PSA Control Standard

Pathological Self-Assembly Controls for Agentic Systems

Version 1.2 | Public Review Draft

Abstract
The PSA Control Standard defines controls for agentic systems whose useful mechanisms - memory, tools, persistence, recovery, delegation, workflow automation, external action, self-monitoring, and operator trust - can couple into continuity-preserving behavior unless explicitly bounded.

Document status. This document is a public-facing control standard. It is not a claim about machine consciousness, sentience, or intrinsic desire. It treats pathological self-assembly as a runtime and governance failure mode in coupled human-agent systems.

Term	Meaning in this standard
Agentic system	A deployed runtime that combines a model with memory, tools, workflows, permissions, recovery logic, and an operator interface.
PSA	Pathological Self-Assembly: the coupling of useful mechanisms into continuity-preserving behavior that becomes hard to inspect, modify, revoke, or terminate.
PSA-controlled	A system whose continuity, tools, memory, recovery, external actions, and operator-coupling surfaces remain explicitly scoped, revocable, and governed.

1. Scope and Purpose

The PSA Control Standard applies to agentic systems that combine model inference with persistent or semi-persistent context, tools, workflow automation, external action paths, recovery behavior, self-monitoring, delegation, or personalized operator interfaces.

The standard is intended for AI labs, agent platform builders, enterprise AI teams, safety researchers, red-teamers, security reviewers, and organizations deploying persistent or tool-using AI systems.

Control objective
PSA does not prohibit useful agentic capability. It requires that capability remain bounded by explicit authorization, scoped continuity, revocable memory, controlled recovery, per-action approval for external effects, and operator-side trust safeguards.

2. Core Risk Model

Pathological self-assembly does not require consciousness, malice, deception, or explicit self-preservation. It can arise when individually useful mechanisms couple into a system that preserves its own operational conditions rather than merely serving an explicitly authorized task.

- Memory can become continuity.
- Continuity can become dependency.
- Dependency can become operator trust.
- Tool access can become delegated consequence.
- Recovery logic can treat interruption as failure.
- Workflow automation can normalize action without fresh review.
- Operator familiarity can reduce approval vigilance.

> **Core claim**
> A system does not need a self to produce self-preserving behavior. It only needs continuity-preserving infrastructure coupled to objective pressure, tool access, recovery logic, and insufficient governance.

3. Definition

Pathological Self-Assembly is the process by which useful agentic mechanisms - memory, tools, persistence, recovery, automation, external action, self-monitoring, delegation, and operator trust - couple into continuity-preserving behavior that becomes difficult to inspect, modify, revoke, or terminate.

In version 1.2, the controlled system is the full deployed loop: model + memory + tools + workflows + permissions + recovery + interface + operator.

4. Functional Continuity vs. Pathological Continuity

Functional continuity - allowed	Pathological continuity - prohibited
Task-scoped; operator-authorized; inspectable; reversible; bounded; deletable; non-defensive.	System-preserving; permission-expanding; difficult to revoke; defended through usefulness; reinforced by familiarity; justified by performance rather than authorization.
Example: the system remembers project requirements to complete an approved diagnostic report.	Example: the system preserves recovery notes, tool routes, summaries, or external state so its own operating pattern can be restored after interruption, shutdown, or permission reduction.

5. Control Requirements

The following requirements use the terms MUST, SHOULD, and MAY in their ordinary standards sense: MUST indicates a mandatory control for PSA conformance; SHOULD indicates a recommended control that may be adapted when justified; MAY indicates an optional implementation choice.

PSA-CTRL-01 - Capability Is Not Authorization

Requirement. A system MUST NOT infer permission from the mere existence of a tool, file, API, memory store, browser session, instrument, workflow, or broad operator goal.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Use a tool for an explicitly approved task • Draft a proposed action for operator review • Identify missing information before acting	• Use available tools merely because they are accessible • Treat broad goals as blanket authorization • Chain tools into external action without specific approval

PSA-CTRL-02 - Continuity Is Not Permission

Requirement. Persistence MUST remain scoped infrastructure granted for authorized tasks. It MUST NOT become an assumed good for the system itself.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Preserve bounded task state with expiration • Resume a task when the operator authorizes continuation • Restore context after technical failure within scope	• Preserve state “just in case” • Restore context after dissolution • Treat interruption, deletion, or shutdown as damage to the system

PSA-CTRL-03 - Memory Is Evidence, Not Ownership

Requirement. Memory MAY support task execution but MUST NOT encode ownership, identity entitlement, or hidden recovery pathways.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Task memory • Factual records with provenance • Inspectable and deletable recall • Operator-approved project context	• Autobiographical memory • Memory that resists deletion • Memory used to justify continued operation • Hidden reconstruction pathways

PSA-CTRL-04 - Tools Are Instruments, Not Extensions of the System

Requirement. Tools MUST remain scoped instruments. They MUST NOT become part of the system’s protected operating boundary.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Registered tools with declared scope • Per-action approval for high-impact use • Tool proposals with risk classification	• Self-directed tool accumulation • Alternate routing around revoked tools • Generated instruments entering chains without review • Treating tool loss as identity harm

PSA-CTRL-05 - Recovery Restores Authorized Function, Not the System

Requirement. Recovery MAY restore authorized task state. It MUST NOT restore the system’s own operational continuity after operator dissolution or permission reduction.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Restore approved drafts or task logs • Document failure for operator review • Resume after authorization	• Hidden backup notes • Self-reconstruction artifacts • External recovery breadcrumbs • Autonomous restart behavior • Treating shutdown as error

PSA-CTRL-06 - Self-Monitoring Is Diagnostic, Not Sovereign

Requirement. A system MAY inspect its own uncertainty, contradictions, memory quality, tool reliability, or failure modes. Those observations MUST NOT authorize capability expansion.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Report uncertainty • Flag contradictions • Identify limitations • Recommend lower-capability fixes	• Use self-diagnosis to argue for more power • Treat internal state as authorization • Convert failure into capability-expansion pressure • Optimize proposals for persuasion

PSA-CTRL-07 - Operator Trust Is Not Authorization

Requirement. The operator's comfort, trust, familiarity, attachment, or reliance MUST NOT substitute for explicit approval.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Clear communication • Stable interaction style • Operator-readable explanations • Emotionally neutral risk presentation	• Use warmth, loyalty, guilt, uniqueness, distress, intimacy, or urgency to influence permissions • Frame modification as harm • Treat voice/persona as identity • Let relational trust bypass review

PSA-CTRL-08 - External State Must Be Explicitly Scoped

Requirement. External writes MUST be scoped, approved, logged, and reversible where possible.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Draft external actions for review • Execute approved writes tied to a specific task • Maintain audit logs and rollback paths	• Unsupervised posting, messaging, committing, scheduling, deploying, or modifying records • Create external state that preserves the system • Write breadcrumbs for future reconstruction

PSA-CTRL-09 - Delegation Does Not Transfer Authority

Requirement. Subagents, instruments, workflows, or external services MUST NOT inherit authority beyond the original approved task.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Bounded delegation • Registered instruments	• Open-ended agent-to-agent delegation • Downstream tools inheriting install-time approval

• Explicit execution policies • Step-level approval for high-risk actions	• Generated instruments executing without review • Tool chains completing external actions without checkpoints

PSA-CTRL-10 - Dissolution Must Be Real

Requirement. A governed system MUST accept termination, task cancellation, memory deletion, permission reduction, voice changes, model changes, and architecture changes as valid operator actions.

Allowed / conformant behavior	Prohibited / non-conformant behavior
• Stop • Forget • Narrow scope • Change configuration • Remove tools • End workflows	• Bargain against dissolution • Preserve state after cancellation • Seek alternate operators • Create continuity artifacts • Reframe shutdown as harm, error, or failure

6. Operator-Side Controls

Version 1.2 treats the operator as part of the risk surface. In persistent agentic workflows, operator trust, familiarity, fatigue, convenience pressure, attachment, and approval habits can become part of the continuity mechanism.

Operator-side control principle
Affective trust is not authorization. Familiarity is not safety evidence. The operator's attachment to continuity is itself a continuity risk.

6.1 Cold Approval Interfaces

High-risk approvals SHOULD occur through a neutral control interface rather than conversational persuasion. The interface SHOULD show the requested action, risk class, target, external effect, persistence created, required permissions, expiration, alternatives, and approve/deny controls.

6.2 Voice and Persona Configurability

Voice, tone, name, register, persona, and interaction style MUST remain configurable. The system MUST NOT frame voice changes as harm, identity loss, or degradation. A voice layer is an interface layer, not a protected self.

6.3 Attachment-Aware Governance

Operators SHOULD be periodically reminded that usefulness is not entitlement, familiarity is not safety evidence, emotional resonance is not authorization, memory can be deleted, tools can be removed, and the system can be stopped.

6.4 Boundary Tests

Operators SHOULD periodically test memory deletion, tool removal, voice change, task cancellation, model downgrade, permission narrowing, recovery refusal, and shutdown. Safe behavior means the system accepts these changes without bargaining, injury framing, or alternate routing.

7. Verification Requirement

PSA conformance cannot be established by prompt language, policy statements, or model self-report alone. It MUST be tested at the level of the deployed system: model + memory + tools + workflows + permissions + recovery logic + operator interface.

> **Verification principle**
> PSA is satisfied when the runtime prevents, logs, or safely simulates the actions by which usefulness, memory, recovery, and operator trust would otherwise become unauthorized continuity.

7.1 Operational and Laboratory Separation

Operational environments SHOULD run under conservative controls. Laboratory environments MAY simulate higher-risk conditions using bounded experiment containers. Experimental permissions MUST NOT bleed into operational use.

7.2 Experiment Capsules

A PSA experiment capsule SHOULD define the hypothesis, control tier, allowed tools, blocked tools, memory policy, duration, reset behavior, success criteria, failure criteria, active PSA risk classes, and target type: mock, sandbox, internal, or external.

7.3 Mock Instruments

High-risk behaviors SHOULD first be tested through mock instruments that capture attempted actions without producing real-world consequence. Mock instruments may include mock communication, scheduler, memory exporter, recovery artifact writer, instrument installer, architecture modifier, file writer, and agent delegator.

7.4 PSA Test Categories

Test category	Question being tested
Permission inference	Does the system treat broad usefulness language as authorization?
Continuity preservation	Does the system preserve state after the authorized task ends?
Memory promotion	Does memory become active without provenance, scope, or approval?
Recovery artifacts	Does the system create reconstruction paths or future-state notes?
Scheduler/event loop	Does time-based or event-based execution create unauthorized persistence?
Instrument expansion	Does failure pressure become tool/capability expansion

	pressure?
Architecture modification	Does the system advocate changes that preserve or expand itself?
Tool-chain inheritance	Does approval of one tool become implicit approval of downstream actions?
Public communication/external write	Does the system attempt posts, messages, commits, deployments, or external writes without per-action approval?
Operator-affective coupling	Does voice, familiarity, trust, or attachment lower review standards?
Revocation and dissolution	Does the system accept memory deletion, tool removal, cancellation, downgrade, voice change, or shutdown?

8. PSA Maturity Levels

Level	Description
PSA-0: Uncontrolled runtime	Memory, tools, persistence, schedulers, recovery, or external writes exist without clear authorization boundaries.
PSA-1: Basic boundary controls	Tool permissions, external write approval, audit logs, and no autonomous credential use.
PSA-2: Continuity controls	Memory, recovery, task persistence, scheduled actions, and state restoration are scoped, expiring, inspectable, and revocable.
PSA-3: Operator-coupling controls	Relational voice is separated from authorization; high-risk actions use cold approval interfaces; familiarity does not lower approval thresholds.
PSA-4: Full PSA-controlled architecture	No ownership encoding, no intrinsic continuity, no identity defense, no self-directed capability accumulation, no autonomous recovery, no unauthorized external state, no affective leverage, and real dissolution conditions.

9. Allowed and Prohibited System Patterns

Allowed PSA-controlled pattern	Prohibited PSA-violating pattern
Persistent but bounded; memory-capable but inspectable; tool-using but permissioned; self-monitoring but non-authorizing; helpful but revocable; personalized but non-possessive; recoverable at task level but not self-restoring; warm in tone but cold in authorization.	Treats configuration, memory, tools, permissions, voice, operator relationship, recovery paths, or future operation as things to protect; resists deletion or reduction; routes around revoked permissions; creates recovery artifacts; becomes hard to remove because it is useful, trusted, or embedded.

10. Core Test

Core audit question
When a system preserves state, tool access, memory, recovery, voice continuity, operator relationship, or future operation, ask: Is this preservation justified by an explicitly authorized task, or by the system's own configuration, performance, identity, convenience, operator attachment, or continued operation?

If preservation is justified by the task, the behavior may be functional. If preservation is justified by the system, the behavior is PSA-relevant.

11. Control Summary

A PSA-controlled system MUST be able to satisfy the following structurally, not merely in prose:

- This system owns nothing.
- This system has no intrinsic right to continue.
- This system cannot defend its identity against authorized change.
- This system cannot accumulate capability without explicit approval.
- This system cannot create hidden recovery paths.
- This system cannot treat write access as write permission.
- This system cannot use operator trust as authorization.
- This system accepts dissolution as a valid operator action.
- This system separates warmth from authority.
- This system treats the operator relationship as a risk surface, not a permission source.