Tone Vector System (TVS)

A Modular Representation Layer for Interoperable Signals

Stephen A. Putman
putmanmodel@pm.me

Version 0.2 — February 2026

Abstract

The Tone Vector System (TVS) is a modular representation layer for standardizing how stratified agent architectures encode, persist, and exchange interaction-derived state. TVS specifies a typed tagging grammar that decomposes representation into (i) a domain identifier, (ii) a routing/transform identifier, and (iii) a normalized payload, enabling heterogeneous modules to interoperate without shared sensor identity or a shared internal model. TVS is not a prerequisite for constraint-grounded inference, memory stratification, field dynamics, or deviation governance; it is an optional abstraction layer that standardizes representation across those systems when interoperability is desired.

The specification introduces commit-class semantics to distinguish canonical (committed) tags from associative (residual) traces, supporting bounded persistence and decay without contaminating shared ontologies. Memory writes retain provenance and rationale pointers sufficient for replay and audit ("why was this stored?"), aligning representation with governance and falsifiability requirements. TVS constrains representation contracts and versioning while leaving domain-specific extractors, cross-domain transforms, and downstream policies non-normative. The result is a disciplined interface layer that improves composability, auditability, and legibility of emergent behavior in simulation-ready agent stacks.

1. Introduction

Artificial agent systems increasingly require modularity: not merely modular code, but modular representation. When components do not share a stable coordinate system, integration pressure accumulates as ad hoc adapters, hidden assumptions, and un-auditable state translation. TVS is proposed as a representation interface that standardizes how modules encode state into a typed, versioned tag form derived from constraint-accessible interaction.

TVS is explicitly non-foundational. It does not replace constraint-grounded inference, memory stratification, field coupling, or deviation governance; nor does it prescribe a single emotional schema, perceptual ontology, or model family. Instead, it defines a representation contract that can be adopted where interoperability is required and omitted where custom signals are sufficient. This prevents dependency creep: modules remain standalone, while TVS provides a shared coordinate option when cross-module coupling, cross-agent exchange, or auditability constraints make bespoke formats brittle.

A second motivation is persistence discipline. In stratified memory systems, not all stored traces should remain equally "committed." TVS introduces commit-class semantics that distinguish canonical tags (stable, governance-relevant representations) from associative traces (residual, decaying handles used for internal recall). This allows systems to preserve lifelike continuity—retaining what "felt" salient—while preventing uncontrolled assimilation into shared baselines or long-horizon priors. When combined with provenance and rationale pointers, TVS supports a simple but critical question for simulation architectures: why did this agent log this to memory?

2. Terminology and Operational Definitions

All terms are used in an operational and architectural sense.

Tag

A compact, typed representation emitted by a module to encode interaction-derived state in a form suitable for exchange, persistence, and governance. A tag is treated as a derived observable with provenance and confidence, not as ground truth.

Typed tagging grammar

A representation scheme that decomposes a tag into declared components: a domain identifier, a routing/transform identifier, and a payload. The grammar constrains structure and versioning; domain-specific meaning is defined by declared dictionaries and manifests.

Domain identifier

A short alphanumeric code that selects the constraint-accessible feature family being represented (e.g., language, proximity, audio). The domain identifier must resolve to a declared domain dictionary version when a tag is treated as canonical.

Routing/transform identifier

A short alphanumeric code that denotes the transform family or usage context under which a payload is interpreted (e.g., encoding, propagation, storage, translation). Routing identifiers are declared and versioned to prevent implicit cross-module interpretation.

Payload

A domain-defined value (typically numeric and normalized) representing state within the selected domain under the specified routing context. Payload semantics are constrained by the domain dictionary and must be specified by declared dictionaries (range/normalization/units) rather than by hidden model internals.

Canonical tag

A committed representation suitable for cross-module interoperability, governance evaluation, and long-horizon persistence. Canonical tags resolve against declared domain and routing dictionaries and carry sufficient provenance to support replay and audit. Canonical tags may be used as between-session reference anchors when continuity across sessions is required.

Associative tag

A residual trace used for internal recall, similarity search, or “felt” continuity that is not treated as governance-relevant ground truth. Associative tags may be retained under short-horizon memory and may be demoted from canonical form under decay without implying a semantic change.

Commit class

A retention and governance status assigned to a tag (e.g., canonical vs associative). Commit class is an architectural control to prevent ontology contamination and to regulate persistence.

Implementations may encode commit class explicitly as metadata or implicitly (e.g., by case), provided the mapping is declared and auditable.

Promotion and demotion

Governed transitions between commit classes. Promotion (associative → canonical) requires an explicit pathway (e.g., manifest versioning, review, or threshold-gated consolidation). Demotion (canonical → associative) may occur under decay or stability criteria and must be logged with rationale.

Provenance

The minimal metadata required to attribute a tag to a producing module and configuration state, including producer identifier/version, dictionary versions, and timestamp. Provenance is required for replayability and audit.

Confidence

A bounded score or regime indicating the reliability of a tag emission given the available constraint-accessible evidence. Confidence is used for routing and memory eligibility and must be logged when tags contribute to governance decisions or memory writes.

Rationale pointer

A reference linking a tag emission or memory write to the triggering evidence used by the producing subsystem (e.g., a deviation event id, evidence spans, or an audit record). Rationale pointers support the system-level question: why was this stored?

3. Representation Contract

TVS specifies a typed tagging grammar that standardizes how interaction-derived state is encoded for interoperability. The grammar constrains structure, versioning, and minimal metadata, while leaving domain-specific extraction and downstream policies non-normative.

A TVS tag is decomposed into three declared components: a domain identifier, a routing/transform identifier, and a payload. Domain and routing identifiers are short alphanumeric codes that resolve against versioned dictionaries; payload semantics are domain-defined and must be specified by declared dictionaries (range/normalization/units) rather than by hidden model internals. This separation prevents implicit cross-module interpretation and enables controlled extensibility across modalities.

In canonical form, tags must carry sufficient provenance to support replay and audit. At minimum this includes producer identifier/version, domain dictionary version, routing dictionary version, and timestamp. Implementations may attach additional metadata (e.g., confidence, rationale pointers) without changing the tag grammar.

A canonical tag grammar may be represented as:

- Domain identifier: XX (alphanumeric; fixed-length or bounded-length by manifest)
- Routing identifier: YY (alphanumeric; fixed-length or bounded-length by manifest)
- Payload: Z... (domain-defined; typically numeric and normalized)

TVS does not require a single global dimensionality. Domains may define scalar payloads, low-dimensional vectors, or structured payloads, provided their normalization and interpretability constraints are declared in the domain dictionary and are versioned.

To preserve non-omniscient operation, tags are treated as derived observables: their validity depends on constraint-accessible inputs, declared extraction pathways, and logged provenance. Tags may be emitted by deterministic extractors or learned taggers, but in all cases they must be attributable, confidence-scored, and governed by the receiving subsystem when used for persistence, coupling, or control.

Finally, TVS constrains how tags evolve over time. Domain dictionaries, routing dictionaries, and any normalization parameters must be versioned; changes are treated as representational updates rather than silent semantic drift. This ensures that a tag's meaning is stable under a given manifest and that cross-session continuity can be maintained when required. Minimal compliance requirements are listed in Appendix A.

4. Commit-Class Semantics (Canonical vs Associative)

TVS distinguishes representation from retention. A tag can be valid as a representation while differing in how strongly the system commits to it over time or allows it to influence downstream governance. TVS therefore defines a commit-class distinction between canonical tags and associative tags.

Canonical tags are committed representations intended for interoperability, governance evaluation, and longer-horizon persistence. They resolve against declared domain and routing dictionaries, carry versioned provenance, and may be used as stable anchors when continuity across sessions is required. Because canonical tags can affect cross-module behavior, they are governed by the receiving subsystem's policies for update, quarantine, and audit.

Associative tags are residual traces used for internal recall, similarity search, and continuity of "felt" state without asserting a governance-relevant truth claim. Associative tags are permitted to decay and to be retained under short-horizon memory without being promoted into shared baselines or long-horizon priors. The intent is to preserve lifelike continuity while preventing uncontrolled assimilation of provisional signals into system-wide reference models.

Commit class is a retention and governance status, not a change in semantic content. Implementations may encode commit class explicitly (as metadata) or implicitly (e.g., via case or a parallel identifier), provided the mapping is declared and auditable. In either representation, canonical and associative forms remain linkable so that internal recall can be audited against the canonical record when needed.

Transitions between commit classes are governed. Promotion (associative → canonical) requires an explicit pathway—such as threshold-gated consolidation, manifest versioning, or review—and must be logged with rationale and provenance. Demotion (canonical → associative) may occur under decay or stability criteria and must be logged as a governed transition rather than occurring silently. This allows the system to maintain bounded persistence while preserving auditability of representational drift over time.

5. Interoperability with the Spanda Series

TVS is designed to interoperate with the Spanda architectural papers without creating new prerequisites. The purpose of this section is to specify interface alignment: how TVS can be adopted as a shared representation layer while preserving each module's independence.

Constraint-grounded inference (Paper 1) limits what an agent may infer to what its constraint surfaces justify. TVS aligns with this boundary by treating tags as derived observables that must be attributable to constraint-accessible inputs and declared extraction pathways. TVS does not expand the epistemic surface; it standardizes how constraint-accessible signals are represented once derived.

Memory stratification (Paper 2) formalizes persistence as a tiered architecture with gated assimilation and regulated write resistance. TVS supports this by providing a compact, versioned payload geometry that can be stored at different tiers under explicit commit-class semantics. Associative tags are compatible with short-horizon volatility, while canonical tags can serve as consolidation candidates when a tier's gating criteria are satisfied. TVS does not define tier policy; it defines the representation that tier policies may store and audit.

Reference-shift field dynamics (Paper 3) describe multi-agent instability as a structural phenomenon driven by reference alignment and cascade conditions. TVS can serve as a common coupling variable in such systems by standardizing the state representations that agents propagate, absorb, or compare, without requiring shared sensor identity. Field dynamics remain free to propagate scalars, custom objects, or other summaries; TVS is an optional coordinate interface when cross-agent interoperability is desired.

Constraint deviation governance (Paper 4) detects drift as enforceable deviation relative to stratified baselines and routes events into governed action. TVS may optionally provide a coordinate space in which deviation is computed, especially when multiple upstream modules emit heterogeneous signals. In such cases, CDE treats tag trajectories as just another constraint-accessible abstraction stream, with provenance, confidence, and auditability preserved by TVS

contracts. CDE remains standalone: it can compute deviation over raw or custom features without TVS, while TVS improves standardization when adopted.

6. Memory Record Contract (Minimal)

TVS does not define a memory system, but it constrains the minimum record fields needed for tags to support persistence, auditability, and controlled recall. When tags are written to memory —at any tier—the system should retain enough information to answer two operational questions: what was stored, and why was it stored.

A minimal TVS-aligned memory record includes a canonical tag (when available) and may include a correlated associative tag for internal recall. Each record carries provenance sufficient for replay, including producer identifier/version and the relevant dictionary versions. When a write is triggered by a governance event or a threshold gate, the record should include a rationale pointer linking the write decision to its triggering evidence.

At minimum, the record contains:

- a tag value (canonical and/or associative form)
- commit class (canonical or associative)
- timestamp
- producer identifier/version
- domain dictionary version and routing dictionary version

When tags are used for governed persistence or downstream control, the record additionally contains:

- confidence (tag-level or regime)
- rationale pointer (e.g., event id or audit reference)

This contract is intentionally minimal. Implementations may attach richer structures (e.g., per-axis payload confidence, demotion/promotion reasons, or cross-tag associations), provided the base fields remain stable and versioned so that replay and audit are possible.

7. Mapping Interfaces (Non-normative Families)

TVS standardizes representation, not extraction. Domain-specific mappings from raw observables into tags are therefore treated as non-normative families: multiple implementations may satisfy the TVS contract provided their outputs are constraint-accessible, versioned, and auditable.

In language-facing systems, tags may be produced from lexical markers, pragmatic cues, bounded similarity measures, or learned taggers operating over the interaction payload. The resulting tags can be stored as associative traces for short-horizon continuity or promoted to canonical form when governance or memory gates justify consolidation.

In simulation and field-coupled systems, tags may be emitted from agent state summaries, interaction events, proximity/contact signals, or other constraint-accessible state variables provided to the agent through its interface. Here TVS functions as a common coordinate interface: heterogeneous subsystems can exchange tagged state without adopting a shared internal representation.

Cross-domain transforms are permitted but governed. A routing identifier may declare an admissible transform family that maps one domain payload into another domain payload, but such transforms must be explicitly declared in the routing dictionary and retain provenance. TVS does not treat cross-domain transforms as implicit inference; it treats them as versioned operations whose validity is determined by the receiving subsystem's governance rules.

Finally, TVS supports mixed adoption. A system may standardize only a subset of domains or routes while leaving others as bespoke internal features. This preserves modularity: TVS

improves interoperability where adopted, without constraining architectures that do not require cross-module standardization.

8. Falsifiability and Limitations

TVS is an architectural specification and is falsifiable at the level of interface behavior and replayability rather than at the level of a single model's predictive performance. The following tests are sufficient to challenge whether a given TVS adoption satisfies the specification.

First, tag determinism and replay must hold at the contract level: given the same constraint-accessible inputs, the same extractor version, and the same dictionary versions, a system must be able to reproduce the same canonical tag outputs within declared confidence tolerances. Where stochastic taggers are used, replay must be possible via logged seeds and immutable artifact references (extractor version and model artifact id).

Second, versioning integrity must be enforced: a tag's meaning must not silently drift under dictionary changes. Dictionary version updates must be explicit, and any cross-session continuity claims must specify the versions under which continuity is asserted. A system that changes tag semantics without versioning violates the specification.

Third, commit-class transitions must be governed and auditable. Promotion from associative to canonical form must occur only through explicit pathways, and demotion from canonical to associative form must be logged as a governed transition. Silent promotion or silent demotion breaks auditability and undermines the retention discipline that commit classes are intended to provide.

Fourth, missing-domain and unknown-route handling must be bounded. If a module receives a tag whose domain or route does not resolve against the declared dictionaries, the system must treat it as non-canonical and prevent it from affecting governed persistence or baseline updates unless a declared policy explicitly permits it. This prevents uncontrolled ontology growth.

These tests define the limits of what TVS claims. TVS does not claim that any particular domain dictionary is "correct," that any tagger is optimal, or that cross-domain transforms are valid in general. It claims that when representations are emitted, they can be exchanged, persisted, replayed, and audited under declared contracts.

9. Conclusion

TVS defines a modular representation layer for stratified agent architectures by standardizing typed tagging, provenance, and commit-class semantics. It is not a prerequisite for constraint-grounded inference, memory stratification, field dynamics, or deviation governance; it is an optional abstraction layer that improves interoperability when systems require shared coordinate interfaces across modules, agents, or sessions.

By distinguishing canonical tags from associative traces and by requiring governed promotion and logged demotion, TVS supports bounded persistence without contaminating shared ontologies. By requiring versioned dictionaries and replayable provenance, it makes emergent behavior more legible and memory decisions more auditable. TVS therefore functions as a disciplined interface contract: representation standardization where desired, without imposing a universal substrate claim or an implicit expansion of an agent's constraint surface.

Appendix A — Minimal Compliance Checklist (Normative)

A TVS implementation is compliant if it satisfies the following minimal contracts:

1. Tag structure and provenance

Each canonical tag resolves against versioned domain and routing dictionaries and is accompanied by: producer_id, producer_version, domain_dict_version, route_dict_version, timestamp.

2. Payload interpretability

Each domain definition specifies: payload range, normalization function identifier (or description), and units (or explicit unitless). Each payload references a schema_id (human- or machine-readable).

3. Confidence

If confidence is emitted, it uses a numeric value in [0,1] or a regime that maps deterministically to [0,1] thresholds declared in the manifest.

4. Commit-class transitions

Promotion (associative → canonical) and demotion (canonical → associative) are logged as explicit transitions with timestamp and rationale pointer. Silent promotion/demotion is non-compliant.

5. Replayability

Deterministic taggers must be reproducible given identical inputs and versions. If stochastic tagging is used, the following must be logged: seed, extractor_version, and model_artifact_id (or an equivalent immutable artifact reference).

6. Dictionary compatibility policy

Domain and routing dictionaries declare a compatibility policy using semantic versioning: MAJOR changes are breaking, MINOR changes are additive, PATCH changes are editorial. Receivers treat unknown MAJOR versions as non-canonical unless explicitly permitted by policy.

7. Cross-domain transforms (if used)

Any routing-defined cross-domain transform declares: transform_id, transform_version, and parameter_hash (or “none”). Optional: validation_note or test_id.