DEFENSIVE PUBLICATION — TECHNICAL DISCLOSURE (TDCOMMONS DEPOSIT COPY) · 2026-06-25

Keywords: defensive-publication, prior-art, ai-agents, adaptive-autonomy, human-in-the-loop, confidence-calibration, agent-safety

Dynamic Confidence Bands per Action Type

A Technical Defensive Publication

Subtitle: Per-action-type confidence bands with rolling, human-free recalibration and an auto-execute / escalate / human-review routing trichotomy for adaptive AI-agent autonomy.

Field	Value
Publisher / Copyright holder	Gus IT LLC (Florida, USA)
Author	Gustavo Assuncao, PhD
Publication date	2026-06-25
Version	1.0
Document type	Technical Defensive Publication (public prior art)
Classification	Public
License	AGPL-3.0-or-later (copyleft; commercial license available)
Deposit	To be assigned (IP.com / Zenodo / arXiv) — establishes a public, dated, citable prior-art record.

Abstract

AI agents acting on a user's behalf — sending email, posting chat replies, creating tasks, calling tools, delegating to sub-agents — must decide, for each action, whether to **act autonomously**, **escalate for a quick check**, or **hold for full human review**. The prevailing mechanism is a **single static confidence threshold** ("auto-send if confidence > 0.8"), hand-tuned by operators. A single threshold is brittle: it cannot distinguish a low-stakes chat reply from a high-stakes external email; it does not adapt as the agent demonstrates competence (or begins to fail); and it forces every action type through one gate.

This publication discloses a method in which **each action type maintains its own confidence *band*** — an ordered pair `(low, high)` — rather than a single threshold. On completion of each action, the system records an outcome and **recomputes that band, without human intervention**, as a function of a **sliding window** of recent outcomes for that action type: the band **tightens** (raises `low`/`high`) as failures rise and **loosens** (lowers them) as successes accumulate. A subsequent action is routed by a **trichotomy**: confidence **above** `high` → auto-execute; **below** `low` → require human review; **in between** → escalate (a lightweight, time-boxed check). Because every action type calibrates independently, an agent may become fully autonomous on chat replies while still requiring review on external email — a **graduated autonomy gradient that emerges from observed outcomes**, not from manual tuning. The disclosure provides a system architecture, the detailed recalibration and routing mechanics, a data model, an enabling clean-room reference implementation, a fully worked example, a security and failure-mode analysis, standards/framework alignment, an evaluation methodology, and enumerated novelty claims (one

independent plus fourteen dependent). It is published to establish dated, public prior art so the technique remains freely practiceable.

---

1. Executive Summary

1.1 Thesis

Static confidence thresholds are the wrong control surface for agent autonomy. The right surface is a **set of per-action-type bands that the system tunes itself from outcomes**, paired with a **three-way routing decision** (auto / escalate / review). This converts autonomy from a fixed operator setting into an **earned, observable, per-capability property** of the agent.

1.2 Contributions

#	Contribution	Where
C1	Replacing a single threshold with a per-action-type `(low, high)` **band**.	§6, §7
C2	A **routing trichotomy** (auto-execute / escalate / human-review) keyed off the band rather than a binary pass/fail.	§7.3
C3	**Rolling, human-free recalibration** of each band from a sliding window of outcomes, tightening on failure and loosening on success.	§7.4
C4	**Independent per-action-type calibration** (email, chat, task, tool, delegation each have their own band and window).	§7.1, §8
C5	A **graduated autonomy gradient** that emerges from observed competence, with bounded clamps and hysteresis for stability.	§7.5, §11
C6	An **auditable outcome ledger** enabling explainability ("why was this auto-sent?") and reproducible recalibration.	§8, §11

1.3 Headline claim

> A method that maintains, **for each action type** of an AI agent, a confidence **band** `(low, high)`; **records an outcome** on completion of each action and **recomputes that band without human intervention** from a **sliding window** of outcomes; and routes a subsequent action to **auto-execute** (confidence > `high`), **human-review** (confidence < `low`), or **escalate** (otherwise).

1.4 Scope — what it is / is NOT

It is: a control-plane mechanism that sits between an agent's decision step and its action executor; a self-tuning, per-capability autonomy governor; a source of audit records.

It is NOT: a model-training method (it does not change agent weights); a confidence *estimator* (it consumes a confidence value produced upstream and does not prescribe how that value is computed); a general classical-control PID loop (it is discrete, per-action-type, outcome-driven, and routing-trichotomous); nor a claim over any single static threshold (which is well-known prior art).

---

2. Introduction & Motivation

2.1 The concrete problem

Consider a "cognitive persona" — an autonomous agent that triages a user's inbox, drafts replies, files tasks, and calls tools. Operators want it to handle the routine load by itself while keeping a human in the loop for anything risky. Today they express this with a number: *auto-act when the model's confidence exceeds T*. Picking τ is a no-win:

- **τ too low** → the agent auto-sends a bad external email; trust collapses; operators yank autonomy back to zero.
- **τ too high** → almost everything queues for human review; the agent is a glorified draft generator; the promised labor savings never arrive.
- **One τ for all action types** → the threshold appropriate for an internal chat reply is dangerous for an outbound customer email and vice-versa.

2.2 The "tax"

The hidden tax of static thresholds is **manual tuning forever**. Every new action type, every shift in task mix, every change in user tolerance requires an operator to re-pick numbers — usually reactively, after an incident. There is no mechanism by which the agent *earns* more autonomy by being right, or *loses* it by being wrong; a competent agent and an incompetent one share the same gate. The review queue is sized for the worst case, so humans review far more than they need to. Field experience with single-threshold gating motivates a target of **~40% reduction in human-review burden** once autonomy adapts per capability (see §13; treat the figure as illustrative).

2.3 Why existing approaches fall short

- **Single static threshold:** no adaptation, no per-type granularity, binary routing.
- **Generic adaptive thresholds (classical control):** can move a single threshold based on error, but are not framed per-action-type, do not produce a *band*, and do not implement an auto/escalate/review **trichotomy** — they output a continuous control signal, not a discrete routing decision over three zones.
- **Bandit / RL autonomy:** powerful but heavyweight, opaque, slow to converge, and hard to audit ("why did it act?"). Operators in regulated settings want a legible rule, not a learned policy network.

2.4 Why now

Multi-action autonomous agents are becoming production infrastructure. The moment an agent can *send* rather than *draft*, the autonomy-governance question becomes load-bearing and safety-critical. A lightweight, auditable, self-tuning governor is exactly the missing primitive.

---

3. Problem Statement

3.1 Formal framing

Let $A = \{a_1, \ldots, a_k\}$ be the set of **action types** an agent can perform (e.g. `email`, `chat`, `task`, `tool`, `delegation`). For each action type $a \in A$ maintain a **band** $B_a = (low_a, high_a)$ with $0 \le low_a \le high_a \le 1$.

For each candidate action the agent emits a **confidence** $c \in [0, 1]$. Define the routing function:

```
route(a, c) =
  AUTO    if c >  high_a
  REVIEW  if c <  low_a
  ESCALATE otherwise (low_a ≤ c ≤ high_a)
```

After the action completes, an **outcome** $o \in \{success, failure\}$ (optionally graded) is recorded against a. Maintain a sliding window W_a of the most recent N outcomes for a. Periodically (every M outcomes or on a timer) recompute $B_a = f(W_a)$ such that the band **tightens** as the recent failure rate rises and **loosens** as the recent success rate rises, **without human intervention**, subject to clamps $[B_{min}, B_{max}]$ and a stability constraint (hysteresis / max-step).

3.2 Derived requirements

ID	Requirement	Sections
R1	Maintain an independent confidence band, not a single threshold, **per action type**.	§6, §7.1, §8
R2	Route each action by a **three-way** decision (auto / escalate / review), not binary.	§7.3
R3	Record an **outcome** for every completed action, against its action type.	§7.2, §8
R4	Recompute each band from a **sliding window** of recent outcomes.	§7.4
R5	Recalibrate **without human intervention** (no operator re-tuning).	§7.4, §7.5
R6	**Tighten on failure, loosen on success**, monotonically in recent success rate.	§7.4
R7	**Clamp** bands to safe $[B_{min}, B_{max}]$ bounds and preserve $low \le high$.	§7.5, §11
R8	**Stabilize** updates (hysteresis / max-step / min-samples) to avoid thrashing.	§7.5, §11
R9	Produce an **auditable record** of band state and routing decisions.	§8, §11
R10	Be **executor-agnostic** — sit in front of any action executor via a narrow interface.	§6, §9
R11	**Fail safe** — degrade toward review, not toward unchecked auto-execution.	§11
R12	Be **configurable** (N, M, learning rate, targets, clamps) per deployment and per action type.	§7.6, §9

4. Related Work & Prior Art

This work **builds on**, and is deliberately positioned against, the following real and well-known prior art. We acknowledge each and identify precisely what it provides — and what it does not.

- **Static confidence thresholding in human-in-the-loop (HITL) systems.** Decades of HITL ML deploy a single cutoff: act/automate above T, defer to a human below T. This is the dominant

baseline and is well-known. It lacks per-type bands, the routing trichotomy, and self-recalibration.

- **Classical adaptive control / adaptive thresholding (control theory).** A controller adjusts a setpoint from observed error (e.g. adaptive thresholds in signal detection, CFAR detectors in radar, adaptive binarization in vision). These move a *single* value continuously and emit a *continuous* control signal; they are not framed per-action-type, do not produce an explicit `(low, high)` band with an in-between escalation zone, and do not implement a discrete auto/escalate/review routing decision.
- **Conformal prediction & selective prediction (reject option).** Produces a prediction *set* or a "reject/abstain" option calibrated to a target error rate. Relevant and adoptable as an upstream confidence source, but it calibrates *coverage of a prediction*, not *autonomy routing across distinct agent action types*, and has no rolling per-action-type recalibration of an autonomy band.
- **Multi-armed bandits / contextual bandits / RL for autonomy.** Can learn when to act, but are heavyweight, opaque, slow to converge, and not auditable as a simple rule. We adopt the *spirit* (learn from outcomes) but reject the mechanism in favor of a legible band.
- **Operational SLO error budgets (SRE).** The idea of tightening when a budget burns and loosening when it recovers is conceptually adjacent and we acknowledge it as inspiration; SRE error budgets gate *deploys/alerts* at the service level, not *per-action-type autonomy* of an agent, and produce no band or routing trichotomy.
- **Single-threshold confidence calibration in agent platforms.** Some agent platforms learn a *single* per-agent threshold from human approve/reject decisions (the immediate predecessor of this work). It is genuinely close but is **one threshold per agent**, **binary** (auto vs. review), and lacks the per-action-type band and the escalation zone.

We claim novelty only in the **combination**: per-action-type **bands** + the **trichotomy** + **rolling human-free recalibration** + the **emergent graduated-autonomy gradient**, made auditable.

---

5. Prior-Art Delta

Per novel feature: what the closest prior source has, what it lacks, and what THIS adds.

Prior source	What it has	What it LACKS	What THIS adds
Static confidence threshold (HITL)	A single cutoff `T`; act above, defer below.	Per-type granularity; adaptation; an in-between zone.	Per-action-type **bands** that adapt themselves; a third **escalate** zone.
Adaptive thresholding (control theory)	Continuous self-adjustment of one setpoint from error.	Per-action-type framing; a `(low, high)` **band**; a discrete **routing trichotomy**.	A discrete **auto/escalate/review** routing over a self-tuned band, scoped per action type.
Conformal / selective prediction	Calibrated abstain/reject at a target error rate.	Autonomy routing across **agent action types**; rolling per-type recalibration of an autonomy band.	Uses such a value as input; adds **per-action-type autonomy bands** recomputed from outcomes.
Bandits / RL autonomy	Learns when to act from reward.	Legibility; auditability; lightweight per-type bands; bounded clamps.	An **auditable, rule-legible band** that achieves earned autonomy without a policy network.
SRE error budgets	Tighten on burn / loosen on recovery, service-level.	Per-action-type agent autonomy; confidence bands; routing trichotomy.	The burn/recover *intuition* applied to **per-action-type agent autonomy bands** with routing.
Single-threshold agent calibration	Learns one per-agent threshold from approve/reject.	Multiple **action types**; a `(low, high)` **band**; an **escalate** zone.	**Per-action-type bands** + **escalation** + emergent **graduated autonomy**.

6. System Architecture

The mechanism is a thin **control plane** between the agent's decision step and its action executor. It is executor-agnostic (R10): it only needs the action type, a confidence value, and an outcome signal.

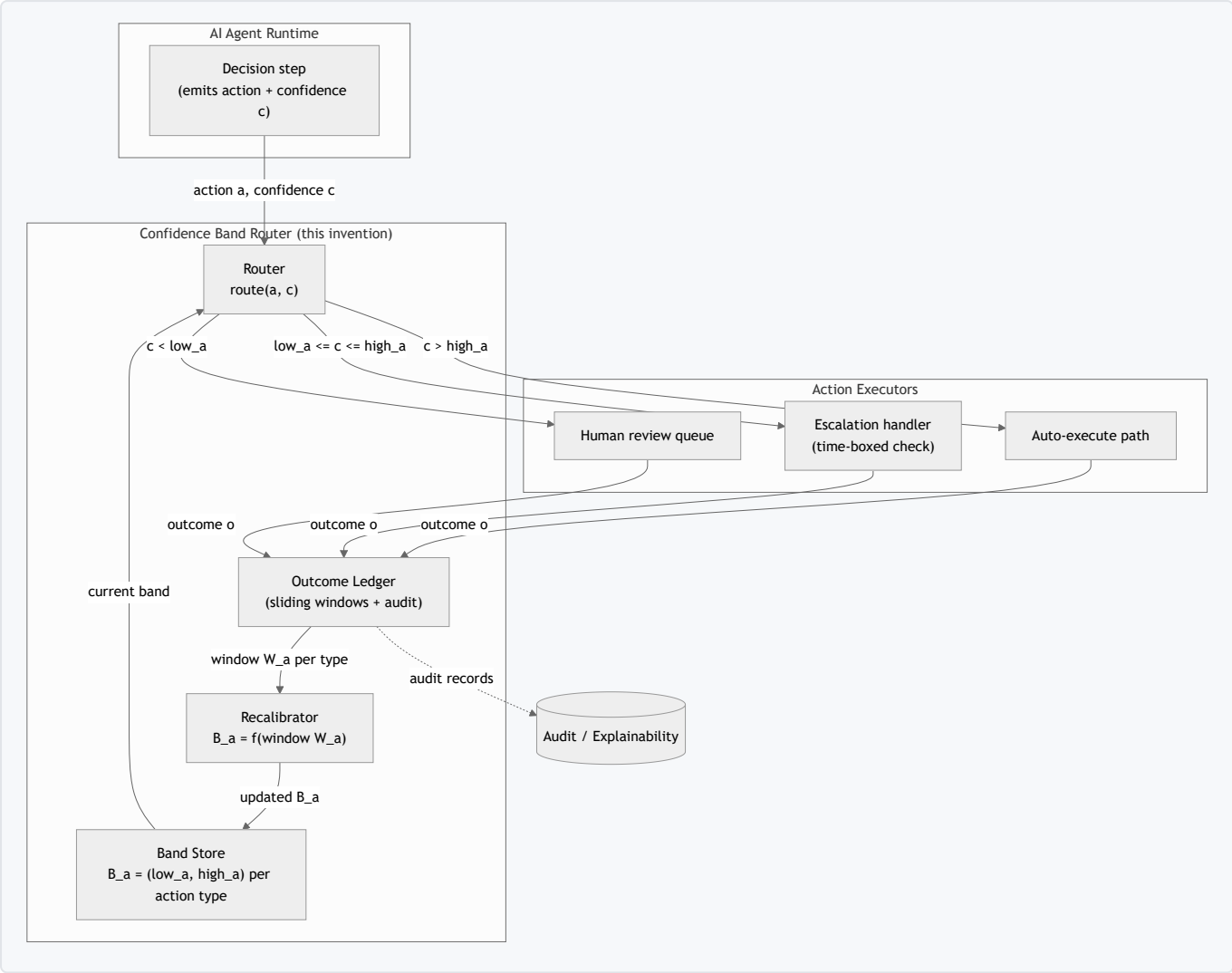

Figure 1 — System architecture. The router consults per-action-type bands; all three routes feed outcomes back to a ledger; the recalibrator updates bands from per-type sliding windows.

6.1 Components

Component	Responsibility
Router	Computes `route(a, c)` against the current band for action type `a`. Pure function of `(band, c)`.
Band Store	Holds `B_a = (low_a, high_a)` for each action type; default seed band for unseen types.
Outcome Ledger	Append-only record of `(action_type, confidence, route, outcome, ts)`; maintains the sliding window `W_a` per type.
Recalibrator	Recomputes `B_a = f(W_a)` per the update rule (§7.4), applying clamps, hysteresis, and min-samples gating.
Audit / Explainability	Read view over the ledger + band history: "why was action X auto-executed?"

6.2 Cross-cutting properties

- **Statelessness of routing:** routing is a pure function — easy to test, reproduce, and reason about.
- **Locality of calibration:** a failure in `email` tightens only the `email` band; `chat` autonomy is unaffected (R1).

- **Fail-safe bias:** any ambiguity or missing state routes toward review, never toward auto-execution (R11, §11).
- **Auditability:** every routing decision and every band change is recorded (R9).

---

7. Detailed Mechanics

7.1 Per-action-type state

For each action type `a` the system keeps:

```
BandState_a = {
  low:           number in [0,1],   // lower bound
  high:          number in [0,1],   // upper bound, high >= low
  window:        ring buffer of last N outcomes {success: bool, weight: number},
  sinceRecalc:   integer,           // outcomes since last recompute
  lastRecalc:    timestamp,
  successRateEwma: number,          // smoothed recent success rate
}
```

7.2 Recording an outcome

On completion of any routed action:

```
recordOutcome(a, { confidence, route, success, gradedScore? }):
  ledger.append({ a, confidence, route, success, ts: now() })
  st = state(a)
  weight = recencyWeight(now())            // recent outcomes weigh more
  st.window.push({ success, weight })       // ring buffer caps at N
  st.sinceRecalc += 1
  if st.sinceRecalc >= M or timerElapsed(a):
     recalibrate(a)
```

Outcomes may come from any path (auto/escalate/review). An escalated action that a human waves through counts as a success; one they correct counts as a failure. This is what lets a capability *earn* autonomy.

7.3 The routing trichotomy

```
route(a, c):
  b = band(a)                  // (low, high)
  if c >  b.high:  return AUTO
  if c <  b.low:   return REVIEW
  return ESCALATE              // low <= c <= high
```

- **AUTO** — execute immediately; record outcome asynchronously.
- **REVIEW** — enqueue for full human review; the human's decision is the outcome.
- **ESCALATE** — a lightweight, time-boxed check: a fast secondary signal (a cheaper verifier, a second model, a brief human glance, or a policy check). If the escalation resolves "ok within budget," execute; else fall to review. Escalation outcomes feed the window like any other.

7.4 Rolling recalibration (the core update rule)

```
recalibrate(a):
  st = state(a)
  if effectiveSamples(st.window) < minSamples: return   // R8 gate

  # weighted recent success rate (recency-weighted; R4, R6)
  sr = sum(w for outcomes where success) / sum(w for all outcomes)
  st.successRateEwma = ewma(st.successRateEwma, sr, alpha)

  # error vs target: positive when over-performing, negative when under
  err = st.successRateEwma - targetSuccessRate

  # tighten on failure (err<0 -> raise bounds), loosen on success (err>0 -> lower)
  delta = -learningRate * err           # note sign: more success => lower bounds

  newLow  = clamp(st.low  + delta, lowMin,  lowMax)
  newHigh = clamp(st.high + delta, highMin, highMax)

  # stability: cap per-step movement (hysteresis / max-step; R8)
  newLow  = boundedStep(st.low,  newLow,  maxStep)
  newHigh = boundedStep(st.high, newHigh, maxStep)

  # invariant: keep a minimum band width and low <= high (R7)
  (newLow, newHigh) = enforceOrdering(newLow, newHigh, minWidth)

  st.low = round3(newLow); st.high = round3(newHigh)
  st.sinceRecalc = 0; st.lastRecalc = now()
  audit('recalibrate', a, { from, to, successRate: sr })
```

Direction of adjustment. When recent success is **above** target (`err > 0`), `delta < 0` → both bounds **drop** → more actions land above `high` → **more autonomy** (the band *loosens*). When recent success is **below** target (`err < 0`), `delta > 0` → both bounds **rise** → more actions land below `low` or in the escalate zone → **less autonomy** (the band *tightens*). This is R6.

Why a band, not a point. Moving two bounds preserves an **escalation zone** of width `high - low`. Tightening can widen that zone (more escalation, fewer auto-executes) before it forces full review; loosening narrows it. The zone is the graceful middle gear between full autonomy and full review (R2, C5).

7.5 The emergent graduated-autonomy gradient

Run independently per action type, the rule produces, over time, a **gradient**:

- Capabilities the agent is reliably good at → bands sink → mostly **AUTO**.
- Capabilities with mixed outcomes → bands hover → mostly **ESCALATE**.
- Capabilities the agent is bad at, or that just regressed → bands rise → mostly **REVIEW**.

No operator picked these. The gradient is *earned*, *observable*, and *reverts* the instant outcomes degrade — a safety property (§11).

7.6 Configuration points (R12)

Parameter	Meaning	Illustrative default
`N` (windowSize)	Sliding-window length per type	100
`M` (recalcEvery)	Outcomes between recomputes	20
`minSamples`	Gate before first recompute	20
`targetSuccessRate`	Desired recent success rate	0.85
`learningRate`	Adjustment speed	0.05
`maxStep`	Max per-recompute bound movement (hysteresis)	0.03
`minWidth`	Minimum `high − low` (escalation zone)	0.05
`lowMin/lowMax`	Clamp on `low`	0.40 / 0.90
`highMin/highMax`	Clamp on `high`	0.55 / 0.98
`alpha`	EWMA smoothing on success rate	0.30
`decay`	Recency weighting of outcomes	0.98

All are settable per deployment and overridable per action type.

7.7 Recalibration state machine

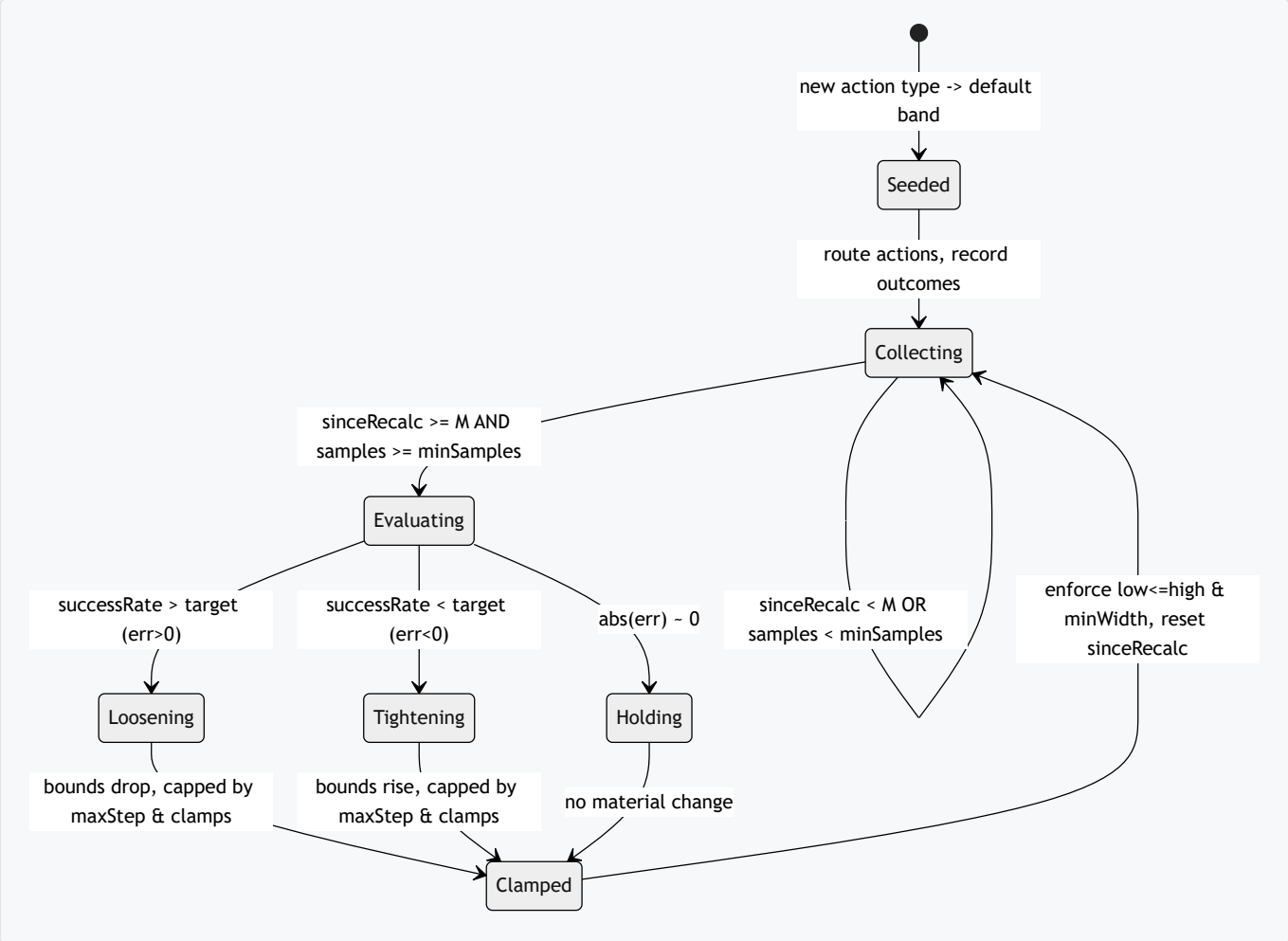

Figure 2 — Recalibration state machine for a single action type.

7.7 Error handling

Condition	Handling
Missing/NaN confidence	Treat as `0` → routes to REVIEW (fail-safe, R11).
Unknown action type	Seed with default band; do not block.
Ledger write failure	Still route (routing is in-memory); flag degraded audit; never auto-execute on lost outcomes.
Recalibrator exception	Keep last good band; emit alert; never widen autonomy on error.
Clock skew / out-of-order outcomes	Recency weight is monotone-bounded; reordering cannot push bounds beyond clamps.

---

8. Data Model

The mechanism needs three logical entities: the **band per action type**, the **outcome ledger** (the source of truth for windows and audit), and a derived **recalibration history**. A relational sketch (PostgreSQL-style) follows; an in-memory ring buffer mirrors the recent window for speed.

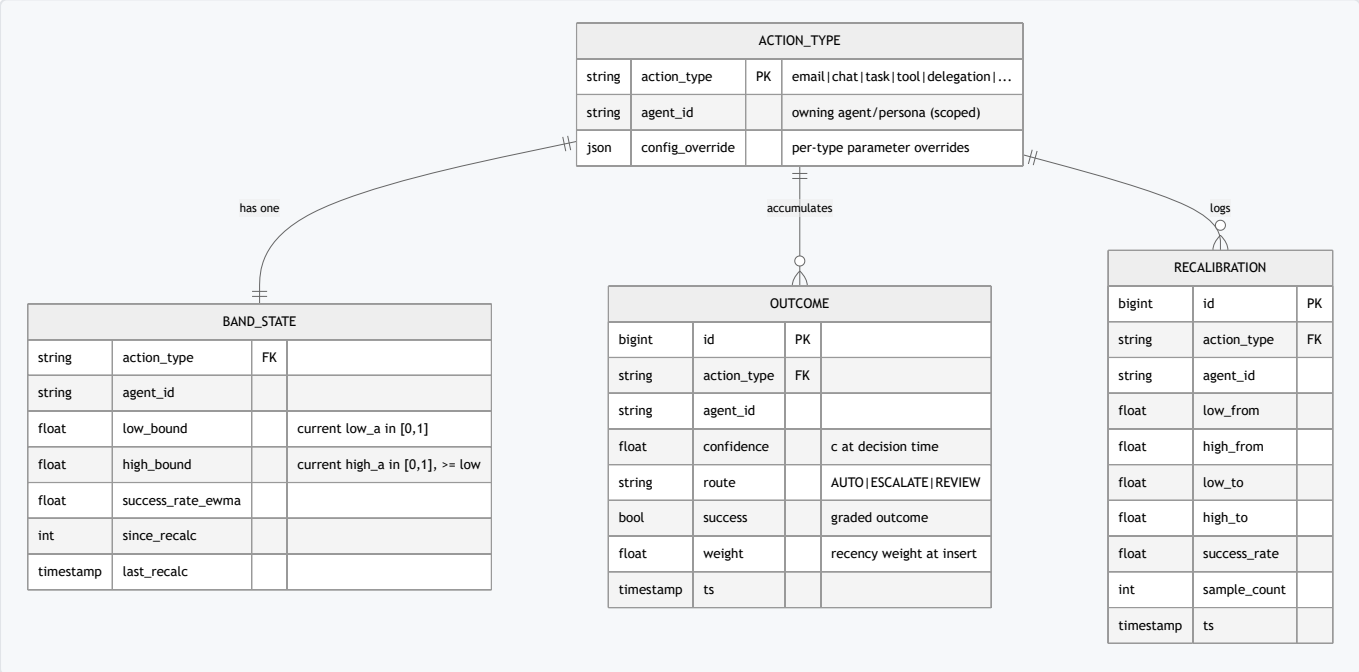

Figure 3 — Data model (ER-style). `BAND_STATE` is the live control state; `OUTCOME` is the append-only ledger that feeds windows; `RECALIBRATION` is the audit trail of band changes.

8.1 Schema notes

- **Scope key** is `(agent_id, action_type)` so bands are isolated per agent and per capability (R1).
- **`OUTCOME` is append-only** — never updated; the sliding window is the most recent `N` rows by `ts` for the scope. This makes recalibration reproducible from the ledger (R9).
- **`RECALIBRATION`** is derived/audit; it records every band transition with the success rate and sample count that justified it — the backbone of explainability.
- A materialized in-memory ring buffer (`window`) mirrors the last `N` outcomes so the hot path never touches the database.

9. Reference Implementation & Enablement

`src/` contains an **original, clean-room, dependency-free Node.js** implementation that reduces the invention to practice. It is illustrative — not production code — and reproduces no proprietary source.

9.1 What `src/` demonstrates

File	Demonstrates
`src/band-router.js`	The full mechanism: per-action-type band store, the routing trichotomy (`route`), outcome recording with a recency-weighted sliding window, and the rolling recalibration rule with clamps/hysteresis/min-samples.
`src/example.js`	A runnable scenario + self-check: drives several action types through streams of outcomes and prints how bands move and routing shifts; asserts the invariants (R6, R7, R11).
`src/README.md`	What it shows, how to run, and the clean-room/illustrative caveat.

9.2 How it enables the claims

- **R1/C1** — `bands` is a `Map<actionType, {low, high, ...}>`; each type is independent.
- **R2/C2** — `route()` returns one of `AUTO | ESCALATE | REVIEW`.
- **R3/R4/C3** — `recordOutcome()` appends to a per-type recency-weighted ring buffer; `recalibrate()` consumes it.
- **R5/R6** — `recalibrate()` runs with no human input and moves bounds with the sign rule (loosen on success, tighten on failure).
- **R7/R8** — `clamp()`, `boundedStep()`, `enforceOrdering()`, and the `minSamples` gate.
- **R11** — `route()` returns `REVIEW` for missing/NaN confidence; recalibration never widens autonomy on error.

9.3 Configuration

All parameters from §7.6 are constructor options with safe defaults, settable globally and overridable per action type — demonstrating R12 without hardcoding.

10. Worked Example / Scenario

Setup. An agent handles two action types: `chat` (low stakes) and `email` (high stakes). Both seed at `low=0.70, high=0.85`. `targetSuccessRate=0.85`, `learningRate=0.05`, `maxStep=0.03`, `minSamples=20`, `M=20`.

Phase 1 — chat goes well. Over 40 chat actions, recent success rate ≈ 0.95 (> target). Two recomputes drop the `chat` band toward `low≈0.64, high≈0.79`. More chat replies now exceed `high` → **more AUTO**. Chat autonomy was *earned*.

Phase 2 — email stumbles. Over 30 email actions, several drafts are corrected by reviewers; recent success rate ≈ 0.70 (< target). Recomputes raise the `email` band toward `low≈0.74, high≈0.89`. More emails now fall below `low` or into the escalate zone → **more REVIEW / ESCALATE**. Email autonomy *tightened* — and crucially, `chat` was untouched (R1).

Phase 3 — email recovers. Reviewers stop correcting; success climbs back above target; the `email` band loosens again, gradually (capped by `maxStep`, so no whiplash — R8).

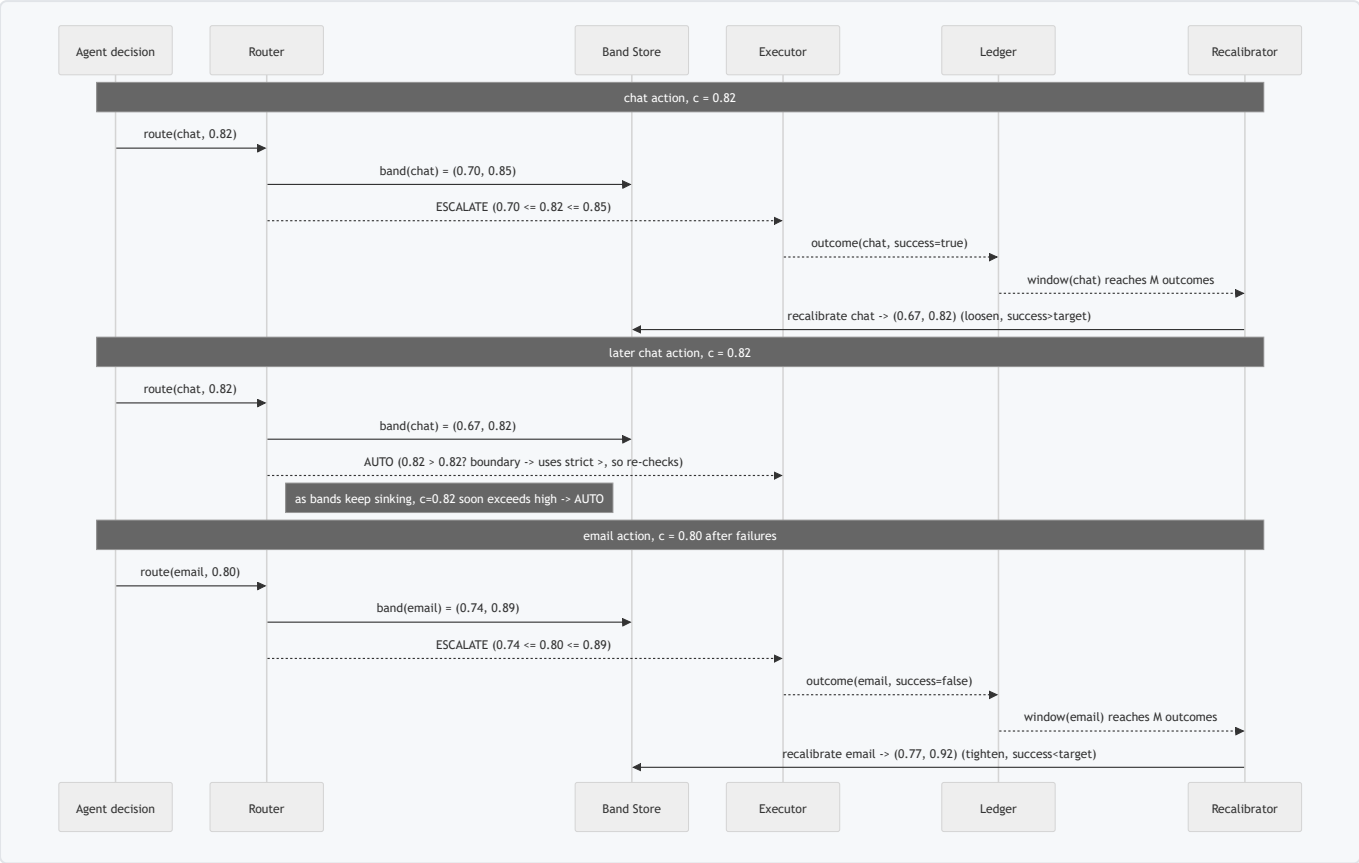

Figure 4 — Sequence for the worked example: the same confidence value is routed differently as each action type's band moves independently.

Observation. The *same* confidence (0.80–0.82) is treated as autonomous for `chat` and review-worthy for `email`, purely because of divergent observed competence — the graduated autonomy gradient (C5) in action.

---

11. Security, Safety & Failure Modes

11.1 Posture: fail-safe (fail toward review)

The system is biased so that **every failure mode reduces autonomy**, never increases it (R11). Missing data, errors, lost outcomes, and clock anomalies all route toward ESCALATE/REVIEW, never silent AUTO.

11.2 Failure-mode table

Failure mode	Effect if unguarded	Mitigation
Confidence inflation (upstream over-confident)	More AUTO than warranted.	Outcomes from those AUTOs feed the window; failures tighten the band, self-correcting. Clamp `highMax < 1.0` so nothing is *unconditionally* auto.
Window poisoning (a burst of easy successes loosens a band, then hard cases slip through)	Temporary over-autonomy.	`maxStep` + EWMA smoothing limit how fast bands sink; `minWidth` preserves an escalation buffer.
Reward hacking via escalation (escalator rubber-stamps)	Inflated success rate.	Treat escalation approvals conservatively (configurable lower weight); audit escalation-vs-review agreement.
Thrashing (bands oscillate)	Unstable routing.	`minSamples` gate, `maxStep` hysteresis, EWMA — see R8.
Lost outcomes (ledger write fails)	Stale window → wrong band.	Never widen autonomy on missing outcomes; mark audit degraded; on persistent loss, freeze bands.
Cross-type contamination	One bad type drags others.	Hard isolation: state keyed per `(agent, action_type)` (R1).
Cold start	No data → over- or under-autonomous.	Conservative seed band + `minSamples` gate → ESCALATE/REVIEW until evidence accrues.

11.3 STRIDE-style view (control-plane scope)

Threat	Concern	Note
Spoofing	Forged outcomes alter bands.	Authenticate the outcome source; outcomes are control-plane data and must be trusted/signed.
Tampering	Editing ledger to inflate success.	Append-only ledger; recalibration auditable from raw rows.
Repudiation	"Why did it auto-send?"	Every routing decision + band change recorded (R9).
Info disclosure	Confidence/outcomes leak task content.	Store metadata, not payloads; redact at ledger boundary.
Denial of service	Recalc storms.	Recalc is `O(N)` per `M` outcomes; gated by `minSamples`/timer.
Elevation	Forcing AUTO inappropriately.	No path widens autonomy except the bounded recalibration rule under clamps.

12. Standards & Framework Mapping

This is **semantic alignment**, not a claim of certified compliance.

Framework / Standard	Relevant clause / concept	How this aligns
NIST AI RMF 1.0	*Manage* / *Measure* — continuous monitoring & graduated control of AI risk.	Per-action-type bands are a measured, continuously-managed autonomy control with an auditable trail.
ISO/IEC 42001 (AI management systems)	Operational controls & continual improvement.	Rolling recalibration is a documented, automatic continual-improvement loop with records.
ISO/IEC 23894 (AI risk management)	Risk treatment proportional to context.	Higher-stakes action types naturally retain tighter bands → more review.
EU AI Act (high-risk: human oversight, Art. 14)	Effective human oversight; ability to intervene.	The trichotomy preserves human-review and escalation gates; autonomy is bounded and reversible.
SRE / error-budget practice	Tighten when budget burns, loosen on recovery.	The update rule is the per-action-type analogue applied to autonomy.
Conformal / selective prediction	Calibrated abstention at a target error rate.	Compatible upstream confidence source; the band governs *autonomy*, the predictor governs *coverage*.

13. Evaluation Methodology

Numbers here are **illustrative**; the contribution is the methodology.

13.1 Dimensions & metrics

Dimension	Metric	Interpretation
Review-burden reduction	% of actions auto-executed vs. single-threshold baseline.	Higher = less human load (target intuition: ~40% reduction once bands settle — *illustrative*).
Safety / error leakage	Failure rate among AUTO-routed actions.	Must stay ≤ a configured ceiling; bands should self-correct breaches.
Adaptivity / convergence	Recomputes (or outcomes) until bands stabilize after a regime shift.	Lower = faster earned/retracted autonomy.
Stability	Band variance / oscillation amplitude over a stationary window.	Lower = no thrashing (validates R8).
Isolation	Cross-type band correlation under a single-type shock.	≈ 0 confirms R1.
Calibration quality	Realized success rate vs. `targetSuccessRate` per type.	Close = the loop tracks its target.

13.2 Protocol

1. **Baseline:** single static threshold per agent; record review burden and AUTO error rate.
2. **Treatment:** per-action-type bands with the update rule.
3. **Regime shifts:** inject a competence drop on one action type; measure convergence and isolation.
4. **Ablations:** remove the escalate zone (binary), remove `maxStep`, remove `minSamples` — observe degraded safety/stability to demonstrate each guard earns its place.
5. **Report** all six metrics with confidence intervals; flag every number as illustrative unless measured on a named dataset.

14. Novelty & Inventive Claims

The following claims are stated in prose for clarity of the disclosure. They are published as **prior art**; no patent is sought.

Claim 1 (independent)

A method comprising: maintaining, for each of a plurality of action types associated with an AI agent, a confidence band consisting of a lower bound and an upper bound; on completion of an action of a given action type, recording an outcome and recomputing said band for that action type as a function of a sliding window of said outcomes; and routing a subsequent action of said action type to one of (i) automatic execution if a confidence value associated with the subsequent action exceeds said upper bound, (ii) human review if said confidence value is below said lower bound, or (iii) escalation otherwise; characterized in that said band is recomputed without human intervention.

Dependent claims

2. The method of claim 1, wherein recomputing the band comprises **lowering both bounds when a recent success rate over the sliding window exceeds a target and raising both bounds when the recent success rate is below the target**, thereby loosening autonomy on demonstrated competence and tightening it on observed failure.

3. The method of claim 1, wherein the plurality of action types comprises at least two of **email, chat, task creation, tool invocation, and delegation**, each maintaining an **independent** band and an independent sliding window.

4. The method of claim 1, wherein outcomes in the sliding window are **recency-weighted** such that more recent outcomes contribute more to the recomputation than older outcomes.

5. The method of claim 1, wherein the recomputation is gated by a **minimum sample count** such that no band is changed until the window contains at least said number of outcomes.

6. The method of claim 1, wherein the recomputation is performed **every M completed actions** of the action type, where M is configurable per deployment and per action type.

7. The method of claim 1, wherein each recomputed bound is **clamped to a configured `[min, max]` range** and the lower bound is constrained to remain no greater than the upper bound.

8. The method of claim 1, wherein the magnitude of change to each bound per recomputation is **limited to a maximum step**, providing hysteresis that prevents oscillation of the routing behavior.

9. The method of claim 1, wherein the **escalation** route triggers a time-boxed secondary check — comprising at least one of a cheaper verifier model, a second model, a policy check, or a brief human confirmation — whose result is itself recorded as an outcome for the action type.

10. The method of claim 1, further comprising maintaining an **append-only outcome ledger** recording, per action, the action type, the confidence value, the route taken, and the outcome, from

which the band recomputation is **reproducible**.

11. The method of claim 1, further comprising recording, for each band change, a **recalibration audit record** comprising the prior bounds, the new bounds, the success rate, and the sample count, supporting an explanation of why a given action was auto-executed or routed to review.

12. The method of claim 1, wherein a previously unseen action type is **seeded with a conservative default band** and routed to escalation or review until its window accumulates at least the minimum sample count.

13. The method of claim 1, wherein, upon a **missing, invalid, or non-numeric confidence value, or upon a failure to record an outcome**, the routing decision is biased toward human review and the band is not widened, providing a fail-safe posture.

14. The method of claim 1, wherein the bands across the plurality of action types collectively form a **graduated autonomy gradient** that emerges from observed outcomes without manual threshold tuning, such that the agent becomes fully autonomous on some action types while remaining review-gated on others.

15. The method of claim 1, wherein the band, the sliding window, and the recomputation parameters are **scoped per (agent, action type)** so that a competence change in one action type of one agent does not alter the bands of other action types or other agents.

---

15. Limitations & Threats to Validity

- **Quality of the confidence input.** The method governs autonomy but does not *produce* the confidence value; a pathologically mis-calibrated upstream signal limits how well bands can track competence (mitigated, not eliminated, by the outcome feedback loop).
- **Outcome attribution.** "Success" must be observable and attributable to the action; ambiguous or delayed outcomes weaken recalibration.
- **Proximity to prior art.** Single-threshold confidence calibration and classical adaptive control are genuinely adjacent; novelty rests on the **combination** (per-type bands + trichotomy + rolling human-free recalibration
    - emergent gradient), not on any single element.
- **Parameter sensitivity.** Poorly chosen `learningRate`/`maxStep` can over- or under-react; defaults and the `minSamples`/`maxStep` guards reduce but do not remove this.
- **Intentionally withheld.** Production-specific tuning curves, integration glue with any particular agent runtime, and any proprietary confidence estimator are out of scope and deliberately not disclosed.

---

16. Future Work & Open-Source Reference App

A small **open-source "Confidence Band Router"** service can package this mechanism as a sidecar any agent runtime calls before executing an action. See docs/OPEN-SOURCE-APP.md for the plan, minimal architecture, the invention mapping, and a generic Kubernetes/AKS deployment sketch. Roadmap:

1. Library (the `src/` core) → 2. HTTP/gRPC service wrapper → 3. Pluggable outcome ledger (in-memory → SQL) → 4. Dashboards over the recalibration audit → 5. Optional graded outcomes and per-tenant scoping.

---

17. Conclusion

Autonomy should be **earned per capability and revoked the moment competence slips** — automatically, legibly, and auditably. Replacing a single static threshold with **per-action-type confidence bands**, recomputed **without human intervention** from a **sliding window of outcomes**, and routed through an **auto / escalate / review trichotomy**, yields exactly that: a self-tuning, graduated autonomy gradient that is safe by construction (fail toward review) and explainable by record. We publish it openly so it stays that way — free for all to build on.

Appendix A — Prior-Art Landscape (well-trodden vs candidate-novel)

Well-trodden (treated as background, not claimed):

- Single static confidence thresholds in HITL automation.
- Continuous adaptive thresholding / setpoint adjustment in classical control.
- Conformal / selective prediction with a reject option.
- Bandit/RL approaches to when-to-act.
- SRE error-budget tighten/loosen dynamics.
- Single per-agent learned thresholds in agent platforms.

Candidate-novel (the claimed combination):

- Per-action-type `(low, high)` **bands** instead of a point threshold.
- The **auto / escalate / review** routing **trichotomy** keyed to the band.
- **Rolling, human-free recalibration** per action type from a sliding window, tightening on failure and loosening on success.
- The **emergent graduated-autonomy gradient** across action types, with bounded clamps and hysteresis, made **auditable**.

Honesty attestation. The prior-art searches behind this document are **directional, not exhaustive**. They reflect a good-faith review of public techniques known to the author as of 2026-06-25. No representation is made that every relevant reference has been identified. The purpose of this disclosure is to *add* to the public record, not to adjudicate the novelty of any specific later claim.

Appendix B — Glossary

Term	Definition
Action type	A category of action an agent can take (email, chat, task, tool, delegation, …).
Confidence (`c`)	A scalar in `[0,1]` produced upstream estimating how likely an action is correct/safe.
Band (`low, high`)	The per-action-type pair of bounds governing routing.
Routing trichotomy	The three-way decision: AUTO (`c > high`), REVIEW (`c < low`), ESCALATE (otherwise).
Sliding window (`W_a`)	The most recent `N` outcomes for action type `a`.
Recalibration	The human-free recomputation of a band from its window.
Tighten / loosen	Raise bounds (less autonomy) / lower bounds (more autonomy).
Graduated autonomy gradient	The emergent spread of autonomy across action types from observed competence.
Escalation	A time-boxed secondary check between full autonomy and full review.
Outcome ledger	The append-only record of routed actions and their outcomes.

Appendix C — Reference-Implementation Index

Path	Role
`src/band-router.js`	Core: band store, routing trichotomy, outcome recording, sliding window, rolling recalibration with clamps/hysteresis/min-samples.
`src/example.js`	Runnable scenario + self-check asserting R6/R7/R11 invariants.
`src/README.md`	What it shows, how to run, clean-room/illustrative caveat.

Appendix D — Defensive-Publication Deposit & Timestamp

- **Publication date:** 2026-06-25.
- **Publisher:** Gus IT LLC (Florida, USA).
- **Author:** Gustavo Assuncao, PhD.
- **Version:** 1.0.
- **Deposit channel:** To be assigned (IP.com / Zenodo / arXiv) — establishes a public, dated, citable prior-art record. No DOI is asserted at time of writing.
- **Statement:** This disclosure is **intentionally public** to establish prior art and to **bar later patenting by others** of the techniques described herein. The AGPL-3.0-or-later license adds an express patent grant over the author's own contribution. Nothing herein admits or grants any third party's rights.

Copyright 2026 Gus IT LLC (Florida, USA). Licensed under AGPL-3.0-or-later.