherence to the architectural principles.

We've now completed Level 2 (Data Pipeline). Ready to move to Level 3 (Feature Preparation) with the MatrixAssemblers?