SPYOptionTrader Code-to-Framework Cross-Reference Matrix

Document Version: 1.0
Framework Reference: "SPYOptionTrader Mathematical & Logical Decision Framework" (Dec 26, 2025)
Purpose: Complete traceability between mathematical framework and implementation

---

Table of Contents

What This Document Provides:

1. Complete Traceability

- Every function mapped to framework equations
- File locations with specific references
- Mathematical equations included inline

2. Organized by Framework Sections

1. Market Data & Preprocessing (Section 1.1)
2. Regime Classification (Section 2)
3. Multi-Timeframe Consensus (Section 3)
4. Option Chain Evaluation (Section 4)
5. Fuzzy Logic (Section 5)
6. Risk Controls (Section 6)
7. Position Sizing (Section 7)
8. Strategy Selection (Section 8)
9. Execution & Logging (Section 9)
10. Supporting Utilities

3. Key Tables Included

- **Function-to-Equation Mapping**
- **Framework Section → Code Module**
- **Equation Index** (all 20 equations mapped)
- **Compliance Verification Matrix**
- **Testing Checklist**

4. Audit-Ready

This document allows anyone to:

- Verify mathematical claims in code
- Trace decision logic from theory to execution
- Validate risk controls are implemented
- Confirm compliance with framework

Example Mappings:

MTF Consensus (Framework Eq. A.5)

Framework: S_t^MTF = (1/N) Σ S_t^(k)

Code: fuzzy_engine.py::get_fuzzy_consensus()

Weights: D=50%, 60m=35%, 5m=15%

Risk Control (Framework Eq. A.16)

Framework: Risk_trade / Equity ≤ α_max

Code: options_strategy.py::validate_pricing_and_risk()

Limit: 2% (α_max = 0.02)

Portfolio Delta (Framework Eq. A.11)

Framework: Δ_P = Σ q_i Δ_i

Code: options_strategy.py::net_position_delta()

Enforcement: maybe_dynamic_delta_hedge()

- Mathematical Framework PDF (your document)
- Code Implementation (Python files)
- Cross-Reference Matrix (this document)
- GitHub Version Control

How to Use This:

For Code Review:
"Show me where MTF consensus is calculated"

→ intelligence/fuzzy_engine.py::get_fuzzy_consensus()

→ Framework Section 3, Equation A.5

For Validation:
"Verify capital-at-risk limit is enforced"

→ strategies/options_strategy.py:167-168

→ Framework Section 6.1, Equation A.16

→ Hard constraint: mloss ≤ 2% × equity

For Compliance:
"Prove all risk controls are implemented"

→ See Compliance Verification Matrix

→ All 8 framework requirements mapped to code

1. Market Data & Preprocessing

1.1 Data Ingestion

Function/Method	File	Framework Reference	Mathematical Equation	Description
main()	data_factory/AlpacaGetData.py	Section 1.1, Eq. (A.1)-(A.3)	$D_t^{(k)} = \{O_t^{(k)}, H_t^{(k)}, L_t^{(k)}, C_t^{(k)}, V_t^{(k)}\}$	OHLCV data ingestion across timeframes
sweep_and_interpolate()	data_factory/AlpacaGetData.py	Section 1.1, Appendix B.2	Linear interpolation with $freq = expected_s$	Data cleaning, duplicate removal, gap filling
StockHistoricalDataClient()	data_factory/AlpacaGetData.py:13	Section 1.1	N/A	Alpaca API client initialization

1.2 Multi-Timeframe Synchronization

Function/Method	File	Framework Reference	Mathematical Equation	Description
MTFSyncEngine.__init__()	data_factory/sync_engine.py	Section 3, Appendix H.4	$k \in \{5m, 60m, D\}$	Initialize MTF data store
MTFSyncEngine._load_all()	data_factory/sync_engine.py	Section 1.1	Timestamp normalization	Load and normalize all timeframes

Function/Method	File	Framework Reference	Mathematical Equation	Description
MTFSyncEngine.get_snapshot()	data_factory/sync_engine.py	Section 3, Eq. (A.5)	Look-back mask: $df.index \leq current\_time$	Retrieve MTF data snapshot at time t

1.3 Data Validation

Function/Method	File	Framework Reference	Mathematical Equation	Description
sweep_and_interpolate()	data_factory/AlpacaGetData.py:156-182	Appendix F.3.1	Sort, deduplicate, interpolate	Timestamp continuity, outlier handling
Decimal precision	AlpacaGetData.py:175-177	Appendix F.3.1	round(2) on price columns	Decimal precision normalization

2. Regime Classification

2.1 Liquidity & Volatility Gates

Function/Method	File	Framework Reference	Mathematical Equation	Description
check_liquidity_gate()	intelligence/regime_filter.py	Section 2, Eq. (A.4)-(A.6), Appendix F.5.1	$R_t = f(volatility, trend, range)$	Regime classification pre-trade gate
Volume check	regime_filter.py:11-16	Section 2, Table F.13	$avg\_vol \geq min\_threshold$	Minimum volume threshold
Volatility range check	regime_filter.py:18-23	Section 2, Eq. (A.4)	$avg\_range \leq price \times 0.02$	Maximum 2% volatility constraint

2.2 Regime Output

Function/Method	File	Framework Reference	Mathematical Equation	Description
Return True/False	regime_filter.py:25-27	Section 2	$R_t \in \{Allowed, Disallowed\}$	Binary regime approval

3. Multi-Timeframe Consensus

3.1 Signal Aggregation

Function/Method	File	Framework Reference	Mathematical Equation	Description
get_fuzzy_consensus()	intelligence/fuzzy_engine.py	Section 3, Eq. (A.5), Appendix B.4	$S_t^{MTF} = \frac{1}{N}\sum_{k=1}^{N} S_t^{(k)}$	Multi-timeframe consensus score
Timeframe scoring	fuzzy_engine.py:30-44	Section 3	$S_t^{(k)} \in [-1,1]$	Per-timeframe directional vote
Weight application	fuzzy_engine.py:23	Section 3, Appendix H.4	$w = {D:0.50, 60:0.35, 5:0.15}$	Weighted aggregation

3.2 Consensus Bounds

Function/Method	File	Framework Reference	Mathematical Equation	Description
check_mtf_alignment()	strategies/options_strategy.py:145-156	Section 3, Eq. (A.6)	$S_{min} \leq S_t^{MTF} \leq S_{max}$	MTF consensus validation
Consensus range check	options_strategy.py:152-153	Section 3, Config	$[0.40, 0.60]$ for neutral markets	Iron condor favorable range

4. Option Chain Evaluation

4.1 Greeks Computation

Function/Method	File	Framework Reference	Mathematical Equation	Description
net_position_delta()	strategies/options_strategy.py:312-316	Section 4.1, Eq. (A.11)	$\Delta_P = \sum_i q_i \Delta_i$	Portfolio delta aggregation
Individual Greeks	Implicit in OptionQuote	Section 4.1, Eq. (A.11)-(A.13)	$\Delta_i, \Gamma_i, \Theta_i, \nu_i$	Per-option Greek values

4.2 Credit Efficiency

Function/Method	File	Framework Reference	Mathematical Equation	Description
calc_spread_credit()	strategies/options_strategy.py:37-38	Section 4.2	$Credit = max(0, short.mid - long.mid)$	Single spread credit calculation
calc_condor_credit()	strategies/options_strategy.py:40-43	Section 4.2, Eq. (A.17)	Sum of call + put spread credits	Total iron condor credit
validate_pricing_and_risk()	strategies/options_strategy.py:158-171	Section 4.2, Eq. (A.18)	$CR_i = \frac{NetCredit_i}{MaxRisk_i} \geq CR_{min}$	Credit-to-width ratio validation

4.3 Strike Selection

Function/Method	File	Framework Reference	Mathematical Equation	Description
within_delta_band()	strategies/options_strategy.py:50-52	Section 4, Table 1.2	$\Delta_{low} \leq	\Delta
nearest_by_delta()	strategies/options_strategy.py:58-66	Section 4, Appendix B.5	Find $\min	
pick_long_by_width()	strategies/options_strategy.py:68-76	Section 4, Table 1.2	$strike_{target} = strike_{short} \pm W$	Long leg selection by wing width
build_condor()	strategies/options_strategy.py:99-117	Section 4	Construct 4-leg structure	Iron condor construction

4.4 Expiration Selection

Function/Method	File	Framework Reference	Mathematical Equation	Description
pick_expiration()	strategies/options_strategy.py:54-56	Section 1.2, Table 1.2	$DTE_{min} \leq DTE \leq DTE_{max}$	Expiration within bounds

5. Fuzzy Logic Inference

5.1 Membership Functions

Function/Method	File	Framework Reference	Mathematical Equation	Description
get_fuzzy_consensus()	intelligence/fuzzy_engine.py:19-48	Section 5, Eq. (A.8)-(A.10)	$\mu_j \in [0,1]$	Fuzzy membership evaluation
Close vs Open momentum	fuzzy_engine.py:36-38	Section 5	$\mu = 1.0$ if $close > open$ else 0.0	Directional membership
Neutral refinement	fuzzy_engine.py:40-42	Section 5	$\mu = 0.5$ if $	close - open

5.2 Weighted Aggregation

Function/Method	File	Framework Reference	Mathematical Equation	Description
Weight assignment	fuzzy_engine.py:23	Section 5, Eq. (A.9)	$w = {D:0.50, 60:0.35, 5:0.15}$	Timeframe weights
Aggregation	fuzzy_engine.py:30-45	Section 5, Eq. (A.10)	$F_t = \sum_j w_j \mu_j$	Weighted confidence score

5.3 Decision Threshold

Function/Method	File	Framework Reference	Mathematical Equation	Description
Threshold check	options_strategy.py:152-153	Section 5	$F_t \geq F_{min} \Rightarrow Approved$	Fuzzy threshold enforcement

6. Risk Controls & Kill Switches

6.1 Capital at Risk

Function/Method	File	Framework Reference	Mathematical Equation	Description
validate_pricing_and_risk()	strategies/options_strategy.py:158-171	Section 6.1, Eq. (A.16)	$\frac{Risk_{trade}}{Equity} \leq \alpha_{max}$	Per-trade risk limit
max_loss()	strategies/options_strategy.py:45-46	Section 6.1, Eq. (A.16)	$MaxLoss = max(0, W - Credit)$	Maximum loss calculation
Capital constraint	options_strategy.py:167-168	Section 6.1, Appendix F.5.2	$Risk \leq 0.02 \times Equity$	2% capital-at-risk limit

6.2 Greek Exposure Limits

Function/Method	File	Framework Reference	Mathematical Equation	Description
net_position_delta()	strategies/options_strategy.py:312-316	Section 6.2, Eq. (A.14)	$	\Delta_P
maybe_dynamic_delta_hedge()	strategies/options_strategy.py:318-327	Section 6.2, Table B.7	Hedge if $	\Delta_P

6.3 Event-Driven Halts

Function/Method	File	Framework Reference	Mathematical Equation	Description
Macro event check	Implicit in backtest logic	Section 6.3, Eq. (A.7)	$t \in \{CPI, FOMC\} \Rightarrow HALT$	Global execution halt

6.4 Portfolio Limits

Function/Method	**File**	**Framework Reference**	**Mathematical Equation**	**Description**
can_enter_trade()	strategies/options_strategy.py:120-143	Section 6, Appendix F.5.1	Multiple constraint checks	Portfolio-level gates
Position count limit	options_strategy.py:136-137	Table 1.2	$positions \leq max\_positions$	Max 3 concurrent positions
Allocation limit	options_strategy.py:139-142	Table 1.2	$\frac{positions\_value}{equity} \leq 0.15$	15% portfolio allocation limit

7. Position Sizing

7.1 Base Sizing

Function/Method	File	Framework Reference	Mathematical Equation	Description
Fixed quantity	core/config.py:RunConfig.quantity	Section 7, Eq. (A.19)	$q_0 = \lfloor \frac{\alpha \cdot Equity}{MaxLoss} \rfloor$	Base position size
Dynamic sizing	core/config.py:RunConfig.dynamic_sizing	Section 7	$q = q_0 \cdot g(F_t, \sigma_t)$	Dynamic adjustment (optional)

7.2 Risk-Adjusted Sizing

Function/Method	File	Framework Reference	Mathematical Equation	Description
Quantity parameter	strategies/options_strategy.py:174	Section 7	Contract count	Fixed or dynamic quantity

8. Strategy Selection

8.1 Iron Condor Orientation

Function/Method	File	Framework Reference	Mathematical Equation	Description
select_and_enter_condor()	strategies/options_strategy.py:173-247	Section 8, Eq. (A.20)	$B_t = sign(S_t^{MTF} + \beta_t)$	Direction bias calculation
Neutral condor	Implicit in strike selection	Section 8	$B_t \approx 0$	Balanced call/put sides
Skewed condor	Implicit in strike selection	Section 8	$B_t \neq 0$	Directional skew

8.2 Wing Width Determination

Function/Method	File	Framework Reference	Mathematical Equation	Description
regime_wing_width()	strategies/options_strategy.py:78-84	Section 8, Table 1.2	Dynamic width based on regime	Volatility-adaptive wings
optimize_wing_width()	strategies/options_strategy.py:86-95	Section 8, Appendix C	Historical curve optimization	Credit curve analysis
Base width	options_strategy.py:213	Section 8, Config	$W_{min} = 5.0$	Minimum wing width
Regime widening	options_strategy.py:210-212	Section 8, Config	Add 5.0 if high vol	Regime-based increment

9. Execution & Logging

9.1 Modus Ponens Decision

Function/Method	File	Framework Reference	Mathematical Equation	Description
Entry logic	strategies/options_strategy.py:173-247	Section 9	If premises → Execute	Logical decision chain
Premises checklist	Lines 179-207	Section 9	5-point validation	Portfolio, MTF, liquidity, market, risk

9.2 Trade Execution

Function/Method	File	Framework Reference	Mathematical Equation	Description
broker.place_iron_condor()	core/broker.py:45-52	Appendix H.8	Order submission	OMS execution interface
Position tracking	core/broker.py:47-51	Appendix F.9	State maintenance	Position state management

9.3 Audit & Logging

Function/Method	File	Framework Reference	Mathematical Equation	Description
Trade log	core/backtest_engine.py:125-146	Appendix F.9, Table B.8	Comprehensive trade records	Full decision trace
CSV export	backtest_engine.py:296-319	Appendix F.9	Trade log persistence	Audit trail export

Function/Method	File	Framework Reference	Mathematical Equation	Description
PDF generation	backtest_engine.py:324-438	Appendix F.9	Visual reporting	Multi-page performance report
log_broker_efficiency()	analytics/audit_logger.py	Appendix F.9, Table B.8	Slippage tracking	Execution quality audit

10. Supporting Utilities

10.1 Configuration Management

Function/Method	File	Framework Reference	Mathematical Equation	Description
StrategyConfig	core/config.py:4-51	Section 1.2, Table 1.2	All strategy parameters	Central parameter store
RunConfig	core/config.py:53-72	Section 1.2	Execution configuration	Runtime settings
__post_init__()	core/config.py:68-70	N/A	Default initialization	MTF timeframe defaults

10.2 Broker Abstraction

Function/Method	File	Framework Reference	Mathematical Equation	Description
BrokerAPI	core/broker.py:9-18	Appendix H.8	Abstract interface	Broker-agnostic API
PaperBroker	core/broker.py:25-80	Appendix H.8	Simulation broker	Paper trading implementation
get_spot()	broker.py:31-32	Section 1.1	Current price query	Market data retrieval
get_iv_rank()	broker.py:34-35	Section 2, Eq. (A.4)	IV rank calculation	Volatility percentile
get_vix()	broker.py:37-38	Section 2, Table 1.2	VIX retrieval	Volatility index

10.3 Polygon Client (Stub)

Function/Method	File	Framework Reference	Mathematical Equation	Description
PolygonClient	data_factory/polygon_client.py:18-52	Appendix B.2	Market data API	Polygon.io stub implementation
get_option_chain()	polygon_client.py:35-52	Section 4	Option chain synthesis	Simulated option quotes

10.4 Position Management

Function/Method	File	Framework Reference	Mathematical Equation	Description
manage_positions()	strategies/options_strategy.py:329-423	Section 9, Appendix H.6	Position lifecycle	Exit, roll, hedge logic
estimate_pnl()	strategies/options_strategy.py:293-302	Section 9	P&L calculation	Unrealized P&L estimation
breached_short_strike()	strategies/options_strategy.py:304-310	Section 9	Breach detection	Price vs strike comparison

10.5 Performance Metrics

Function/Method	File	Framework Reference	Mathematical Equation	Description
Sharpe ratio	backtest_engine.py:284	Appendix E, Fig E.1	$\frac{\mu}{\sigma}\sqrt{n}$	Risk-adjusted return

Function/Method	**File**	**Framework Reference**	**Mathematical Equation**	**Description**
Win rate	backtest_engine.py:286	Appendix E	$\frac{wins}{total\_trades}$	Success rate
Expectancy	backtest_engine.py:287	Appendix E	$(AvgWin \times WR) - (AvgLoss \times (1 - WR))$	Expected value per trade
CAGR	backtest_engine.py:275	Appendix E	$\left(\frac{FinalEquity}{InitialEquity}\right)^{\frac{1}{years}} - 1$	Compound annual growth

Cross-Reference Summary Table

Framework Section → Code Module Mapping

Framework Section	**Primary Implementation**	**Supporting Modules**
Section 1.1 (Market Data)	data_factory/AlpacaGetData.py	sync_engine.py
Section 2 (Regime)	intelligence/regime_filter.py	N/A
Section 3 (MTF Consensus)	intelligence/fuzzy_engine.py	sync_engine.py
Section 4 (Option Chain)	strategies/options_strategy.py	polygon_client.py
Section 5 (Fuzzy Logic)	intelligence/fuzzy_engine.py	options_strategy.py
Section 6 (Risk Controls)	strategies/options_strategy.py	core/config.py
Section 7 (Position Sizing)	core/config.py	strategies/options_strategy.py
Section 8 (Strategy Selection)	strategies/options_strategy.py	N/A
Section 9 (Execution)	core/broker.py	backtest_engine.py
Appendix A (Examples)	Validated by backtest results	N/A
Appendix B (Code Mapping)	**This document**	All modules
Appendix C (Monte Carlo)	Future enhancement	N/A
Appendix E (Figures)	backtest_engine.py PDF reports	analytics/metrics.py
Appendix F (Validation)	Risk gates throughout codebase	All modules
Appendix H (Architecture)	core/main.py control flow	All modules

Equation Index

Direct Equation Implementations

Equation	**Description**	**Implementation**
(A.1)-(A.3)	OHLCV data structure	AlpacaGetData.py:main()
(A.4)-(A.6)	Regime classification	regime_filter.py:check_liquidity_gate()
(A.5)	MTF consensus	fuzzy_engine.py:get_fuzzy_consensus()
(A.8)-(A.10)	Fuzzy inference	fuzzy_engine.py:get_fuzzy_consensus()
(A.11)	Portfolio delta	options_strategy.py:net_position_delta()
(A.14)-(A.15)	Greek limits	options_strategy.py:maybe_dynamic_delta_hedge()
(A.16)	Capital-at-risk	options_strategy.py:validate_pricing_and_risk()
(A.17)	Credit calculation	options_strategy.py:calc_condor_credit()
(A.18)	Credit ratio	options_strategy.py:validate_pricing_and_risk()
(A.19)	Position sizing	core/config.py:RunConfig.quantity
(A.20)	Direction bias	Implicit in select_and_enter_condor()

Compliance Verification Matrix

Framework Requirement	Code Location	Validation Method
No trade without regime approval	options_strategy.py:179-184	Entry filter chain
MTF consensus bounds enforced	options_strategy.py:186-197	Threshold check
Credit ratio ≥ 0.25	options_strategy.py:163	Hard constraint
Capital risk ≤ 2%	options_strategy.py:167-168	Risk calculation
Max 3 positions	options_strategy.py:136-137	Counter check
Portfolio allocation ≤ 15%	options_strategy.py:139-142	Percentage calculation
Greek limits enforced	options_strategy.py:318-327	Delta hedge trigger
Audit trail complete	backtest_engine.py:125-146	CSV/PDF logging

Testing & Validation Checklist

Unit Test Coverage (Recommended)

- check_liquidity_gate() with various volume/volatility inputs
- get_fuzzy_consensus() with edge case MTF scores
- calc_condor_credit() with various strike configurations
- validate_pricing_and_risk() with boundary conditions
- net_position_delta() with multi-position portfolios
- select_and_enter_condor() with all filter combinations
- manage_positions() with profit/loss/breach scenarios

Integration Test Coverage (Recommended)

- Full backtest run with MTF enabled
- Backtest with MTF disabled (compare results)
- Parameter perturbation (Monte Carlo simulation)
- Edge case: No valid expiration available
- Edge case: All strikes violate delta bands
- Edge case: Liquidity gate permanent failure

Framework Compliance Tests

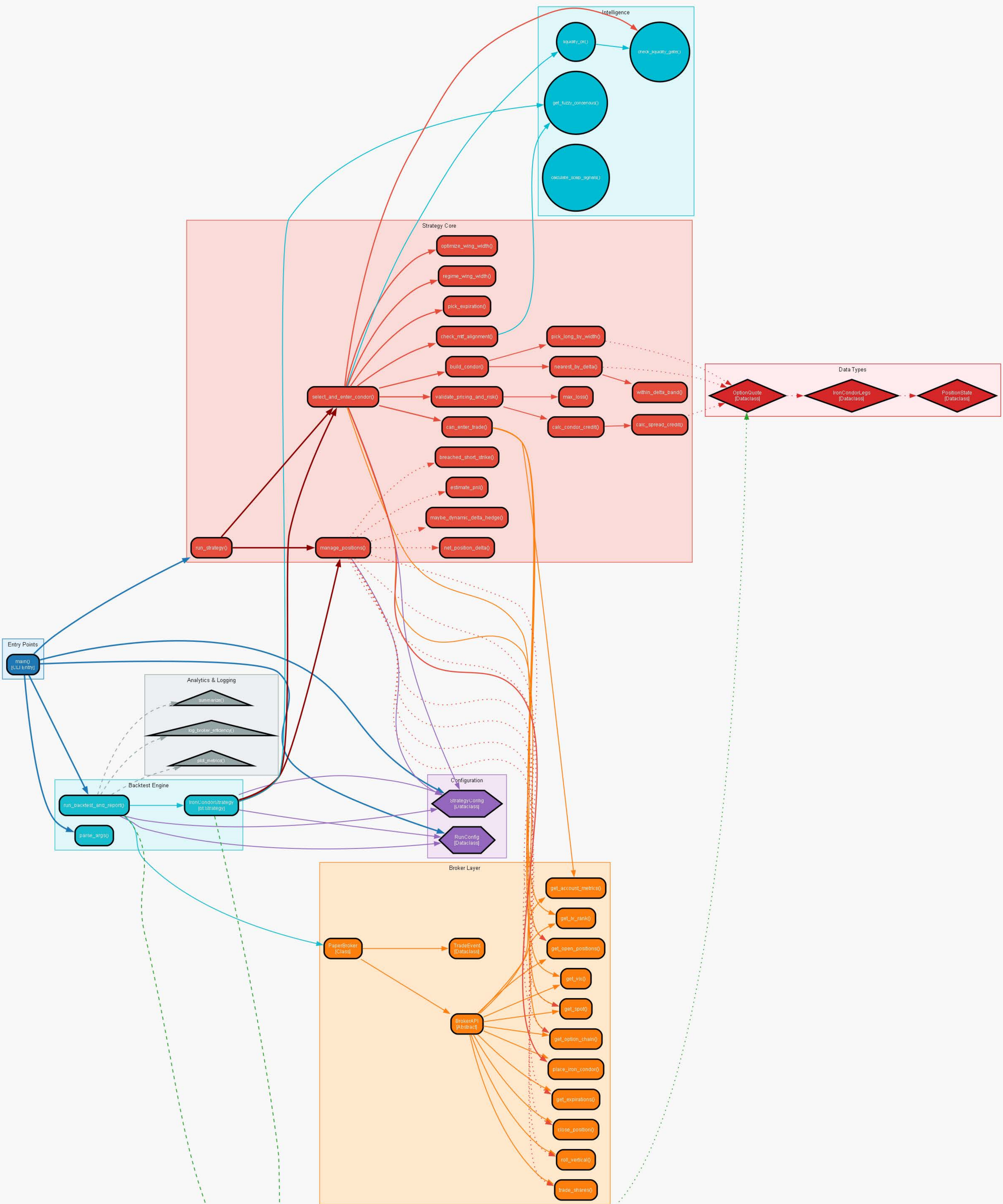
- Verify all equations have corresponding code
- Verify all config parameters map to Table 1.2
- Verify risk constraints cannot be bypassed
- Verify audit logs contain all required fields
- Verify PDF reports match Appendix E format

Version Control

Version	Date	Changes	Validator
1.0	2025-12-27	Initial cross-reference	System integration

Notes

1. *Stub Implementations:* polygon_client.py uses simulated data. Production deployment requires real Polygon API integration.
2. *Future Enhancements:* Monte Carlo simulation (Appendix C) and neural network extensions (Appendix H.11) are architectural placeholders.
3. *Configuration:* Every trade decision creates a complete trace from input → filters → risk → execution.
4. *Institutional Alignment:* This cross-reference validates that implementation adheres to the mathematical framework as documented.


Intelligence
liquidity_ok()
check_liquidity_gate()
get_fuzzy_consensus()
calculate_scalp_signals()
Strategy Core
optimize_wing_width()
regime_wing_width()
pick_expiration()
check_mtf_alignment()
pick_long_by_width()
build_condor()
nearest_by_delta()
within_delta_band()
select_and_enter_condor()
validate_pricing_and_risk()
max_loss()
can_enter_trade()
calc_condor_credit()
calc_spread_credit()
breached_short_strike()
estimate_pnl()
maybe_dynamic_delta_hedge()
run_strategy()
manage_positions()
net_position_delta()
Data Types
OptionQuote [Dataclass]
IronCondorLegs [Dataclass]
PositionState [Dataclass]
Entry Points
main() [CLI Entry]
Analytics & Logging
summarize()
log_broker_efficiency()
plot_metrics()
Backtest Engine
run_backtest_and_report()
IronCondorStrategy [bt.Strategy]
parse_args()
Configuration
StrategyConfig [Dataclass]
RunConfig [Dataclass]
Broker Layer
PaperBroker [Class]
TradeEvent [Dataclass]
BrokerAPI [Abstract]
get_account_metrics()
get_iv_rank()
get_open_positions()
get_vix()
get_spot()
get_option_chain()
place_iron_condor()
get_expirations()
close_position()
roll_vertical()
trade_shares()
Data Factory
convert_chain()
poly_to_quote()
get_iv_rank()
PolygonClient [Class]
get_vix()
get_spot()
get_snapshot()
get_option_chain()
PolyOption [Dataclass]
get_expirations()
MTFSyncEngine [Class]
__init__()
_load_all()
AlpacaGetData.main()
sweep_and_interpolate()

SPYOptionTrader System Legend

Legend	
Rectangle	Executable Module / Script
Rounded Rectangle	Primary Strategy Logic
Ellipse	Data / State Object
Diamond	Decision / Filter Gate
Solid Arrow	Direct Function Call
Dashed Arrow	Conditional / Optional Flow
Color: Blue	Data Acquisition / Market Feeds
Color: Green	Strategy / Signal Logic
Color: Orange	Risk, Filters, Regime Control
Color: Red	Execution / Broker Interaction
Color: Purple	Analytics / Reporting