Trading Control Protocol (TrCP) v1.0

A Runtime Control Layer for TradeToken-Governed Markets

Research Brief — 2025

Robert Hansen
Chief Semantic Architect
github.com/designlogic-robert

2025

Abstract

The Trading Control Protocol (TrCP) v1.0 defines a deterministic, safety-aware control layer for executing trading intentions under the Universal Semantic Token (UST) Model using Trade Tokens (TrT). Where Trade Tokens encode the semantic structure of positions, risk, and intent, TrCP governs the operational lifecycle of a trade: from initial hypothesis to order placement, monitoring, adjustment, and closure. This research brief presents the protocol's state model, invariants, governance principles, and integration points with the Universal Semantic Runtime (USR), the Universal Semantic Engine (USE), and domain-specific trading engines (FinCE). TrCP is designed to be engine-agnostic, broker-neutral, and auditable, making it suitable for both algorithmic and assisted human trading systems.

Contents

1 Introduction

Trading is an unusually dense domain: information, risk, leverage, latency, and psychology collide. Traditional algorithmic trading frameworks operate in a numerical mode — prices, indicators, signals — with only weak structural representations of trader intent or risk posture.

The Trading Control Protocol (TrCP) v1.0 addresses this gap by:

- formalizing the lifecycle of a trade as a semantic process
- ensuring that TradeToken-governed intent is honored without being exceeded
- enforcing invariants around risk, leverage, and compliance
- enabling transparent, auditable flows from idea to executed position

TrCP does not decide *what* to trade. Instead, it defines *how* trade instructions, once encoded as Trade Tokens, may safely and coherently flow through a runtime, from human or agent intent into concrete orders and back into semantic state.

2 Background

2.1 Universal Semantic Token Model and Trade Tokens

The Universal Semantic Token (UST) Model provides a typed substrate for representing structured meaning. Within that model, Trade Tokens (TrT) form a specialized family for describing:

- instruments (e.g., NASDAQ equities, futures, options)
- position structures (entry, size, stop, target, timeframe)
- risk constraints (max loss, max exposure, leverage rules)
- strategy context (trend-following, mean reversion, breakout)
- execution preferences (limit, market, partial fills, slippage bounds)

Trade Tokens are designed to be independent of specific broker APIs or platforms. They express *what* the trader or agent wants in a stable, schema-governed form.

2.2 The Role of TrCP

If Trade Tokens define the "semantic payload" of a trade, TrCP defines the *control protocol* by which:

(a) those tokens are validated and risk-checked,

(b) transformed into one or more execution plans,

(c) sequenced into broker-level orders,

(d) monitored, adjusted, and closed, and

(e) reintegrated into TradeToken-level state for journaling and learning.

TrCP lives between semantic intent (UST/TrT) and execution environments (FinCE, broker APIs, backtesting engines, simulation sandboxes).

3 Protocol Overview

3.1 High-Level Phases

TrCP is organized into six canonical phases:

P1: PREPARE — ingest, validate, and normalize Trade Tokens.

P2: EVALUATE — risk, edge, and constraint evaluation.

P3: AUTHORIZE — explicit approval under risk and governance rules.

P4: EXECUTE — order generation and broker-level submission.

P5: MONITOR — real-time tracking, adjustments, and conditional logic.

P6: SETTLE — closure, journaling, and semantic reintegration.

Each phase has clear entry and exit conditions and must satisfy specific invariants to prevent uncontrolled risk or semantic drift.

3.2 Conceptual Diagram

Conceptually, the flow can be expressed as:

```
TradeTokenSet
   -> PREPARE
   -> EVALUATE (risk, edge, constraints)
   -> AUTHORIZE (explicit or rule-based)
   -> EXECUTE (orders via FinCE/Broker)
   -> MONITOR (runtime adjustments)
   -> SETTLE (results -> TradeTokenJournal)
```

4 TradeToken Structures under TrCP

For reference, we assume a canonical Trade Token structure:

```
TRADE_TOKEN {
    INSTRUMENT: <symbol, venue, asset_class>,
    DIRECTION: LONG | SHORT,
    SIZE: <units or notional>,
    ENTRY: {
        TYPE: LIMIT | MARKET | STOP | OTHER,
        PRICE: <float or formula>,
        VALIDITY: GTC | DAY | SESSION
    },
    RISK: {
        MAX_LOSS: <currency or %>,
        STOP: <price or dynamic rule>,
        MAX_PORTFOLIO_RISK: <fraction>
    },
```

```
    TARGETS: [
        { PRICE: <float>, SIZE: <fraction> },
        ...
    ],
    TIMEFRAME: INTRADAY | SWING | POSITION,
    STRATEGY: <label or reference>,
    META: {
        TAGS: [...],
        NOTES: <string>,
        ORIGIN: HUMAN | AGENT | HYBRID
    }
}
```

TrCP does not dictate the exact schema, but assumes that Trade Tokens expose at least: instrument, direction, size, risk, and execution preferences.

5 State Machine

5.1 States

TrCP defines a finite state machine that governs the lifecycle of a TradeTokenSet:

- **S0: DRAFT** — Trade Tokens created but unvalidated.
- **S1: READY** — Tokens syntactically and semantically valid.
- **S2: RISK_CHECKED** — Risk and constraints evaluated.
- **S3: AUTHORIZED** — Explicit or policy-based approval granted.
- **S4: EXECUTING** — Orders live in the market.
- **S5: ACTIVE_POSITION** — Position exists; monitoring ongoing.
- **S6: CLOSED** — Position fully closed; P&L realized.
- **S7: JOURNALED** — Trade integrated into semantic history.

5.2 Transitions

$$\text{S0 (DRAFT)} \xrightarrow{\text{Validate}} \text{S1 (READY)}$$

$$\text{S1 (READY)} \xrightarrow{\text{Risk Engine OK}} \text{S2 (RISK\_CHECKED)}$$

$$\text{S2 (RISK\_CHECKED)} \xrightarrow{\text{User/Policy Approval}} \text{S3 (AUTHORIZED)}$$

$$\text{S3 (AUTHORIZED)} \xrightarrow{\text{Order Sent}} \text{S4 (EXECUTING)}$$

$$\text{S4 (EXECUTING)} \xrightarrow{\text{Order Filled}} \text{S5 (ACTIVE\_POSITION)}$$

$$\text{S5 (ACTIVE\_POSITION)} \xrightarrow{\text{Exit Conditions Met}} \text{S6 (CLOSED)}$$

$$\text{S6 (CLOSED)} \xrightarrow{\text{Journal Commit}} \text{S7 (JOURNALED)}$$

Rejections, cancellations, or risk violations cause transitions to fallback or terminal error states (not detailed here but recommended in implementations).

6 Invariants

TrCP enforces a set of invariants to prevent misuse and uncontrolled risk.

6.1 I1: No Execution without Authorization

No trade may move into EXECUTING state without passing through:

$$\text{READY} \rightarrow \text{RISK\_CHECKED} \rightarrow \text{AUTHORIZED}$$

Any bypass of this path is an invalid protocol violation.

6.2 I2: Risk Budget Integrity

Let R_{max} be the configured maximum portfolio risk and R_{new} the incremental risk of the proposed trade. Let $R_{current}$ be the risk of existing open positions.

TrCP requires:

$$R_{current} + R_{new} \leq R_{max}$$

If this inequality fails, the trade cannot be authorized unless other positions are reduced or closed.

6.3 I3: Leverage and Instrument Constraints

Each instrument class has associated leverage and regulatory bounds. TrCP must ensure that the effective leverage implied by SIZE, STOP, and account capital does not exceed configured or legal maxima.

6.4 I4: Semantic Consistency

The TradeTokenSet must remain semantically consistent before and after execution. For example:

- Position direction must match net executed exposure.
- Risk fields must accurately reflect actual stops and targets.
- Timeframe tags must not contradict holding behavior.

6.5 I5: Journaling Completeness

Every finalized trade must eventually reach JOURNALED state. Trades that close without journaling are protocol violations and should trigger alerts.

7 Integration with USR and USE

7.1 USR (Universal Semantic Runtime)

USR provides the semantic operating environment in which Trade Tokens and TrCP live. For TrCP, USR is responsible for:

- validating token types and field schemas
- enforcing the invariants specified above
- routing TrCP operations to appropriate engines (FinCE, backtester)
- logging semantic events for audit and learning

TrCP is implemented as a protocol module within USR, with well-defined input (TradeTokenSet plus context) and output (state transitions plus execution instructions).

7.2 USE (Universal Semantic Engine)

USE operationalizes semantic structures into concrete actions. In the context of TrCP, USE:

- transforms validated Trade Tokens into execution plans
- decomposes plans into atomic broker operations (orders, modifies, cancels)
- sequences monitoring rules and conditional adjustments
- generates semantic deltas (e.g., risk used, realized P&L) back into UST

TrCP relies on USE for the "how" of execution, while TrCP itself governs the "whether" and "when".

8 FinCE and Broker Adapters

8.1 FinCE as Trading Engine

FinCE, as a dedicated cognitive engine for financial markets, is a natural host for TrCP. Inside FinCE:

- Trade Tokens express structured trading intent.
- TrCP manages the lifecycle of those tokens.
- Broker adapters map execution plans to specific broker APIs.

8.2 Broker Neutrality

TrCP is explicitly broker-neutral. All broker-specific concerns (order syntax, API quirks, rate limits) are encapsulated in adapter layers that:

- translate USE-level execution primitives into broker calls
- normalize broker responses into semantic events

This allows TrCP and Trade Tokens to remain stable even if execution venues change.

9 Failure Modes and Safeguards

9.1 FM1: Partial Execution

Orders may fill partially or across multiple venues. TrCP must treat the position as ACTIVE_POSITION only to the extent of filled size and must preserve invariant consistency with the fractional exposure.

9.2 FM2: Slippage and Gaps

Markets may gap, causing stop orders to fill worse than expected. TrCP should:

- record actual fill prices and update realized risk
- trigger post-trade risk review if slippage exceeds thresholds

9.3 FM3: Connectivity Loss

If connectivity to broker or data feeds is lost, TrCP should:

- freeze new AUTHORIZE $\rightarrow$EXECUTE transitions
- flag existing EXECUTING/ACTIVE_POSITION states for manual review

9.4 FM4: Protocol Violations

Any attempt to bypass phases, ignore risk constraints, or execute trades without journaling should be logged as a critical protocol violation, with appropriate alerts to operators or supervisory systems.

10 Human-in-the-Loop vs Fully Automated Modes

TrCP is designed to support a spectrum of autonomy:

10.1 Assisted Mode

- Humans specify Trade Tokens.
- TrCP performs risk checks and recommends actions.
- Humans explicitly approve AUTHORIZE transitions.

10.2 Hybrid Mode

- Agents propose Trade Tokens under policy bounds.
- Humans review batches or thresholds.

10.3 Automated Mode

- Agents generate and authorize trades under strict policy constraints.
- TrCP still enforces invariants and journaling.

In all cases, TrCP ensures that the same state machine and invariants apply.

11 Example Flow

A simplified example:

1. A strategy engine proposes a LONG trade on a NASDAQ stock, with defined stop and target.
2. Trade Token is created (S0: DRAFT).
3. USR validates schema and semantics (S1: READY).
4. Risk engine confirms total portfolio risk remains within bounds (S2: RISK_CHECKED).
5. The trader reviews and approves the trade (S3: AUTHORIZED).
6. USE generates limit orders and submits via FinCE to the broker (S4: EXECUTING).
7. Orders fill, position becomes live (S5: ACTIVE_POSITION).
8. Price hits target, exit order executes; position closes (S6: CLOSED).
9. Trade is encoded into a journal entry and appended to TradeToken history (S7: JOURNALED).

This flow demonstrates how semantic intent, risk integrity, and execution interlock under TrCP.

12 Relationship to Other Protocols

12.1 SCP vs TrCP

The Semantic Control Protocol (SCP) governs semantic reasoning more broadly: question answering, planning, multi-step workflows across arbitrary domains. TrCP specializes in a single high-risk vertical: trading.

- SCP asks: "Which sequence of semantic operations should we run?"
- TrCP asks: "Which of these trades, if any, should be allowed into the market, and under what constraints?"

12.2 Teleo Control Protocol vs TrCP

The Teleo Control Protocol (TCP) focuses on teleogenic arcs in narrative or psychological contexts. TrCP can be viewed as its financial sibling: it applies similar control-layer ideas to capital, risk, and markets.

13 Conclusion

The Trading Control Protocol (TrCP) v1.0 provides a structured, invariant-driven control layer for TradeToken-based trading systems. By formalizing the trade lifecycle as a semantic state machine — from DRAFT to JOURNALED — and integrating tightly with USR, USE, and FinCE, TrCP offers:

- broker-neutral, engine-agnostic trading control,
- rigorously bounded risk and leverage,
- transparent, auditable execution flows,

- compatibility with human, hybrid, and automated operation.

As semantic trading architectures mature, TrCP can serve as a foundational runtime standard for safe, interpretable, and semantically coherent market interaction.