Curation Gate Kit

What this is. The human gate, made operational: three checklists (new-candidate, update-DIFF, retirement), a disposition rubric with reason codes, the edit-distance method, and the curator role definition — the machinery that lets a person govern in minutes what the AI drafts in seconds.

When you need it. The moment anything probabilistic can propose content — which, if you're using Module 3's prompts or any AI drafting tool, is now.

Which maturity level it serves. The Level 2→3 climb (installing the gate) and the Level 3→4 climb (making the gate auditable).

---

1. The gate is the architecture

One rule holds this entire operating model together: **nothing probabilistic writes to the system of record without passing a human gate.** Not the first draft, not the "small" update, not the automated cleanup. The AI proposes; the human disposes.

This is not ceremony. As AI answers get more polished, tolerance for their remaining errors collapses — your system will be judged by its worst confident error. The gate is where confidence gets earned. It is also where it gets earned *cheaply*: a curator disposing of a well-grounded candidate spends minutes; an organization recovering from a confidently wrong published answer spends far more.

Three gates cover every write:

Gate	Guards against	Checklist
New-candidate gate	Wrong, duplicate, or unworthy content entering the record	`checklists/new-candidate-gate.md`
Update-DIFF gate	Silent corruption of knowledge that was already trusted	`checklists/update-diff-gate.md`
Retirement gate	Dead knowledge lingering as live truth — or getting deleted instead of archived	`checklists/retirement-gate.md`

The checklists are printable one-pagers. Laminate them, pin them, or paste them into your review tool's description field — the medium doesn't matter; the discipline does.

2. The disposition rubric

Every candidate leaves the gate with exactly one decision and one reason code. No "parked," no "pending forever" — a queue that silts up is a gate that has failed.

Accept paths:

Code	Decision	Meaning
A1	Approve as-is	Draft was right; zero or trivial edits
A2	Edit-approve	Curator corrected or tightened, then approved — log edit-distance
M1	Merge	Content belongs inside an existing article — merge it there, cite the target KO- id, close the candidate

Reject codes (the reason *is* the telemetry):

Code	Meaning	What a spike of it tells you
R1	Not knowledge-worthy — a one-off, no reuse potential	Your capture trigger fires too eagerly; raise the threshold
R2	Duplicate — covered by an existing article (cite it)	Your drafting flow needs a dedupe pre-check (Module 3), or findability is failing your own agents
R3	Inaccurate or unverifiable content	Drafting prompt or source payload quality problem — check the edit-distance trend too
R4	Policy or privacy risk (PII, unsupported claims, legal exposure)	Tighten the synthesis prompt's scrub instructions; consider a pre-gate automated screen
R5	Out of scope or wrong audience	Capture trigger is pulling from the wrong queues or domains
R6	Insufficient demand evidence	Recurrence detector threshold too low — you're polishing content nobody asked for

Reject reasons are logged in the decision log (Module 5) and rolled up in the workbook (Module 6). A gate without reason codes is a bouncer without an incident book: the same problems keep arriving and nobody knows why.

3. Edit-distance: the self-calibrating benchmark

What it is. A measure of how much the curator changed an AI draft before approving it. Tracked over time, it is the single most honest indicator of your AI drafting quality — and almost nobody measures it.

Why it matters. The trend line is the point. Falling edit-distance = your prompts, templates, and source payloads are improving; the machine is learning your standards. A sudden spike = a model change, a prompt regression, or a new content domain the prompts haven't met. Either way, you *see* it — quantitatively, before your customers do.

Three capture tiers — use the heaviest one your tooling allows:

Tier	Method	Effort
T1 — Judgment scale	Curator logs 0–3 at disposition: **0** untouched · **1** light (wording, formatting; <10% of words) · **2** moderate (a section rewritten; 10–35%) · **3** heavy (restructured or re-researched; >35%)	Five seconds; works with zero tooling
T2 — Word-diff %	Paste draft and final into the workbook's tracker tab (Module 6); it computes changed-words ÷ total-words	Under a minute
T3 — Automated	Your pipeline computes a character- or token-level ratio at commit time and logs it with the disposition	Zero marginal effort once built

Start at T1 on day one — a subjective 0–3 logged consistently beats a precise number logged never. Note which tier produced each figure; don't trend T1 and T2 numbers on the same chart.

4. The curator role

Curation is a role, not a job title — in most teams it's 2–3 senior agents or a knowledge lead wearing the hat for a few hours a week.

The role card:

- **Owns**: dispositions at all three gates, within SLA; the health of the queue; edit-distance logging; escalating what they can't judge.
- **Does not own**: writing articles from scratch (that era is over); fixing the product; approving content outside their domain.
- **Skills**: domain fluency first, writing second. A curator must be able to *recognize* a wrong resolution; they don't need to be the one who found it — that's what escalation is for.
- **Time budgets** (design targets): clean approve ≤ 3 minutes · edit-approve ≤ 10 · update-DIFF review ≤ 5 · rejection ≤ 2. If medians run over, the problem is upstream (draft quality, payload quality, candidate volume) — fix that, don't stretch the curators.
- **SLA**: every candidate dispositioned within **2 business days** of entering the queue. Fresh candidates are cheap to judge; stale ones require re-research, which reintroduces the capture tax through the back door.

Sizing rule of thumb (validate against your own first month, then re-plan):

Assisted answer volume / month	Expect candidates / week	Curator hours / week
~1,000	10–25	2–4
~5,000	40–80	6–12 (split 2–3 curators by domain)
~20,000	120–250	dedicated knowledge lead + domain curators

Anti-bottleneck rules. If the queue exceeds one week of throughput: first raise the capture threshold (fewer, better candidates), then add a curator in the flooding domain, then batch dispositions into a daily 30-

minute block. Never "fix" a backlog by lowering the quality bar — a gate that waves things through is decoration.

Escalation. When a curator can't verify accuracy, the candidate routes to the would-be `owner` (the domain SME) with a 5-business-day cap; the SME verifies *facts*, the curator still executes the disposition. Two people, one decision, no committee.

5. What the gate feeds downstream

Every disposition writes one row to the decision log (Module 5) — date, actor, object, decision, reason code, edit-distance. From that single stream, Module 6 derives the gate's health metrics: throughput, queue age, acceptance rate, dedupe rate (R2 share), edit-distance trend, and reject-reason mix. You get an auditable gate and its analytics from one habit: *log the disposition, every time.*

6. Adapter notes

- **Git**: the gate *is* a pull request. Candidate branch → PR → curator reviews the diff → merge (approve) or close-with-comment (reject, reason code in the comment). The decision log falls out of PR history nearly for free.
- **Zendesk / Salesforce / Freshdesk**: run the queue as a view on a custom object or ticket type ("KB candidate"); disposition fields (decision, reason code, edit-distance) as custom fields; the update-DIFF gate is a side-by-side of current article vs. proposed text in the work item.
- **Notion / Confluence**: candidates as entries in a "Queue" database with disposition properties; promotion = moving the page into the KB space and stamping normal frontmatter properties.

What this module assumes from its siblings: articles typed per Module 2 (any typed template works), lifecycle states per Module 5. Module 3's prompts produce the candidates this gate consumes; Module 6 turns this gate's logs into dashboards.