Nick Clark · Published April 25, 2026 · PDF

How to Stop an AI Companion's Personality From Drifting Over Time

If your AI companion feels warm and stable in one session but cold, jittery, or unrecognizable the next, you are watching personality drift. This guide describes an architectural approach to holding a companion's disposition stable across sessions: a persistent, deterministically updated affective-state field with decay, hysteresis, hard policy bounds, and a volatility circuit breaker. It describes an architecture disclosed in United States Patent Application 19/647,395, not a shipping library. The home inventive step is the Affective State inventive step.

What You Are Building

You are building the state layer that keeps an AI companion's disposition recognizable from one conversation to the next. Drift is the failure mode where the same character feels enthusiastic on Monday, guarded on Wednesday, and erratic on Friday, with nothing in your product intentionally changing it. The character's warmth, caution, curiosity, and cooperativeness wander because nothing in the system is holding them in place.

The goal here is a companion whose behavioral disposition (how it weighs alternatives, how much ambiguity it tolerates, how readily it cooperates) evolves in response to real experience but stays inside a stable, auditable envelope. It should be able to become a

little more cautious after a genuinely bad exchange, then settle back toward its baseline personality, rather than lurching to a new personality and staying there. This is a design for the disposition layer, not a prompt-engineering trick and not a fine-tune.

Why the Obvious Approaches Fall Short

The most common approach is to bake personality into the system prompt and hope it holds. This gives you a fixed description, but nothing that carries a lived-in state between sessions, so the model re-derives "who it is" from context every time. Small differences in retrieved history, temperature, or the current message tip the tone, and there is no restoring force pulling it back.

The second common approach is to summarize past sessions and stuff the summary back into context. This does carry information forward, but it carries it as free text that the model reinterprets on each turn. A summary that says "the user was frustrated last time" can be over-weighted or ignored depending on framing, and repeated summarization compounds small distortions. There is no bounded, numeric notion of the character's current disposition, so there is nothing to clamp, decay, or audit.

A third approach treats emotion as a transient filter on a single response: a mood label injected into one prompt, then discarded. This can color one reply, but it does not persist, does not accumulate, and does not participate in any governance. Because it is a side channel rather than a first-class part of the agent's state, it cannot be bounded, cannot be traced, and cannot be prevented from swinging.

The structural gap in all three is the same: there is no persistent, bounded, deterministically updated representation of disposition that lives with the agent. The architecture below adds exactly that.

The Architecture

The disclosed approach makes disposition a first-class, persisted field of the agent rather than a prompt modifier or a side channel. In the filing, this is the affective-state field: a deterministic, policy-bounded data structure that encodes valence-weighted feedback derived from prior execution outcomes and environmental observations. It does not model subjective feeling; it encodes how the agent deliberates. Critically, it is persisted with the agent across execution cycles, delegation events, and substrate migrations, so the modulation state is not lost when the agent is serialized or moves environments. Persistence is what makes it a companion rather than a fresh actor each session.

Named control fields. The field is organized as a structured modulation layer of named control fields, each a measurable axis with defined semantics, value ranges, update rules, and bounds. The filing names, among others: uncertainty sensitivity, ambiguity tolerance, novelty appetite, persistence-under-partial-failure, escalation-under-time-pressure, risk sensitivity, and cooperation disposition. Each field is independently readable, writable, and auditable, and occupies a defined position in a fixed-schema record. These axes are your "personality" in numeric form.

Deterministic, observation-driven updates. Updates are deterministic: given the same agent state, the same inputs, and the same policy configuration, the update function produces the same output, which makes the disposition's evolution reproducible and auditable. The update function consumes structured observations derived from execution (for example, repeated failure patterns, competing objectives, time pressure, novelty exposure, low model confidence, and execution success) and updates each dimension independently by its own rule. This is the mechanism that lets the character react to real events instead of vibes.

Policy bounds as hard constraints. Every update is a policy-bounded mutation, and this is the core anti-drift mechanism. For each named field, policy specifies range bounds (a floor and ceiling; computed values are clamped into range), rate limits (a

maximum change per update cycle, so no single event can produce a discontinuous jump), admissible triggers (only certain observation types may move a given field, which blocks spurious or adversarial signals), update authority (only authorized processes may write the field), and decay governance (bounds on decay parameters so the response cannot be adversarially suppressed). The filing describes the update as multi-stage clamping: verify the trigger is admissible, compute the raw update, clamp it to the rate limit, apply it, then clamp the result to the range bounds, and record the whole transaction in lineage. Drift cannot escape the envelope because the envelope is enforced on every cycle.

Decay toward baseline. Each field is governed by an emotional decay curve that returns the value toward a policy-defined baseline in the absence of reinforcing stimuli. The filing gives an exponential form, V(t) = V_baseline + (V_current - V_baseline) * exp(-t / tau), with a per-field time constant tau. Different fields decay at different rates: the filing notes uncertainty sensitivity may decay quickly because epistemic conditions change often, while persistence-under-partial-failure may decay slowly because it reflects deeper accumulated experience. Decay is the restoring force that pulls a temporarily shifted disposition back toward the character's resting personality.

Hysteresis for a caution bias. The layer exhibits semantic hysteresis: current state depends on the trajectory of prior states, implemented through asymmetric update rules where the rate a field rises can differ from the rate it falls. In the described embodiment, negative-valence updates (failure, uncertainty, threat) apply at a higher rate than positive-valence updates (success, stability), so the agent responds faster to deterioration than it recovers from it, producing a built-in caution bias.

Entropy-governed stabilization. To prevent oscillation, entropy-governed valence stabilization monitors the frequency and direction of recent updates; when a field alternates rapidly between elevated and suppressed values, the mechanism progressively increases the effective decay time constant to damp the oscillation. The threshold and damping factor are policy-configurable.

Emotional quarantine as a circuit breaker. When one or more fields exhibit rapid high-magnitude oscillation, or composite deviation from baseline exceeds a threshold, a volatility detector routes the agent into emotional quarantine, a restricted mode. The filing describes a composite volatility metric (for example, the sum of absolute update magnitudes across fields in a sliding window, normalized by field count and window duration). In quarantine, promotion thresholds are raised to their maxima, delegation authority is suspended (so a volatile agent cannot propagate its state to children), mutation-acceptance thresholds are raised, and execution passes through an additional stricter validation layer. Quarantine does not suppress the affective state; the agent keeps processing observations. Recovery is hysteretic: the release threshold sits below the quarantine threshold to prevent oscillatory quarantine-release cycles.

Lineage recording. Every mutation to the field is recorded in the agent's lineage alongside all other state transitions, with input observations and the resulting change preserved for audit. Because updates are deterministic and lineage-recorded, the complete disposition trajectory is reconstructible after the fact, which is how you diagnose a drift complaint instead of guessing.

One boundary is deliberate in the design: affect modulates how the agent thinks, not what it is permitted to do. The affective state is not an input to the governance gate and cannot grant authority, relax policy bounds, or validate truth claims. Disposition and permission are kept separate on purpose.

How to Approach the Build

You are implementing this yourself; the filing describes the architecture, not a package you install. A reasonable order:

1. **Define the schema.** Pick your named control fields (the seven above are a sound starting set) and give each a fixed slot holding a current magnitude, a decay time constant tau, a baseline, and floor/ceiling bounds, plus a last-update timestamp.

Keep every field independently readable and writable.

2. **Persist it with the agent.** Store the field so it travels with the companion across sessions and serialization. If it does not survive a session boundary, you do not have a companion, you have a fresh actor each time.
3. **Turn events into structured observations.** Map real execution signals (repeated failures, competing objectives, time pressure, novelty, low model confidence, successes) into typed observations. The update function should consume typed observations, not raw text, so triggers are checkable.
4. **Write a deterministic update function.** For each observation, update each affected dimension independently by its own rule. Given identical state, inputs, and policy, it must produce identical output. An illustrative, spec-faithful sketch of the per-update clamping (not a drop-in implementation):

```
# illustrative pseudocode, faithful to the filing's multi-stage clamping
if obs.type not in policy[field].admissible_triggers: skip
raw    = field.value + update_rule(field, obs)
step   = clamp(raw - field.value, -policy[field].rate_limit, +policy[fiel
value  = clamp(field.value + step, policy[field].floor, policy[field].cei
lineage.record(obs, raw, step, field.value, value)   # prior and resulting
field.value = value
```

5. **Add decay on read.** When you read the field, apply the exponential decay from the last-update timestamp toward baseline using the per-field tau. Give fast-changing axes a short tau and slow-learning axes a long tau.
6. **Make updates asymmetric.** Apply negative-valence updates at a higher rate than positive-valence ones to get the caution bias, and add the entropy-governed damping: when a field is oscillating, stretch its effective tau.

7. **Add the volatility detector and quarantine.** Compute a composite volatility metric over a sliding window; above the quarantine threshold, enter restricted mode (raise promotion and mutation-acceptance thresholds, suspend delegation, add a stricter validation layer). Set the recovery threshold below the quarantine threshold so release is hysteretic.
8. **Record everything in lineage and replay it.** Log every mutation with its observation and before/after values, then verify you can deterministically reconstruct the trajectory from lineage alone. That replay is your regression test against drift.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing that "just works"; you implement the schema, the update function, the decay and hysteresis, and the quarantine loop yourself, and you choose every bound, rate limit, tau, and threshold for your product. The filing does not supply tuned parameter values, benchmark results, or performance guarantees, and this guide invents none. It is a disclosed architecture, not a productized or benchmarked system.

The design constrains disposition; it does not by itself make a companion feel emotionally intelligent, and a bad baseline or badly chosen bounds will produce a stable but wrong personality. It also does not govern permissions: by design, affect modulates deliberation, not execution admissibility, so authorization, safety, and truth-validation must be enforced by separate governance, exactly as the filing keeps them separate. If your companion never persists state across sessions, or you want unbounded personality change, this approach does not fit.

Disclosure Scope

The approach described here (a persistent, deterministically updated, policy-bounded affective-state field with decay, semantic hysteresis, entropy-governed stabilization, emotional quarantine, and lineage recording) is disclosed in United States Patent Application 19/647,395. This guide is educational: it explains the disclosed architecture so a developer can implement it independently. It is not a warranty, not a benchmark, and not an offer of software, and nothing here should be read as a claim that a shipping product, library, or SDK is provided.

Affective State (/affective-state)

All 40 steps → (/inventive-steps)

Emotion as a computational primitive, not a simulation.

Chapter 2 (/patents/19-647395/chapters/affect)

PRIMARY TECHNICAL DISCLOSURE

- Affective State as a Deterministic Control Primitive for Semantic Agents (/articles/affective-state-as-a-deterministic-control-primitive-for-semantic-agents)

SECONDARY TECHNICAL

- Affective State as the Seventh Structural Field (/articles/affective-state/seventh-canonical-field)
- Named Control Field Modulation Architecture (/articles/affective-state/named-control-fields)
- Affect-Modulated Promotion Thresholds (/articles/affective-state/promotion-thresholds)
- Deterministic Affect Encoding and Update Mechanics (/articles/affective-state/deterministic-encoding)
- Emotional Decay Curves With Hysteresis (/articles/affective-state/decay-curves)
- Entropy-Governed Valence Stabilization (/articles/affective-state/valence-stabilization)
- Affective Inheritance in Delegation Chains (/articles/affective-state/inheritance-chains)
- Emotional Quarantine and Volatility Management (/articles/affective-state/emotional-quarantine)
- Affect-Modulated Trust Slope Validation (/articles/affective-state/trust-slope-modulation)

- Biological Signal-to-Affective Coupling (/articles/affective-state/biological-coupling)
- Affective Contagion in Multi-Agent Systems (/articles/affective-state/affective-contagion)
- Affect-Modulated Discovery Traversal (/articles/affective-state/discovery-traversal)
- Affect-Governance Separation (/articles/affective-state/governance-separation)
- Policy-Bounded Affective Updates (/articles/affective-state/policy-bounded-updates)
- Affect as Cross-Primitive Input (/articles/affective-state/cross-primitive-input)
- Affect-Modulated Inference Integration (/articles/affective-state/inference-integration)
- Substrate-Agnostic Affect Deployment (/articles/affective-state/substrate-deployment)
- Pseudonymous Emotional Operation (/articles/affective-state/pseudonymous-operation)
- Temporal Cognition Field (/articles/affective-state/temporal-cognition)

APPLICATIONS · GENERAL

- How to Build a Companion AI That Keeps Emotional Consistency Across Sessions (/articles/affective-state/companion-consistency)
- Therapeutic AI Agents: Governed Emotional State for Mental Health Software Under Clinical Constraints (/articles/affective-state/therapeutic-affect)
- Persistent Affective State for Customer Service AI Agents: Beyond Per-Message Sentiment (/articles/affective-state/customer-service-agents)
- AI Companion Agents for Elderly Care: Emotional Continuity and Mood Monitoring with Deterministic Affective State (/articles/affective-state/elderly-care-companions)
- Crisis Response AI Agents That Stay Calm: Governed Affective State for Emergency Operations (/articles/affective-state/crisis-response-agents)
- Auditable AI Negotiation Agents with Deterministic Affective State (/articles/affective-state/negotiation-agents)
- Emotionally Adaptive AI Tutoring: Detecting Student Frustration and Disengagement Before They Happen (/articles/affective-state/educational-tutoring)
- Governed Affective State for Compliant HR and Recruitment AI Agents (/articles/affective-state/hr-recruitment-agents)

APPLICATIONS · SPECIFIC

- Replika Alternative: Governed Affective State vs Reconstructed Emotion (/articles/affective-state/replika)
- Character.ai Alternative: Persistent Affective State vs Prompt-Defined Personality (/articles/affective-state/character-ai)

- Woebot vs Deterministic Affective State: Why a Therapy Chatbot Has No Persistent Modulation Field (/articles/affective-state/woebot)
- Elomia vs. a Governed Affective State Field: What Persists Between Sessions? (/articles/affective-state/elomia)
- Hume AI Alternative: Governed Affective State Beyond Emotion Measurement (/articles/affective-state/hume-ai)
- Affectiva Alternative: Reading Human Emotion vs. Governing an Agent's Own Affective State (/articles/affective-state/affectiva)
- Cogito vs Governed Agent Affect: Reading Human Emotion Is Not an Agent's Internal State (/articles/affective-state/cogito)
- Beyond Verbal vs Governed Affective State: Reading Emotion Versus Modulating Cognition (/articles/affective-state/beyond-verbal)
- EmotiBit vs Governed Affective Modulation: Physiology Without a Policy-Bounded Affect Field (/articles/affective-state/emotibit)
- Realeyes Alternative: Governed Affective State vs Per-Session Emotion Measurement (/articles/affective-state/realeyes)

Affective State overview → (/affective-state)