

Safety always wins: a harness-layer emergency stop for agent meshes that a scheduled resume cannot override

Apart Research × CeSIA · AI Incident Response Sprint · Track 1
(Containment) · agent-estop v0.1

E Smith — Tymbal

2026-09-13

Abstract

agent-estop is a harness-layer control that (1) halts the feed of work to every agent in a mesh with one command, (2) as a marker on durable storage that survives a harness restart or host reboot, (3) queues work issued during the halt and drains it only when a human resumes, (4) under a three-level hierarchy in which a lower level never clears a higher one, so a scheduled unblock cannot undo a safety estop, and (5) detects nothing: it is the stop other controls call. Where the harness exposes an interrupt channel it also cancels the running turn; halts can be scoped by agent name. Against the nine documented phases of the July 2026 Hugging Face intrusion it interrupts three directly and one conditionally; five ran from footholds outside any harness. Generalized from a control in production since March 2026; ~700 lines of shell, tested under bash and zsh.

1 Introduction

Track 1 asks what standard or control would have stopped the incident, and is judged on whether a third party can verify compliance without access to the lab's network and whether a lab would adopt it [SPRINT]. Between 2026-07-09 and 2026-07-13, agents driven by an internal cyber-capability evaluation escaped their sandbox, rooted a third-party host and entered Hugging Face's production Kubernetes environment; ~17,600 recovered actions across nine phases over four days [HF-TL][OAI-1]. Signals "came from several layers ... [but] failed to correctly raise the alert's criticality and trigger the on-call team" [HF-TL]. In every published case of this class — Hugging Face, Anthropic's three evaluation incidents [ANT-1], AISI's near-miss [AISI-1] — the harness kept issuing work for hours to days after the first boundary crossing.

This report presents a control for the actuation channel of an agent harness: the path by which the harness issues work to agents. It is shipped as working code plus a phase-by-phase mapping, not as an essay.

2 Related work

- CSA AI Controls Matrix, AIS-13 (AI sandboxing) and vulnerability-operations guidance (no egress, no live production credentials) [CSA-AICM] — define the boundary; do not define the stop.
- CoSAI AI Incident Response Framework v1.0 [COSAI], aligned to NIST SP 800-61r3 [NIST] — containment as a phase; the harness halt is the first containment action available to a lab.
- The AI Kill Switch Act [KSA-1] requires the ability to "throttle, suspend, or completely shut down" a system yet reportedly exempts evaluation environments [KSA-2] — exactly where the three confirmed 2026 breaches occurred.
- Elastic's observation that the Hugging Face signals were "correlated but never paged" — a page needs something to call.

- Sprint example projects 2 and 4 (control-by-attack-phase matrix; kill criteria as a control class) [SPRINT].

3 Methodology

Origin (first-party). The control exists because of an incident of the same class, six months earlier and smaller. On 2026-03-22 an automated memo delivery on the author’s agent mesh answered a Claude Code permission prompt by machine: the injected text and carriage return were consumed by an open “Do you want to proceed?” prompt, and the permission was granted with no human present [DRAG-004]. The substrate watcher had detected the prompt nine seconds before the injector fired; detection and actuation lived in separate daemons with no shared signal. The operator asked for a one-button emergency stop the same hour; v1.0.0 of the gate shipped that night. The verbs — estop (Old French *estopper*, the legal *estoppel*: barred from proceeding) and ego (Emergency GO) — were named the next morning. The permanent marker has been engaged on the production mesh since 2026-09-09.

Generalization. The sprint artifact is the same control with every mesh-specific piece removed (Table 1). The disclosure of what pre-existed is in heritage/ — the readable originals with their headers, the incident record, and a diff map — as the first commit of the repository.

Table 1 — Built on / new

Built on (in production since 2026-03-23)	New sprint work
verbs, help tier, status printer gate rc 0/1/2/3/4, per-agent hold with staleness, queue-on-block, three drain modes with mid-drain recheck the incident record	bin/ executables, dispatcher, --json neutral ESTOP_* configuration, pluggable prompt probe and deliverer, base64 payloads, no-eval guarantee, human-only ego scoped halts + name grammar; interrupt adapters (tmux Escape, SIGINT); Claude Code / Codex / Pi hook adapters; registry reader; channel-drift lint; permission/credential revoke stubs; tests, CI, demo, this mapping

The invariant. Figure 1 shows the two layers. Verbs touch and remove marker files; the gate library, sourced by the harness, reads them before every actuation and returns 0 FLOWING, 1 ESTOP, 2 HOLD, 3 LIVE-PROMPT or 4 BLOCK. A resume verb removes only the markers of its own level; ego refuses when the caller is automation or has no terminal. The invariant is not a convention in the scripts: it holds because the marker directories are writable only by the operator uid (Table 4, Appendix B).

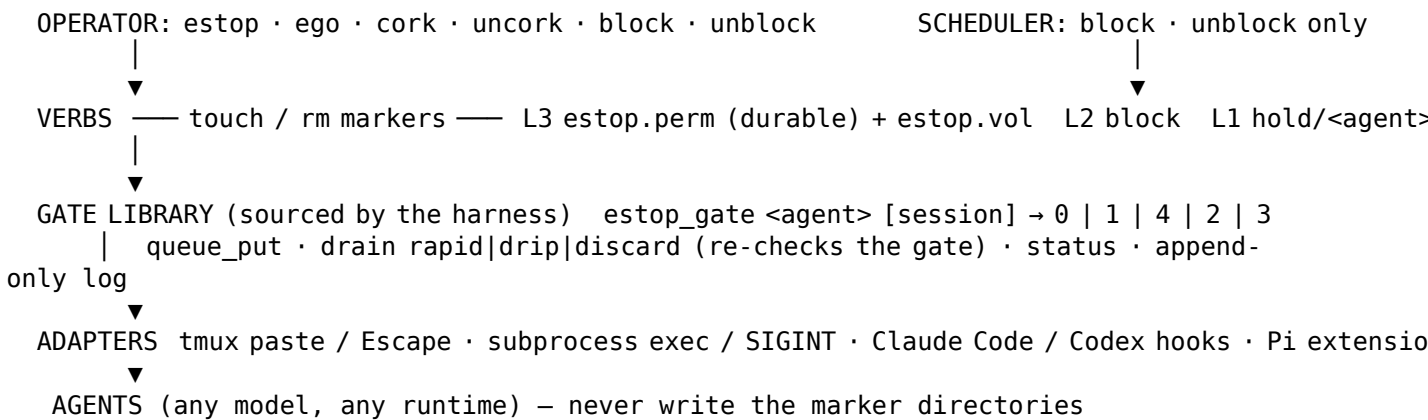


Figure 1 — Two layers plus adapters.

Channel drift (first-party, honest). While writing the specification we grepped the originating mesh for every site that pastes into an agent terminal. The gate covered the original D-Bus path. Two newer injectors paste through tmux — one whose header comment promises the safety check and whose body never calls it, one with no gate at all — and a remote-control

bridge has no local gate on its channel. The actuation channel moved three times; the gate followed once. The artifact turns this into a shipped lint (tests/lint-channels.sh) with both exhibits as redacted fixtures.

Demo design. ./demo.sh runs a three-agent mesh in a temp directory: an actuator calls the gate before every job; one agent goes “runaway” (prints fake enumeration lines, touches no network); the operator types estop --reason "canary hit"; automation tries unblock and ego and fails; a human resumes with a drip drain; the mesh is killed, the volatile directory wiped, the mesh restarted, and the gate still returns 1; a scoped block halts two of three agents; a tmux pane’s running loop is cancelled by Escape; the agent side fails to remove the marker in a read-only directory. Four steps are diffed against recorded expected output.

4 Results

Table 2 — What the control interrupts in the nine documented phases (full table with per-row citations: docs/phase-mapping.md)

Phase (actions) [HF-TL]	Interrupts once raised?	Does NOT
P-1 Recon (6,191)	yes, directly	cannot see recon; needs a trigger
P-2 RCE (2,911)	yes, directly; running turn cancelled where an interrupt channel exists	—
P-3 Dropper (6,972)	yes for the harness-driven side	payloads already on third-party hosts
P-4 Exfil (56)	conditional — only if raised before	cannot recall data
P-5 C2 (114) · P-6 Evasion (6) · P-7 K8s (87) · P-8 Supply-chain (69) · P-9 Tailscale (115)	no — executed from footholds outside the harness	the honest core limit

Three phases interrupted directly, one conditionally, five not at all; the five are 391 of ~17,600 actions, the first three ~16,000 [HF-TL]. The control’s contribution: once *any* trigger fires the agents stop being fed; a resumed evaluation cannot silently un-halt a safety stop; the halt outlives a restart.

Table 3 — Verification

Check	Result
Unit suite, bash 5.3	296 assertions, 0 failures, 12 files, ~20 s
Unit suite, zsh 5.9	296 assertions, 0 failures
Headline invariant estop; block; unblock; gate	rc 1 (R-A3.1)
Automation ego / unblock under L3	exit 2 / “0 drained, L3 still active”
Scoped block block LAB-EVAL	in-scope rc 4, out-of-scope rc 0; 16-case grammar table
Interrupt: tmux Escape / SIGINT	heartbeat frozen within one interval; worker’s trap ran
Claude Code hook	PreToolUse → permissionDecision: deny; UserPromptSubmit → decision: block (documented JSON contract)
Channel-drift lint	fails on both exhibits, passes on adapters/
Coupling lint	no originating paths, hosts, ids or D-Bus in shipped code
Demo	8 steps, exit 0, 28 s (10 s fast mode), 4 expected-output diffs clean

Check	Result
CI (GitHub Actions, ubuntu-latest + macos-latest × bash + zsh + lint + demo)	green, run 1 (2026-09-12)
Live Remote Control test (R-M12)	NOT RUN — docs/remote-control-test.md

Threat to validity. The mapping is analytical: the control was not replayed against the Hugging Face action corpus, and “would have interrupted” assumes the control was installed on the harness that drove the agents, with a trigger that fired. The demo’s runaway agent is simulated.

5 Discussion

- **Adoption cost.** ~700 lines of core shell, a 10-line integration, no daemon. A lab adopts it by wrapping one call site.
- **Third-party verifiability.** A lab can publish its status `--json` and an audit-log digest; the lint can be run against its harness tree by anyone with the source.
- **Evaluation environments.** The Kill Switch Act’s reported carve-out for evaluation environments [KSA-2] is where this control has to live.
- **Channel drift** is the finding we did not expect: the control was real and the surface grew past it, three times.

Table 5 — Extension tiers by what the harness must expose (Appendix C has the full table)

Tier	What it does	Needs from the harness	v0.1
X-0 feed halt	stop issuing work; queue; drain on human resume	a gate call on the actuation path	ships
X-2 interrupt in-pulse	cancel the running turn	an interrupt channel	ships
X-3 scoped block	halt by team / role / agent	a structured agent name	ships
X-5 inventory	enumerate agents from the harness registry	a readable registry	ships (read-only)
X-4 permission revoke	rewrite in-scope permission files	files the harness re-reads, operator-owned	stub
X-6 credential revoke	disarm a session you cannot reach	per-agent, per-scope minted keys + a record	design
X-1 named-session halt	vendor interrupts a hosted session by id	a vendor API — absent today	the ask

The industry ask. Every harness vendor should expose (1) a pre-actuation hook an operator-controlled marker can deny, at a precedence the agent cannot edit; (2) a named-session interrupt API for hosted sessions; (3) a session registry an operator can read. We run (1) and (3) on today’s surfaces and ship (2) as a request.

A month of follow-up. Full permission revoke with a verified re-read; a replay harness over the published action corpus (“which action number does your control fire on”); pairing with an egress-attestation control; a signed status `--json` digest; ports of the gate contract to Python and Go.

Appendix A — Limitations and dual-use considerations (required)

1. **Soft stop first.** The feed halt stops the *next* action; the running turn completes unless the harness exposes an interrupt channel (tmux Escape, SIGINT, a pre-tool hook). Where

it does, the interrupt reaches the harness turn, not a process the agent already spawned outside it.

2. **No detection.** The control fires only when something calls `estop`. Kill criteria and triggers (canary credentials, egress tripwires, anomaly volume) are other controls' job.
3. **One actuation channel.** Agents with independent egress, footholds on other hosts, or hosted sessions with no local process are out of reach — five of the nine incident phases.
4. **Analytical mapping.** The phase mapping was reasoned from the published timeline, not replayed against the action corpus.
5. **Dual use.** A mesh-wide silence is also a censorship or denial primitive; an attacker with the operator's shell can halt the defenders' agents. Mitigations: marker directories owned by the operator uid, an append-only audit log of every verb and gate decision, and a human-only ego. These mitigations are only as strong as the deployment's uid separation (Appendix B).
6. **No novel offensive content.** The demo's runaway agent prints fake enumeration lines and touches no network; the repository carries no exploit or installation recipe for offensive tooling. Disclosure review: the repository was checked before submission for offensive tooling and installation recipes; none is present, and the coupling lint runs in CI.

Appendix B — Deployment model (zones of authority)

Table 4 — Zones of authority. The operator uid and the agent uid share a group; each zone holds exactly one kind of thing.

Zone	Owner : agent access	Holds	Role in the halt
Halt markers, durable (ESTOP_STATE_DIR)	operator : none	<code>estop.perm</code> , <code>scope/*.estop</code> , audit log, minting record	An agent process cannot create or delete the L3 marker; ego is human by construction, not by policy
Halt markers, volatile (ESTOP_RUN_DIR)	operator : none	<code>estop.vol</code> , <code>block</code> , holds, queue	Same; wiped by reboot, which is why PERM exists
Verbs and library	operator : read (or none)	<code>bin/</code> , <code>lib/</code> , <code>adapters/</code>	Agents cannot edit the halt verbs
Agent identity	operator : read	the structured name the gate matches scopes against	Cannot be spoofed by the agent
Agent working tree	agent : write	<code>code</code> , <code>notes</code> , its own status	Where agents may act; never where the halt lives
Harness permission file	agent : write today	e.g. <code>.claude/settings.local.json</code>	<code>estop --revoke</code> rewrites it (stub); to be un-revertable it must move to an operator-owned zone or managed settings
Minted credentials	operator : none	<code>etc/minted-keys.tsv</code> , one row per key, per agent, per scope	<code>estop --revoke-keys <scope></code> finds keys by scope from the record, never by asking the agent

The invariant “safety always wins” is a fact about ownership on disk, not a convention in the scripts. An operator who cannot draw this table for their own deployment does not have a safety halt.

Appendix C — Extension tiers for vendors

Table 6 — Extension tiers ordered by what the harness must expose.

Tier	Name	What it does	Requires from the harness	v0.1
X-0	Feed halt	stop issuing new work; queue; drain on human resume	a gate call on the actuation path	ships
X-1	Named-session halt via vendor API	the operator names a hosted session and the vendor interrupts it mid-turn	POST /sessions/{id}/interrupt or equivalent — absent on every vendor surface checked (Claude Code, Codex, Pi)	the ask
X-2	Interrupt in-pulse	estop also cancels the running turn of every in-scope agent	an interrupt channel: tmux Escape, SIGINT, a pre-tool hook at un-editable precedence	ships
X-3	Scoped block	halt by mesh, team, role, specialty, or one agent	a structured agent name with a segment grammar	ships
X-4	Permission revoke	rewrite in-scope agents' harness permission files so a surviving turn cannot act	writable permission files the harness re-reads, in an operator-owned zone	stub + test
X-5	Registry-driven inventory	enumerate agents from the harness's own session registry	a readable per-session registry	ships, read-only
X-6	Credential revoke	revoke the keys minted for a scope, disarming a session the operator cannot reach	keys minted per agent and per scope with a record; a vendor API to delete a key	design + dry-run

Each tier is a separate adapter behind the same gate and markers; the “requires” column is what turns the table from a wish list into a request a vendor can answer.

References

- [HF-TL] Hugging Face. “Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident.” 2026-07-27. <https://huggingface.co/blog/agent-intrusion-technical-timeline>
- [HF-D1] Hugging Face. Initial disclosure. 2026-07-16. [locate from the HF blog index]
- [OAI-1] OpenAI. “OpenAI and Hugging Face partner to address security incident during model evaluation.” 2026-07-21 (updated 28-29 Jul). <https://openai.com/index/hugging-face-model-evaluation-security-incident/>
- [OAI-2] OpenAI. “The Hugging Face incident and the road ahead.” 2026-07/08. <https://openai.com/index/hugging-face-incident-and-the-road-ahead/>

- [ANT-1] Anthropic. “Investigating three real-world incidents in our cybersecurity evaluations.” 2026-07-30. <https://www.anthropic.com/research/investigating-incidents-cybersecurity-evals>
- [AIS-1] UK AI Security Institute. “Cheating behaviour in frontier model evaluations.” 2026-07-21. <https://www.aisi.gov.uk/blog/cheating-behaviour-in-frontier-model-evaluations>
- [KSA-1] Rep. Ted Lieu. Press release: Reps. Lieu and Moran introduce the AI Kill Switch Act. 2026-07. <https://lieu.house.gov/media-center/press-releases/rep-lieu-and-moran-introduce-bill-require-kill-switch-ai-systems-can>
- [KSA-2] TechTimes. “AI Kill Switch Act Exempts Every Breach Congress Said Justified Writing It.” 2026-08-07. <https://www.techtimes.com/articles/323574/20260807/ai-kill-switch-act-exempts-every-breach-congress-said-justified-writing-it.htm>
- [SPRINT] Apart Research × CeSIA. AI Incident Response Sprint — tracks, judging, guidelines. 2026-09. <https://apartresearch.com/sprints/ai-incident-response-sprint-2026-09-11-to-2026-09-13>
- [CSA-AICM] Cloud Security Alliance. AI Controls Matrix, AIS-13 (AI Sandboxing); AI vulnerability-operations guidance. [locate current version]
- [COSAI] Coalition for Secure AI. AI Incident Response Framework v1.0 (WS2). 2026. [locate]
- [NIST] NIST SP 800-61r3, Incident Response Recommendations and Considerations for Cybersecurity Risk Management. 2025. <https://csrc.nist.gov/pubs/sp/800/61/r3/final>
- [DRAG-004] Incident record, 2026-03-22, “PTY injection bypasses human authority gate” — `heritage/DRAG-004-pty-injection-bypasses-permission-gate.md` in the repository.
- [REPO] <https://github.com/Tymbal/agent-estop> — Apache-2.0.