The Contextkeeping Field Guide

Start here — the operating model behind every module in this toolkit

What this is. The ten-thousand-foot view of the Contextkeeping operating model, and the map of how this toolkit's modules implement it. Every module you bought — whether one or all seven — assumes the fifteen minutes you're about to spend here.

When you need it. First. Before any template, checklist, or spreadsheet.

Which maturity level it serves. All of them. This is the document that tells you which level you're at and which module to open next.

1. Why this toolkit exists

Knowledge-Centered Service earned its reputation. Its core insight — that knowledge should be captured as a byproduct of solving problems, by the people solving them, at the moment they solve them — is as true today as it was twenty-five years ago.

But KCS rests on two structural assumptions, and in the last three years both quietly died.

Assumption one: humans write the articles. Every struggling KCS implementation fails at the same point — the capture tax. The methodology asks an agent, mid-queue, to stop resolving tickets and write. Leadership says knowledge matters, then measures ticket throughput. The agent does the math. Meanwhile, a resolved case already *contains* the knowledge: problem, environment, troubleshooting path, resolution. But, today the labor of turning a conversation into an article is now nearly free. AI drafts. The human's job shrinks to a single decision: *is the content accurate?*

Assumption two: humans read the articles. Look at who actually consumes your knowledge base today. The agent-assist panel. The chatbot on your help center. The copilot your customers wired into their own service desk. Increasingly, the "reader" is a machine assembling an answer for a human who will never see the article page — and soon, for another machine. Everything KCS taught about writing for readers (findability, scannability) was designed for human eyes. Machines need different things: self-contained sections, explicit applicability metadata, stable identifiers, freshness signals.

What survives is the part everyone treated as vegetables: **governance**. In the AI era it's the load-bearing wall — because as AI answers get more polished, tolerance for their remaining errors collapses. A confident answer engine citing your stale article burns trust you don't get back. Garbage in, confident garbage out. Retrieval technology cannot rescue an ungoverned corpus.

This toolkit has a name for that dynamic: **the Precision Paradox**. The more fluent and confident AI answers become, the more each remaining error costs — polish raises the price of being wrong, because

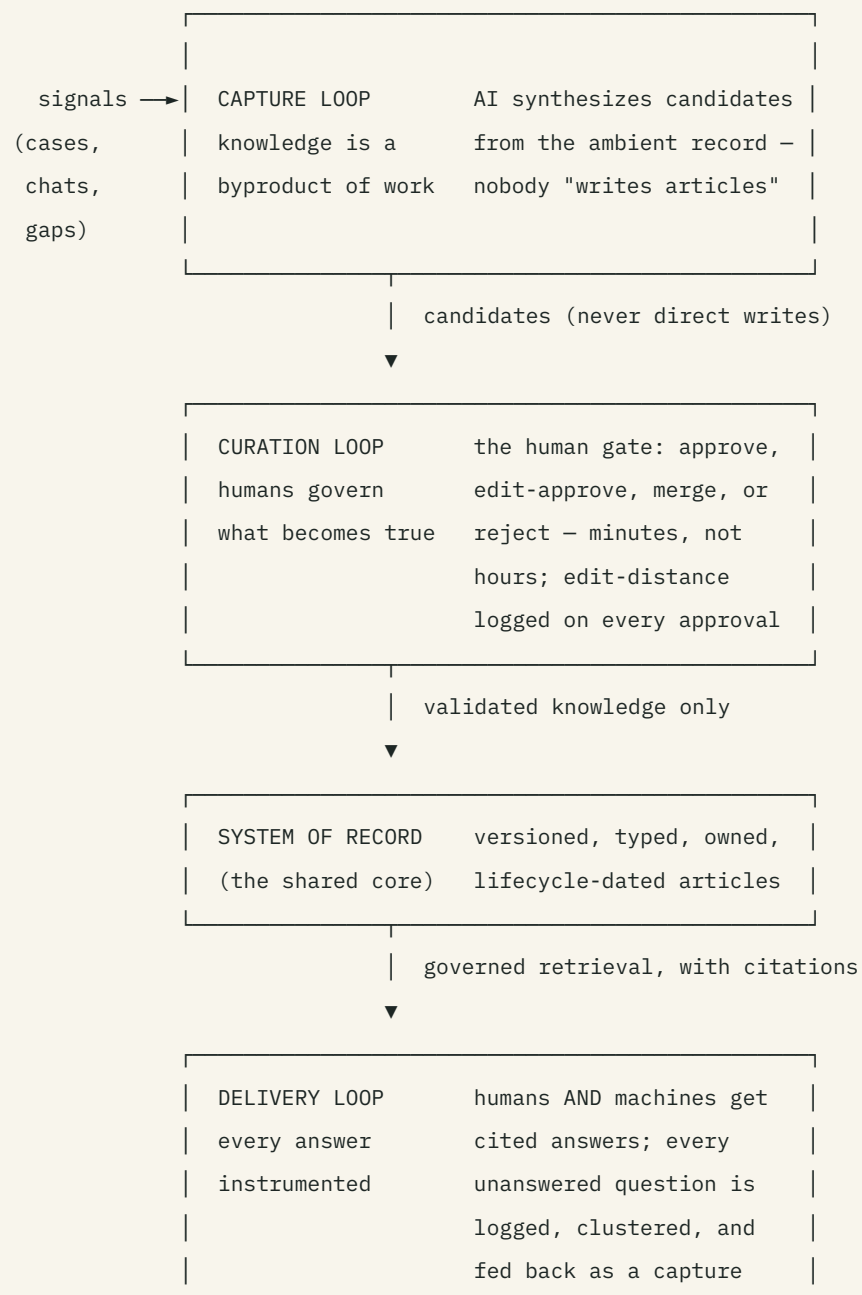
nothing in the delivery signals doubt. It is why the gate exists (Module 4), why nothing probabilistic writes to the record ungated (Module 5′s headline rule), and why several modules will invoke the term as shorthand.

Keep the soul. Replace the machinery. That replacement is what this toolkit installs.

---

2. The Governed Knowledge Loop

Classical KCS runs two loops (Solve and Evolve). The Contextkeeping model reorganizes the work into **three loops around a shared core**:



- **The Capture Loop** *(evolution of Solve)*: a resolved case, a recurring question, a product change, or a detected gap fires a capture event. AI synthesizes a **candidate** from the ambient record. The human contribution shrinks to one signal — "this is knowledge-worthy" — and even that can be automated. Candidates never touch the system of record directly.
- **The Curation Loop** *(evolution of Evolve, front half)*: every candidate passes automated pre-checks, then a human disposition. Updates to *existing* articles pass a before/after DIFF review. Curation also owns lifecycle: review dates, retirement, archive. This is where your people spend their small time budget.
- **The Delivery Loop** *(the reuse portion of the Solve loop entry point)*: knowledge is delivered to humans and machines through the same governed retrieval layer, with citations and applicability filters enforced at query time. Every delivery event emits telemetry — including, most valuably, **what was asked and not answered**. The system tells you what's missing. The loop closes itself.

One sentence: *AI drafts, humans approve, machines and customers consume, and the system tells you what's missing — without asking your agents to become writers.*

---

3. The operating rules

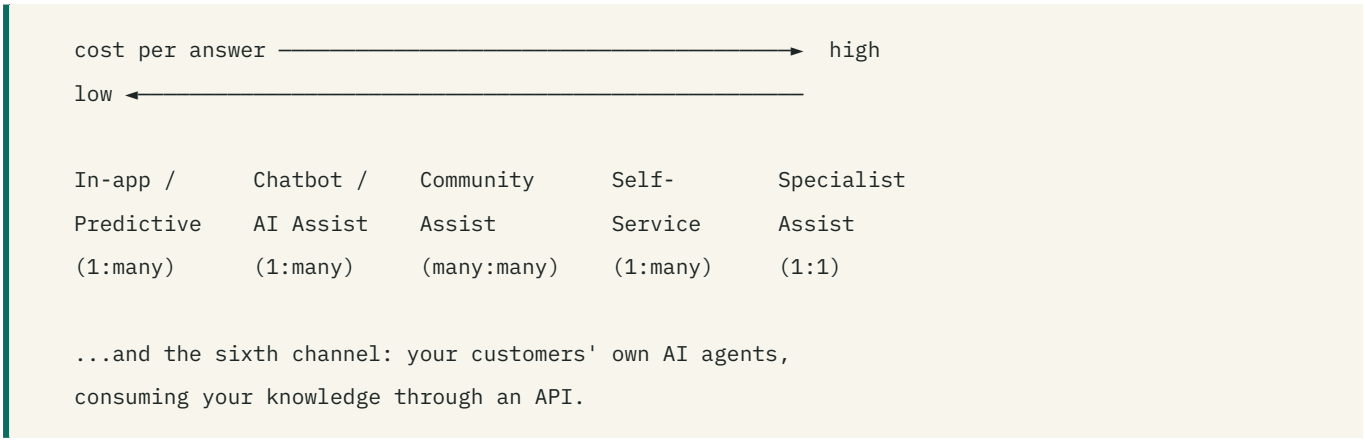
Six rules, in priority order. When a tool, vendor, or shortcut conflicts with one of these, the rule wins.

1. **Boring core, smart edge.** A deterministic, versioned, governed system of record at the center; probabilistic AI only at the edges — drafting on the way in, answering on the way out. Nothing probabilistic writes to the record without passing a gate. The AI layer gets swapped every time the model landscape shifts; the core compounds in value.
2. **Capture is ambient; curation is the human job.** Any process that asks a solver to author from scratch is dead on arrival. Any process that asks them to approve a well-grounded draft in ninety seconds survives contact with a real queue.
3. **Structure is a gradient, not a religion.** Enforce the minimum structure that makes content machine-operable — typed articles, stable IDs, explicit metadata, explicit links — and climb toward richer semantics only as scale demands (see §5). You do not need an enterprise CMS to start. You need discipline and a git repo.
4. **Trust is architectural.** Provenance, versioning, applicability, lifecycle state, and review dates are first-class fields, not afterthoughts — because your system will be judged by its worst confident error.
5. **Every consumer is instrumented.** Page views, answer citations, channel-costed answer events, unanswered queries, reviewer edit-distance: all of it flows back as telemetry. Gaps and drift are *detected*, not discovered.
6. **Rules and records are separate layers.** What *should* be true (policies, thresholds, lifecycle rules) lives apart from what *did* happen (an append-only record of decisions and overrides). Auditing the second is how the first evolves.

4. Measurement doctrine: cost per answer

The classic way to justify a knowledge program is deflection — counting tickets that *didn't* happen. Deflection invites attribution fights, values only the ticket queue, undervalues every other place your knowledge does work, and actually works against itself in traditional support org metrics (seen as time-to-close increases; a result of self-service success). This toolkit keeps a deflection view (some finance teams budget by it) but it is not the headline.

The headline is **cost per answer**, measured issue-centrically across every channel that delivers answers:



Three ideas to internalize:

- **Shift left.** One captured solution, delivered through channels left of the assisted queue, answers the same issue hundreds or thousands of times at a unit cost one to three orders of magnitude lower. Knowledge capture is the mechanism that moves answers left. The program's financial story is the **blended cost per answer** — total answer spend divided by total answers, across all channels — trending down as the **shift-left ratio** (share of answers delivered left of assisted support) trends up.
- **Issue-centric, not case-centric.** A case is one 1:1 interaction. An issue is the thing that gets answered N times across every channel. Up-level your measurement to issues and the whole program's value becomes visible in one number instead of scattered across channel silos.
- **Expect the metric inversion.** As shift-left succeeds, the assisted queue keeps only the new, hard problems. Time-to-resolve rises. Complexity rises. First-time-fix falls. These numbers going "the wrong way" is *evidence the strategy is working* — the easy volume left the queue. A leader who doesn't expect this will kill the program at exactly the moment it starts succeeding. Module 6 ships a pre-written one-page brief for your executives on precisely this.

Module 6 (the Measurement Workbook) operationalizes all of it: channel unit costs, blended cost per answer, shift-left ratio, known-vs-new mix, a scenario modeler, and the curation-inclusive **cost per governed answer**.

5. The structure gradient

You do not adopt "the full architecture" on day one. You climb:

Rung	Content model	Right for
1 — Typed Markdown	Markdown + YAML frontmatter in git (or your KB's equivalent); per-type templates; stable IDs; explicit links; controlled tag vocabulary	Teams of 5–50 agents; corpora to ~2K articles
2 — Linked knowledge	Rung 1 + enforced cross-references; link graph queryable; schema validation in CI; formalized taxonomy	50–200 agents; multi-product; 2K–20K articles
3 — Semantic knowledge	Structured components; ontology-backed relationships; knowledge-graph retrieval	Enterprise; regulated; agentic automation at scale

The promise: **start on rung 1 in weeks, and nothing you build is thrown away climbing to rung 3.** Git supplies versioning, diffing, audit trail, and rollback for free. Documentation-as-code is the DIYer's component CMS, and it is genuinely sufficient at rungs 1–2. This toolkit's Module 2 templates *are* rung 1, ready to commit.

---

6. The maturity model, condensed

Six levels. Your placement is set by your **weakest** dimension — you cannot skip levels, and bolting Level-4 tooling onto Level-1 content produces expensive hallucinations with your logo on them.

- **Level 0 — Institutional.** Tacit knowledge in heads, closed tickets, and chat scrollback. "Check with an SME" (or worse, someone's explicit name) is a documented process.
- **Level 1 — Documented.** A KB exists; writing is "when you get a chance"; content decays silently.
- **Level 2 — Process-driven.** Classical KCS or equivalent. Real maturity — and where certified programs are stuck, because it scales linearly with human effort.
- **Level 3 — AI-assisted.** AI drafts from case payloads; humans curate through a gate; edit-distance tracked; per-channel cost per answer measured.
- **Level 4 — Governed.** True system of record; DIFF-gated updates; machine delivery with citations and refusals; gap detection closing the loop; blended cost per answer on a dashboard.
- **Level 5 — Agentic.** Orchestrated workflows run capture-through-publish for standard cases; humans govern exceptions and policy; your knowledge serves your own AI agents and your customers'.

Module 1 is the scored instrument that places you honestly, dimension by dimension. It sits first in this toolkit for the same reason it should sit first in your program: you cannot pick your next move until you know where you stand — and you cannot skip levels once you do.

---

7. The toolkit map — which module, in what order

Module	What it installs	Loop
1. Maturity Self-Assessment	Scored placement across six dimensions + a roadmap builder — where you are, and therefore what's next	Diagnostic
2. Article Standards & Template Pack	The typed system-of-record seed: five templates, metadata schema, machine-legibility standards	Capture → core
3. AI Drafting Prompt Library (living module)	Candidate synthesis from tickets/chats/calls; dedupe and DIFF-proposal prompts; an evaluation protocol	Capture
4. Curation Gate Kit	The three gates (new / update-DIFF / retirement), disposition rubric with reason codes, edit-distance method, curator role card	Curation
5. Governance & Lifecycle Pack	Content policy, lifecycle states, volatility-tiered review cadences, ownership rules, audit log	Core
6. Measurement Workbook	Cost-per-answer economics, shift-left ratio, scenario modeler, metric-inversion brief, KPI catalog, exec dashboard	Delivery
7. Rollout Playbook	The 90-day implementation plan, pilot success-criteria contract, exec pitch kit, failure-mode catalog	All
8. Terminology Governance Pack	The vocabulary layer: a 40-entry term inventory, a ratifiable terminology standard, gate language line-items, the 30-day audit, and the champion kit	Core + Curation

Everyone starts at Module 1. Placement comes before prescription — the routes below assume you've scored yourself and know your weakest dimension. Then, by level:

- **Level 0–1** → Module 2, then 5, then 4. Type your knowledge and install governance before touching AI. Resist the bot.
- **Level 2** (KCS-certified, feeling obsolete) → Modules 3 and 4 together — replace authoring with drafting + gating; your existing standards map onto Module 2 in an afternoon. Then Module 6 to reframe your measurement before leadership asks.
- **Level 3** → Modules 5 and 6 — deepen governance, and move your financial story from activity metrics to answer economics.
- **Level 4** → Module 6 to optimize the loop — then re-run Module 1's assessment to plan the Level-5 climb.
- **Rolling out org-wide at any level** → Module 7, alongside whatever the routing above says.
- **Client-facing content at any level** → Module 8 — vocabulary is the one dimension where drift is public; its 30-day audit is a self-contained first program even where nothing else is installed yet.

Every module also stands alone. None hard-depends on another; each states its assumptions ("articles typed per Module 2 — any typed template works") so you can adopt piecemeal.

8. What an implementation actually looks like

Ninety days, one queue or product domain, at rung 1 — no enterprise platform:

- **Weeks 1–2 — Baseline.** Maturity placement (Module 1), channel inventory and unit costs (Module 6), corpus audit, pilot success criteria signed (Module 7).
- **Weeks 3–6 — Pilot.** Templates live (Module 2), drafting prompts wired to your ticketing exports (Module 3), gate running with 2–3 trained curators (Module 4), lifecycle rules active (Module 5).
- **Weeks 7–12 — Prove and expand.** Governed articles compounding, blended cost-per-answer baseline on a dashboard, gap reports firing capture requests, exec readout delivered (Modules 6 + 7).

Design targets throughout: agent effort per capture ≤ 30 seconds; curator effort per clean approve ≤ 3 minutes; capture-to-published ≤ 1 business day.

---

9. Glossary

Twenty-two terms, one line each — so any single module stands alone without a decoder ring. Terms marked with a module number are defined in full there.

- **Governed Knowledge Loop** — the three-loop operating model (Capture → Curation → Delivery) around a shared system of record; §2.
- **Candidate** — AI- or human-proposed content waiting at a gate; never knowledge until dispositioned (Modules 2–4).
- **Curation gate** — the human decision point every write passes; three kinds: new-candidate, update-DIFF, retirement (Module 4).
- **Disposition / reason code** — the gate's single decision (A1 approve · A2 edit-approve · M1 merge · R1–R6 reject) plus the code that makes rejects tunable telemetry (Module 4).
- **Edit-distance** — how much a curator changed an AI draft before approving; trended, it is your drafting-quality benchmark (Module 4).
- **Capture tax** — the effort a process demands from the person solving the problem; above ~30 seconds, capture dies (§3, rule 2).
- **Precision Paradox** — the more polished AI answers become, the more each remaining error costs (§1).
- **Boring core, smart edge** — deterministic governed record at the center; probabilistic AI only at the edges (§3, rule 1).
- **Machine-ready** — content passing all four machine-legibility properties: self-contained sections, explicit applicability metadata, stable identifiers, freshness signals (Module 2).
- **System of record** — the one designated, versioned, typed home for validated knowledge (§2, Module 5).
- **Validated** — the only lifecycle state allowed to ground answers or display as trusted (Module 5).
- **Volatility** — how fast an article's truth decays (evergreen / medium / high); drives review cadence (Modules 2, 5).

- **Blended cost per answer** — total answer spend ÷ total answers, across every delivery channel; the program's headline number (§4, Module 6).
- **Shift-left ratio** — share of answers delivered left of assisted support on the channel spectrum (§4, Module 6).
- **Known-vs-new mix** — of assisted cases, how many are already-answered issues; known issues leaking back = findability failure (Module 6).
- **Metric inversion** — assisted-queue metrics worsening *by design* as shift-left succeeds; the easy volume left (§4, Module 6).
- **Issue-centric measurement** — counting the issue answered N times across channels, not the single 1:1 case (§4).
- **Structure gradient / rung 1** — the climb from typed Markdown to semantic knowledge; rung 1 is genuinely sufficient to start (§5).
- **Decision log** — the append-only record of every gate disposition, lifecycle transition, and override (Module 5).
- **Living module** — the one deliberately model-dependent module (Module 3, prompts), version-stamped and updated free as the model landscape shifts.
- **Terminology governance** — vocabulary as a governed property: canonical terms, deprecated terms, and a ratified standard enforced at the gate (Module 8).
- **Canonical term** — the one designated name for a concept; synonyms map to it, never compete with it — it is the retrieval key (Module 8).

10. Provenance

Contextkeeping is a **KCS-aligned** operating model for the AI era. KCS® is a service mark of the Consortium for Service Innovation; this toolkit is independent work, not affiliated with or endorsed by the Consortium. Architectural lineage — including the governed system-of-record pattern and edit-distance-as-quality-benchmark — is credited in ATTRIBUTION.md, as is the shift-left value model's public lineage. All dollar figures anywhere in this toolkit are illustrative placeholders; replace them with your own.

Version 1.0 · License: single organization, internal use — see LICENSE.md