>_ **PrimeLine** Build Systems, Not Sessions

A FIELD GUIDE FOR THE CURIOUS

The Anatomy of a Thinking System

How Evolving and Kairn actually work - a multi-agent operating system for Claude Code, read as a nervous system: reflexes, instincts, memory, and the wiring that keeps it alive between one thought and the next.

System Evolving + Kairn Author Robin Lorenz >_ primeline.cc

00 PROLOGUE

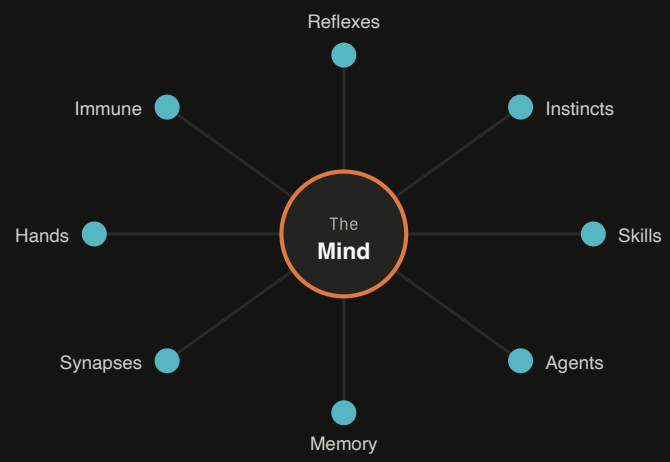
A system that does not forget

Most AI tools are sessions. You open one, it is brilliant for an hour, and then it forgets you ever met. I got tired of re-explaining myself, so I stopped building sessions and started building a system: one that wakes up already knowing what it learned yesterday, corrects its own mistakes, and gets a little sharper each time it runs.

That system is **Evolving**, and its memory is **Kairn**. Under the hood it is hundreds of small parts: commands you can call, reflexes that fire on their own, instincts it never questions, specialists it delegates to, and a web of connections holding it all together. Written out as a spreadsheet, that is a numbing pile of files. But it does not behave like a spreadsheet. It behaves like an organism. So that is how this guide reads it.

Every part maps to a piece of anatomy, not as decoration, but because the shapes genuinely rhyme: some parts think, some react without thinking, some remember, and one quiet layer keeps the whole thing from drifting apart. Here is the map you are about to walk through.

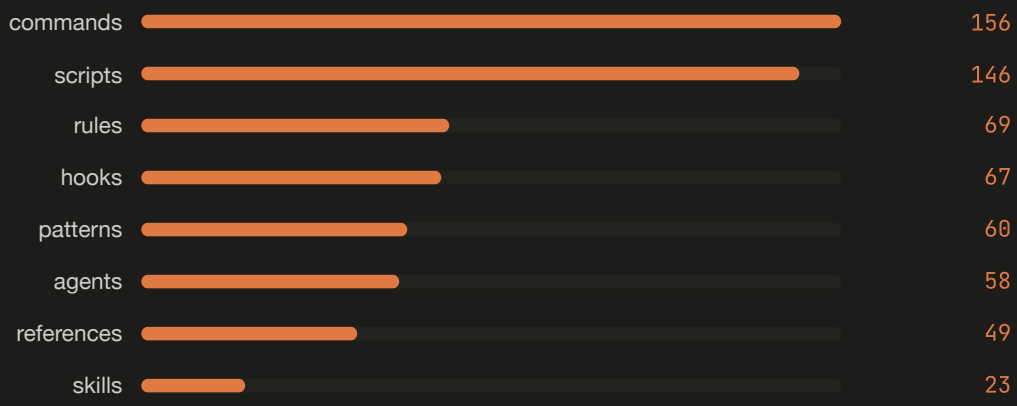
Prefrontal cortex	**Orchestration** decides and delegates	Motor cortex	**Commands** deliberate action
Autonomic reflexes	**Hooks** fire on their own	Instincts	**Rules** never questioned
Cerebellum	**Skills** learned procedures	Cortical regions	**Agents** specialists
Hippocampus and cortex	**Kairn** long-term memory	Synapses	**Knowledge graph** the wiring
Immune system	**Self-healing** fights drift	Neuroplasticity	**AutoEvolve** self-tuning



The whole system, one glance: a deciding core wired to eight standing subsystems. Nothing here is loaded all at once - the wiring is how they find each other on demand.

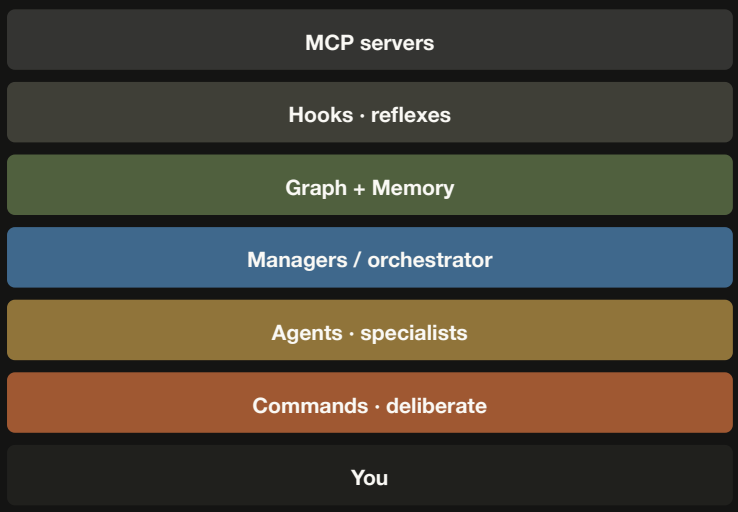
The body, by the numbers

Live parts by type - the eight capability layers



619 working parts across eight types. Historical artifacts - plans, handoffs, ledgers, decisions - are deliberately excluded; this is the living body, not its diary.

The same system, stacked



From your message at the top, down through the machinery, to the MCP layer at the base - the network map on the previous page seen as a stack.

Part I

The Mind

The Orchestrator

> The prefrontal cortex. For every incoming task it decides, in a fraction of a second, whether to do the work itself or hand it to a specialized region, which expert to recruit, how much effort to spend, and what temperament to bring. This is top-down control, not reflex.

560	19	~6%	3.0
COMPOSABLE AGENT PROFILES	TASK TYPES ROUTED	OF COST IS THE PROMPT	BASE DELEGATION THRESHOLD

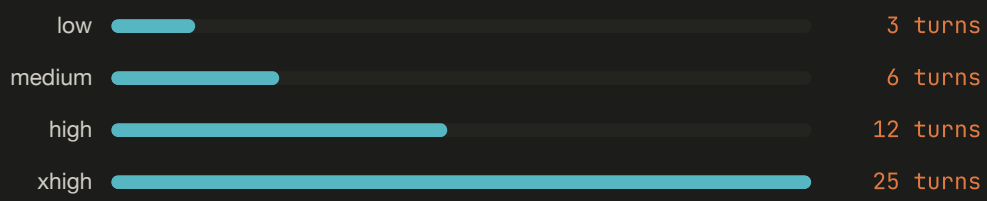
Before Claude acts on anything you type, a scoring engine reads the request and asks one question: should I really do this myself? It weighs signals - is this exploration, how many files, is it production-critical - into a number, and compares that number against a threshold that moves on its own.

If the score clears the bar, the system picks a specialist, a model tier sized to the difficulty, an effort budget that caps how many moves the specialist gets, and even a personality to shape how it works. Then it briefs that specialist on what else is running right now, and lets several specialists quietly trade discoveries while they work, so two of them never investigate the same thing twice.

WHY A ROUTER BEATS A BIGGER BRAIN

Keeping a top-tier model in the orchestrator seat is expensive, because that seat fires on **every single turn**, and re-reading the conversation dominates cost far more than the wording of any one prompt. Measured over 60 days and 326 real sessions: the fresh prompt is only about 6% of what a session costs, while ingesting and re-reading context runs 74 to 93%. So the deep thesis here is that **a routing system is a substitute for an expensive model**. Most setups compensate for having no delegation layer by putting a powerful model in charge of everything. This one built the routing layer instead, then deliberately ran the orchestrator on a cheaper model. An A/B test found no quality loss once the router was in place.

Effort levels are also turn budgets (how many moves a specialist gets)



Effort and turn-count are separate levers, tuned per task type. Security review, for example, was pinned to high effort but kept a 25-turn budget after a test showed high matched the most thorough setting at roughly 0.6x the cost.

The threshold itself is described, in the system's own design notes, with a physics metaphor: a barrier width that narrows when conditions favor delegating (fresh context, a good recent track record) and widens under crowding (two or more specialists already busy). That analogy went almost verbatim from a brainstorm into the formula that runs on every turn.

INSIDE THE EXECUTIVE	
`Delegation Enforcer`	Scores every prompt, picks agent, model, effort and personality, injects the decision before the main agent acts
`Adaptive Threshold`	Recomputes the delegate-or-not bar each turn from context freshness, recent success, and how many agents are already active
`Config Matrix`	One file maps ~19 task types to an agent, model tier, effort and personality together
`Coordination Block`	Reads a live pulse of who is doing what, so the orchestrator is never flying blind
`Self-Tuning Layer`	Gated, observe-first mechanism that lets the routing config improve itself from measured outcomes, with locked fields it may never touch

02 VOLUNTARY MOTION

The Commands

> Voluntary motor commands from the motor cortex. A deliberate, named signal - lift the arm - that fires a pre-wired sequence on demand. The counterpart to the reflexes, which fire without being asked.

~164	~20	196	47
COMMANDS REACHABLE	FUNCTIONAL FAMILIES	NATURAL-LANGUAGE TRIGGERS	SHARED BY SYMLINK

A command is a small text file: a short header, then a script of instructions Claude follows when you type its name. It is the deliberate layer. A command only ever runs because a person named it, which makes it the exact opposite of the ambient reflexes in the next chapter.

Roughly 164 are reachable here: about 117 defined directly, plus 47 shared in from packages through directory links, organized into around twenty loose families - thinking and brainstorming, audits, media generation, research, memory, content. Each one pins the model tier it should run on, so a cheap lookup stays cheap and a hard reasoning job gets the stronger model automatically. Many of them, when run, delegate their real work to a squad of specialists from the previous chapter.

ASK BEFORE YOU LEAP

Well-built commands follow a named pattern called the Intake Gate: check whether the input is actually sufficient, ask one clarifying question if it is vague, confirm, then act - rather than plunging into execution on thin input. Commands are also built to chain: a command's closing line often suggests the next one to run, so the surface behaves less like a flat menu and more like building blocks you string into a workflow.

The count itself has a story. The system's own inventory of what commands exist had quietly drifted out of sync with the files on disk - whole categories missing, ghost entries for commands long deleted - until a recount that reads straight from disk replaced the old habit of trusting a hand-kept index. It is a small preview of a theme that runs through this whole system: what exists and what the system believes exists are two different things, and keeping them equal takes constant work.

Part II

The Reflexes and the Instincts

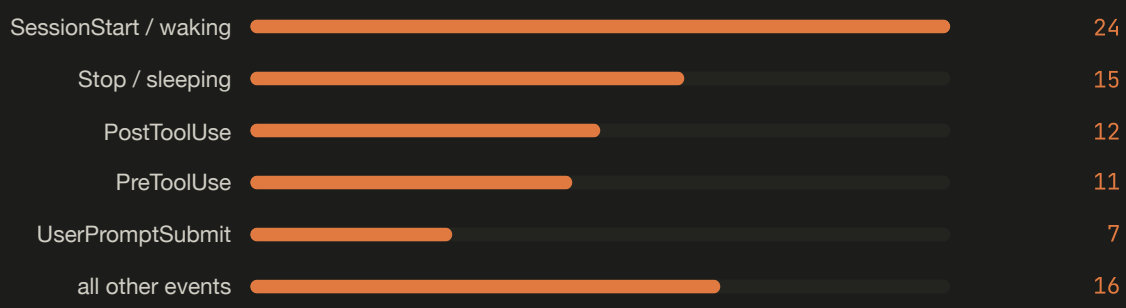
The Reflexes

> Reflex arcs that fire the instant something happens - a tool runs, a message arrives, a turn ends - with no decision from the conscious mind. Like pulling your hand off a hot stove before you have thought the word hot, most of these run and finish before the model itself is even involved.

85	83	24	122
LIVE REFLEX BINDINGS	DISTINCT SCRIPTS	FIRE ON WAKING	SCRIPTS ON DISK

Every action inside a session raises a named event: a message submitted, a tool about to run, a tool just finished, a whole turn ending. Small standalone scripts are wired to those events. They run outside the model's reasoning - each reads a little data, does one narrow job, and exits. They are the body reacting before the mind weighs in.

Where the reflexes cluster (event bindings by moment)



Waking up is the busiest moment: 24 reflexes fire before you type a word. Falling asleep is second. In between, one fires just before and just after every single tool the system touches.

40+ reflex firings in a single turn - nearly all invisible unless one breaks

A single turn passes through all of these, roughly 40 firings before and after the model's own reasoning - nearly all of it invisible unless something breaks.

Most reflexes are built to fail open: if one misfires, the body does not seize up, it fails to react that once and moves on. Every reflex is even wrapped in a check that asks does this script still exist first, so a

moved or deleted file degrades to a harmless no-op instead of crashing the turn.

THE DARK REFLEX - WHY THIS LAYER IS BUILT THE WAY IT IS

Fail-open is not free. A reflex that swallows its own error and exits quietly looks **identical**, from the outside, to one that ran perfectly and had nothing to report. The system learned this the hard way: the gate meant to catch a confident done with no evidence behind it was reading an event field that real turns never fill in. It passed everything, silently, for an unknown stretch of real sessions, while looking healthy and passing every synthetic test. The fix reshaped the rule for every reflex since - pair every silent failure with a health signal, so ran and found nothing can always be told apart from broke without telling anyone.

>_ ***A reflex that fails silently is indistinguishable from one that simply had nothing to do. The most dangerous bug is the invisible one.***

The reflexes run in tiers of consequence. Most are pure observers that can only warn - a scanner reading fetched web content for hidden injected instructions never blocks, only flags. A deliberate few are structural guards that can refuse an action outright: one blocks writes to protected files and irreversible commands, but only while the system runs unattended on its own; another grades every shell command across ten tiers of risk. Everything else is a gentle nudge in between.

The Rules

> Standing reflex arcs plus a much larger library of learned patterns. A handful of instincts fire every cycle without being decided; a far bigger catalogue stays dormant until a specific situation recruits it.

15	64	23.5K	~3.6K
ALWAYS-ON INSTINCTS	ON-DEMAND RULES	TOKENS, BEFORE	TOKENS, AFTER

Some rules load into every single session automatically - the always-on instincts covering how to delegate, how to reason before acting, how to prove a task is actually finished, how to classify a request, and when to consult memory. A much larger body of rules stays parked and loads only when a keyword or situation calls for it, keeping the constant background cost small.

THE COST OF AN INSTINCT

A rule only works if it is reliably present at reasoning time - a rule you have to remember to apply is a rule that gets skipped under load. But every constantly-loaded token is paid on every tool call for the rest of the session. Early on, all rules lived in the always-loaded folder, costing **23,500 tokens per session**. Moving 51 of them to on-demand loading brought that down to roughly 3,600, with no information lost. The instinct set is deliberately kept small and sharp; the library behind it can be enormous.

The always-on tier is split across two locations: a slim copy that applies to every project on the machine, and a fuller copy for this project alone. They are written as non-identical siblings on purpose. An earlier version used a shortcut that pointed both at the same underlying file, which quietly loaded every rule twice into the same session. The fix accepts a little duplication so each tier only ever contributes once. Five of the seven project rules name their global sibling explicitly; two - covering component registration and confidentiality - are project-only, with no global counterpart at all.

A companion mechanism fights instinct fade: a reflex re-injects short rule reminders as the conversation grows, because content loaded once at the very start measurably loses its grip past roughly 200,000 tokens. The rules alone are not trusted to hold attention over a long session; something has to keep whispering them.

THE FIVE ALWAYS-ON INSTINCTS	
`delegation`	When to hand work to a specialist versus do it directly, and on which model
`quick-dsv`	Decompose, suspend, validate - a three-question reasoning check before acting on anything open-ended
`orchestrator`	Classifies every request and routes it before any other rule engages
`mid-session-kairn`	Makes consulting long-term memory the default posture, not a fallback
`empirical-completion`	Forbids treating a file written or a command run as proof the goal was met - evidence of the effect is required

05 LEARNED PROCEDURES

The Skills

> A learned motor program stored in the cerebellum. Not reasoned out fresh each time, but a packaged routine that fires the instant its trigger is recognized - a phrase like map poster or harden this prompt - then runs largely on autopilot, sometimes in an isolated reflex arc of its own.

23	33%	2	1
SKILL FOLDERS	TRACKED BY THE SPINE	EXECUTION MODES	IMPORTED FROM OUTSIDE

A skill is a folder with one instruction file: a header naming its trigger words, then a markdown playbook. When your phrasing matches, the playbook loads into context and gets followed - a reusable expert routine rather than code. Some run inline in the conversation; heavier ones declare themselves forked, spinning up an isolated helper with its own tool budget and timeout, then reporting a result back.

Skills exist so that non-trivial procedures - writing a decision record, generating a city-map poster, triaging a backlog, hardening a prompt - do not have to be re-derived every time. They are captured once and triggered by pattern-matching your own words, discovered through the same registration machinery as commands. One skill even scaffolds new agents, commands, hooks and skills from templates. The mix is not purely home-grown: at least one skill is openly imported from a third-party repository, living under the same discovery mechanism as the rest.

A BLIND SPOT THE SYSTEM CAUGHT ABOUT ITSELF

A validation sprint found that **16 of the 23 real skill folders on disk are entirely invisible to the system's own tracking index** - full coverage measured at just 33%. The skills clearly exist and fire; the index meant to know what skills exist only sees a third of them. This is treated as a real gap in the registration machinery, not a property of the skills, and it is exactly the kind of drift the immune system later in this guide is built to catch. A related idea - a loop that watches for repeated task patterns and writes its own new skills automatically - was seriously weighed and then deliberately parked until a concrete recurring case shows up to justify the build.

06 SPECIALIZED FACULTIES

The Agents

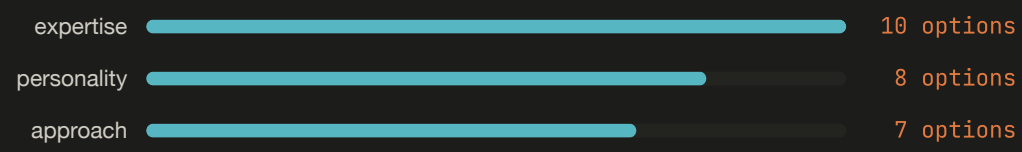
> Cortical specialization: distinct regions dedicated to one job each, rather than one generic tissue doing everything. The named agents are fixed, myelinated pathways for recurring jobs. The Agent Factory is the brain's ability to wire up a bespoke coalition on the fly for a job no dedicated region already covers.

58	560	13	3
NAMED SPECIALISTS	COMPOSABLE PROFILES	DEDICATED AUDITORS	GUARDED DOMAINS

Evolving does not run one do-everything AI. It keeps a directory of 58 small, single-purpose agents - each a plain file with a role, a strict tool allowlist, and a model tier - that the main session calls out to: auditing code, fixing what an auditor found, generating a whole new sub-system from a spec, running multi-source research.

On top of the fixed specialists sits a second layer: the Agent Factory. Instead of writing a new file for every new kind of task, it composes one on the fly by picking three traits - an expertise, a personality, and an approach - and filling a template with the matching voice, tools and instructions. Ten expertises times eight personalities times seven approaches is 560 possible on-demand profiles. Enumerating 560 files would be impractical; composing them from three orthogonal dimensions is cheap.

The Agent Factory: 560 profiles from three dimensions



10 x 8 x 7 = 560. The 7th approach, remediation, was added specifically to give the audit-fixer family its own composable identity, bumping an earlier 480 to 560. Legal, security and medical expertise trigger a disclaimer block injected at compose time - the factory treats some domains as needing guardrails beyond just role and voice.

SPECIALISTS ARE TRUSTED FOR THE JOB, NOT FOR GRADING THEMSELVES

Agents that write code in isolation are explicitly **not** treated as pre-reviewed. A mandatory review pass runs on every agent-authored change before merge, because a retroactive audit found real defects that the agent's own green test suite had missed. The stance is consistent throughout: a specialist is trusted for the narrow job it is scoped to, never for self-certifying the correctness of its own output. The auditor and fixer families are even built in pairs - each checker produces findings in a format its matching fixer is written to consume directly.

Part III

Memory

07 LONG-TERM MEMORY

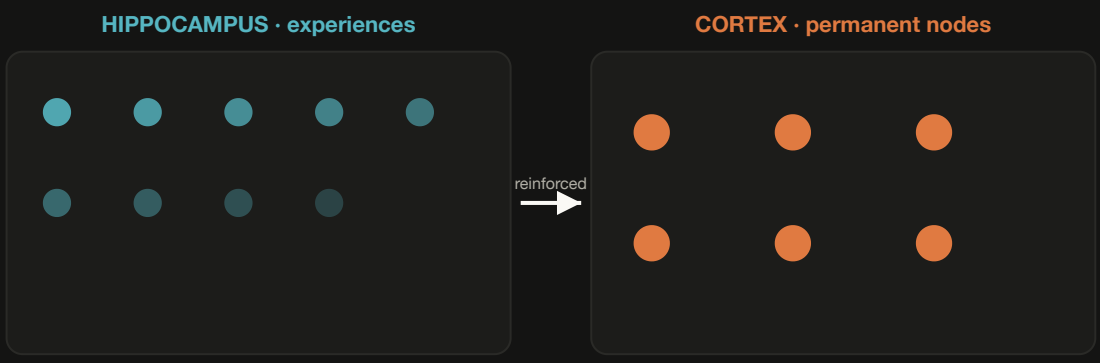
Kairn

> The hippocampus-to-cortex consolidation pipeline. Fresh, uncertain experiences enter as short-term traces that fade on a decay curve unless repeated recall reinforces them - at which point they consolidate into permanent, non-decaying knowledge, exactly like hippocampal traces being promoted into durable cortical memory.

9,603	7,950	12,933	~1.4ms
PERMANENT NODES	DECAYING EXPERIENCES	EDGES	TYPICAL RECALL

Every fresh session otherwise starts from zero: no memory of prior decisions, gotchas, or preferences. Kairn (its open-source name is engram) exists so insight persists across sessions instead of being re-derived every time. It is a knowledge-graph memory server any tool can talk to over 22 commands, and the system's own rules make consulting it before acting, and saving immediately after deciding, the default rather than a fallback.

It stores two kinds of knowledge in one local database. Permanent Nodes are facts, decisions and patterns that never decay. Experiences are the same shape of content but time-stamped and exponentially decaying unless reinforced. Recall is keyword-based by default - zero dependencies, sub-two-millisecond, nothing ever leaves the machine - with an optional local embedding rerank a workspace can switch on for harder semantic queries.



fresh traces fade on a decay curve; repeated recall consolidates them into durable knowledge

Fresh experiences enter as decaying traces; repeated recall consolidates the important ones into permanent, non-decaying nodes - exactly the hippocampus-to-cortex handoff.

Experiences forget at different speeds (decay half-life by type)

These half-lives are not guessed. They were derived by measuring the real re-access tail of a 6,483-experience production store; the original guessed values were 3 to 21 times too long, which left decay effectively inert. A preference should persist near-permanently; a workaround is expected to go stale fast.

FOR MOST OF ITS LIFE, THIS MEMORY FAKED HOW WELL IT REMEMBERED

For most of Kairn's history every permanent node reported the exact same relevance - a perfect 1.0 - no matter how well or poorly it matched a query. The database computed a genuine ranking, but it was silently dropped three function calls up the stack before it reached the answer. That made the abstention safeguard impossible: an alien query like protein folding molecular dynamics against an unrelated knowledge base would still return five nodes, all reported as perfectly relevant. It was found in July 2026 by reasoning backward from a failing benchmark, not by reading the code. On a published memory benchmark Kairn now scores 56.2% across all 500 questions.

>_ ***Recall is only worth trusting if the system can also say: I do not know. For most of this memory's life, it structurally could not.***

One discipline runs through the whole design: honesty over cleverness. The decision to add optional embeddings was forced by evidence, not assumed - keyword recall was measured hitting a hard ceiling of 64 to 66% on an adversarial test set written by two independent non-Claude models specifically so the team could not over-fit to its own benchmark. And a separate bookkeeping of when a fact was true in the world, versus when it was learned, is kept deliberately orthogonal to decay, so the two never contaminate each other.

08 WORKING MEMORY

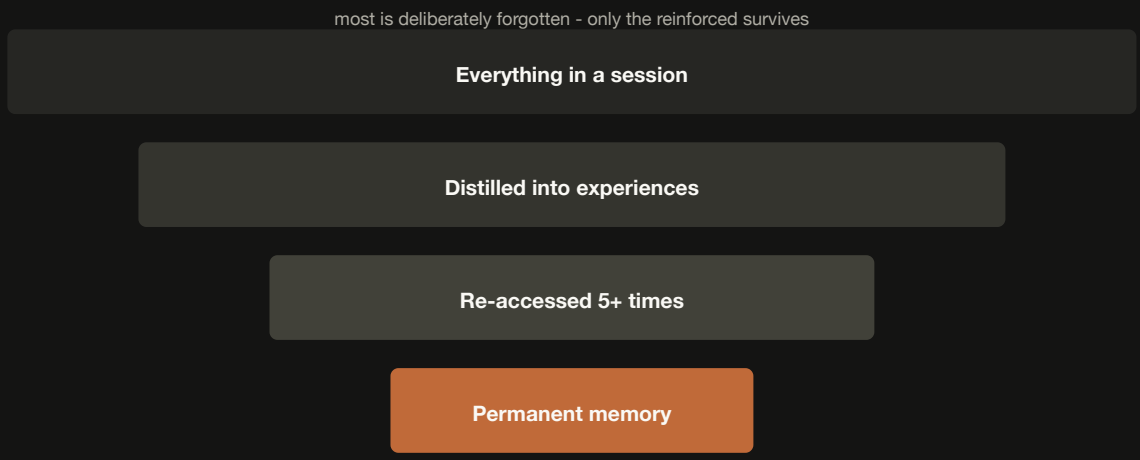
Consolidation

> Hippocampal consolidation. Short-term working memory - the current session - gets triaged: some decays into forgetting, some is rehearsed into stronger recall, and a small fraction is promoted into durable long-term memory that survives any single episode. The bootup ritual is the waking-and-orienting moment before acting.

4	8	~11	5+
MEMORY LAYERS	EXPERIENCE TYPES	BOOTUP SUB-STEPS	ACCESSES TO PROMOTE

Kairn is the durable store; this is the layer that feeds it. A four-layer file-based memory lets a stateless session behave as if it remembers: a session index (the active project and last three sessions), per-project state (goals, progress, recorded failures), the volatile experiences layer, and pure usage analytics. Every session opens with a mandatory bootup ritual that reads this state, and closes by writing progress back plus a human-readable handoff for the next session.

A background pass at session end mines the session's own tool-use journal for repeated patterns and writes new experiences automatically, gated by a noise filter so most of what happens is deliberately forgotten - not every edit loop deserves to become a permanent memory. Frequently re-accessed experiences get promoted into permanent Kairn nodes; the rest decay and are eventually archived.



Most of a session is deliberately forgotten; only the reinforced fraction is promoted into permanent memory.

A CONSOLIDATION LOOP THAT LOOKED WIRED BUT DID NOTHING

For a period, one promotion loop was silently a no-op end to end. A database trigger correctly flagged experiences accessed five or more times as ready to become permanent knowledge - but nothing read that flag. It just accumulated as a dead-letter queue, looking perfectly wired, until a dedicated promotion sweeper was finally built to act on it. It is the same lesson the reflexes taught, one layer deeper: a pipeline that looks connected and produces no error is not the same as a pipeline that works.

This is why the system's highest-priority rule is about honesty of memory: a system that only claims to remember but silently forgets is worse than one that admits it has none. Almost everything in this guide, at some point, was caught quietly not doing what it appeared to do - and the machinery of the next two chapters exists precisely to keep catching it.

Part IV

The Wiring and the Body

09 THE SYNAPSES

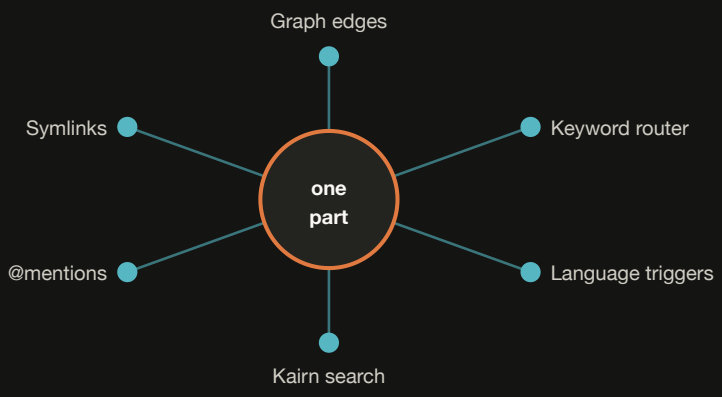
The Knowledge Graph

> The brain's synaptic network and Hebbian plasticity - neurons that fire together wire together. Nodes are neurons, the edge file is the fixed wiring diagram, and the co-activation file is activity-dependent strengthening that decays without reinforcement, on a literal sixty-day half-life.

1,659	2,233	531	6
KNOWLEDGE NODES	TYPED EDGES	KEYWORD ROUTES	DISCOVERY PATHWAYS

Loading the full contents of 1,500-plus components on every turn would blow the context budget, so the system needs cheap, on-demand discovery instead. A set of files lets every agent, skill, command, hook, rule, template and pattern find each other and be found - without ever loading the whole body into memory at once.

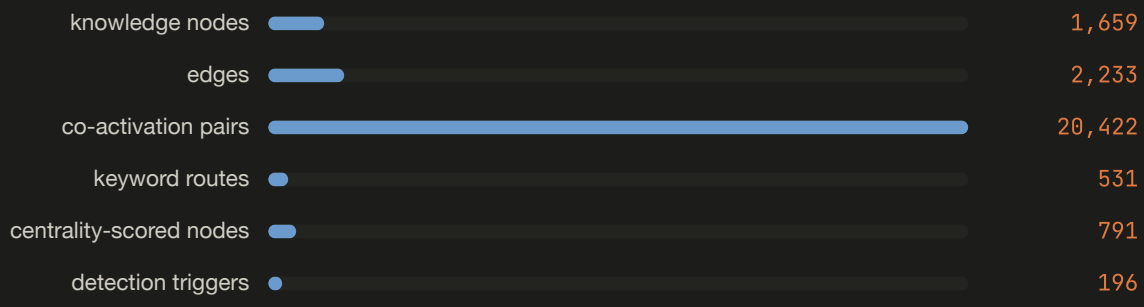
There is no single master index. Six independent pathways cross-reference everything: an explicit relationship graph, a keyword router, a natural-language trigger index, the semantic memory layer, in-file mentions, and filesystem links. A component that looks orphaned by one pathway may be very much alive through another. This doctrine was not designed up front - the document that codifies it was written after deep research revealed that an audit checking only the edge graph would have recommended deleting components that were load-bearing through a different pathway entirely.



six independent ways to be found - invisible on one, load-bearing on another

A component can be invisible on one pathway and fully load-bearing on another - which is why an audit that checks only one would delete things that are still in use.

The graph, by the numbers



20,422 co-activation pairs were learned from 1,420 real usage events, decaying on a 60-day half-life. A health scan of 1,506 components found 706 fully registered, 800 partially, and zero fabricated ghost entries - overall health 0.81.

A FROZEN ATTENTION MECHANISM

In its own designer's words, the router is a frozen attention mechanism: its keyword lists behave like fixed query-key matrices, but the usage signal that should act as a learning gradient mostly does not loop back into it. Only a narrow top-50 slice of that signal reaches any live consumer at all. And there is a deliberate counter-rule: strong co-activation between two components is explicitly not enough to justify adding a real edge, because many pairs fire together by coincidence rather than because data flows between them. Edges mean dependency, not mere company.

Scripts, References, Patterns

> The spinal cord and its reflex-arc library: not the thinking cortex, but the trained motor programs and standing instructions that let the body execute complex maneuvers without re-deriving them every time. Scripts are the reflex arcs; references and patterns are the muscle-memory library of how we do X and shape of solution for Y.

~146	~38	60	2
SCRIPTS	REFERENCES	PATTERNS	SPLIT BY DESIGN

This is the toolbox and the cookbook. Roughly 146 standalone scripts do the mechanical work - scanning, repairing, aggregating, dashboarding, syncing - that would otherwise be done by hand or re-reasoned by an agent every time. Alongside them sit two curated libraries: about 38 reference guides for specific tools, and 60 named patterns - reusable solution shapes for recurring problems, from how independent AI helpers coordinate without knowing about each other to how you let a system safely rewrite its own rules.

The scripts exist because the system self-modifies constantly: new automations, agents and rules get added by the AI itself, not just by hand, and without a mechanical check on that growth, components silently fall out of sync. Diagnosis and repair are deliberately kept as two separate scripts - one diagnoses, one repairs - and repairs are tiered by risk: safe fixes apply automatically, cautious ones ask first, dangerous ones require explicit approval. The libraries exist for the mirror reason: so new work reaches for a named, battle-tested shape instead of reinventing a solution style every session.

WHO CHECKS THE CHECKER

The health scanner that audits the rest of the system for silently-broken wiring was itself found to be silently broken. It dropped an entire category of real components as ghosts, and carried fabricated test entries with no backing file. The auditor needed its own audit before it could be trusted - a very literal case of who checks the checker, and the reason later work treats even the drift-detector as a spine that itself needs periodic validation, not an infallible source of truth.

Part V

Staying Alive

11 THE IMMUNE SYSTEM

Self-Healing

> The body's homeostatic and immune machinery: constant background monitoring that notices when a new part was added but never wired into circulation, flags it, and often repairs it without the brain consciously intervening. Not one organ, but a reflex arc distributed across every tissue boundary.

7	4	~1,900	80%
REGISTRATION POINTS	CHECKPOINTS	LINES IN THE SCANNER	HEALTH TO PASS

The system's coherence depends on a handful of files all agreeing about what capabilities exist and how they connect. This subsystem keeps those files honest. It scans for components added but never fully wired in, warns before an edit touches something already under-registered, silently registers brand-new files the moment they are written, and runs a full health-scored sweep with safe auto-repair at the end of every session. It is the reason a newly created capability does not quietly become invisible to the rest of the system.

Each piece targets a different moment drift can creep in: at session start (stale cross-references), at the moment a file is written (a missed registration, closed within about twenty milliseconds), at the moment a file is edited (surfacing its registration health so the change is not made blind), and at session end (the full scored sweep). Every one of them is deliberately fail-open - a broken safety check must never become a bigger problem than the drift it was meant to catch.

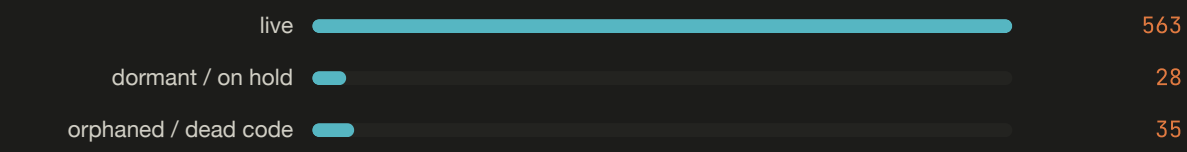
THE FOUR-SECOND MERGE

One guard exists because of a very concrete near-miss. A private repository cannot use the platform's server-side required-checks feature, so a plain merge could complete before its automated checks even finished. A real change was once fully merged into the main line **four seconds** after its checks started running - long before any could have passed - and a red, failing state sat on the trunk unnoticed for about a week. A merge wrapper now refuses to let a change land until every check has actually finished and passed.

Now the body can see itself

The newest organ in this system is a single index it keeps of itself. Instead of searching the tree to answer does this exist, where does it live, is it safe to change, one generated file now answers all of it for every one of its ~625 parts at once - and for each, whether it is live, dormant, or orphaned, plus how many other parts depend on it. It is the same discovery layer this guide's own research leaned on.

The system, seeing itself: every part's activation state



Of ~625 components, most are live. 28 are deliberately parked, and 35 are orphaned - wired to nothing, consumed by nothing. Being able to see that at a glance is how the body prunes itself without a human reading every file.

MAKING DRIFT VISIBLE IS THE WHOLE JOB

A part that nothing depends on is either a mistake or a candidate for deletion, and until recently the only way to find one was to read the tree by hand. The self-index does not fix drift - it makes drift **visible**, which is the prerequisite for everything else the immune system does. You cannot prune what you cannot see, and a system that adds parts faster than a human can track them has to be able to see itself, or it slowly fills with dead weight nobody remembers building.

12 NEUROPLASTICITY

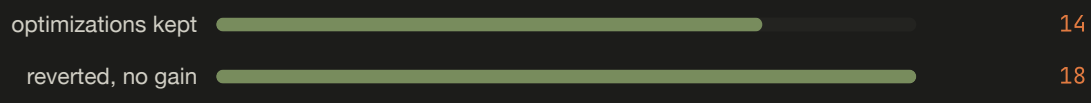
AutoEvolve

> Neuroplasticity and interoception. A mutate-score-keep-or-revert loop is synaptic pruning - fire together and wire together, or get reverted. The fitness ledgers are proprioceptive feedback on how well each muscle performed. And a health monitor with a habituation reflex is like an amygdala that stops screaming about a threat once it has rung the same alarm enough times without new evidence.

~1,497	14	18	0.653
LOGGED OUTCOMES	KEPT OPTIMIZATIONS	REVERTED	CURRENT SELF-HEALTH

A cluster of small deterministic scripts lets the system observe its own behavior and adjust without a human hand-editing config. One arm proposes small edits to specific tunable files, scores each edit against fixed tests, and keeps only edits that measurably improve - working on an isolated branch with cooldowns and a human-review gate. A second keeps rolling scorecards of which trait, personality or delegation choice actually worked. A third watches which pieces of knowledge keep activating together and turns recurring clusters into new named patterns.

Self-tuning keeps only what measurably improves



Every proposed edit is scored against fixed tests; only improvements survive, the rest are reverted and archived. 14 kept against 18 reverted - more rejected than accepted, which is exactly what a working filter looks like.

DELIBERATELY KEPT AWAY FROM AN AI JUDGE

The self-improving loop is deliberately kept away from letting an AI grade its own work on anything safety-adjacent. A design review flagged that an AI judge rewards confident-sounding output over calibrated uncertainty, and that pairing a self-improving optimizer with an AI judge on a high-stakes target could institutionalize false confidence and erode human oversight. So scoring is mostly deterministic, certain zones are structurally frozen from ever being touched, some fields are permanently locked as human-only, and a human merge gate stands at the end. Even an unlocked field can only be tuned after its measured fitness has stayed poor for several consecutive runs - a guard against fidgeting with settings that already work.

The honesty theme peaks here. The system's own self-health score was once pinned at zero for months by 31 drift alerts that had already been fully recognized as stale and decayed toward a floor by a separate mechanism. Every actual performance metric underneath was a healthy 1.0 - but the health formula was counting alarms instead of weighing how loud each one currently was. The system kept reporting itself as unhealthy while functioning perfectly, until the formula was taught to use the habituation data that already existed.

Part VI

The Evidence

The Evidence

> Proprioception - the sense that tells you where your own body is without looking. A system that cannot measure itself cannot improve. This one is instrumented to prove, in numbers, that the machinery earns its keep rather than just looking clever.

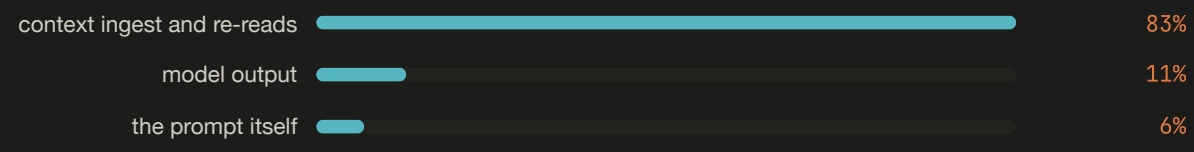
A nervous-system story is only worth telling if the body underneath performs. So here is the part that matters most to me: the measurements. Not vibes, not it-feels-faster - benchmarks, A/B tests and cost studies, several of them run against test sets I deliberately did not write myself, so I could not quietly grade my own homework.

56.2%	~1.4ms	0.6x	~6%
memory recall	**recall latency**	**security-review cost**	**of cost is the prompt**
LongMemEval-S, all 500 questions scored	typical keyword lookup, zero network	same thoroughness, high vs xhigh effort	the other 94% is context, over 326 sessions

Why a cheaper brain, wired well, beats an expensive one

The most expensive habit in AI tooling is keeping a top-tier model in charge of everything, because that seat fires on every single turn. So I measured where the money actually goes across 60 days and 326 real sessions. The wording of any one prompt turned out to be about 6% of the bill. Re-reading the conversation - the context tax - is the overwhelming rest.

Where a session's cost actually goes



Measured over 60 days and 326 sessions. The lesson is blunt: tuning prompt wording optimizes 6% of the bill. Routing work to a cheaper model and keeping context lean optimizes the other 94% - which is exactly why the orchestrator runs on a mid-tier model, and an A/B test found no quality loss once the routing layer was carrying the weight.

THE BENCHMARKS I TRUST ARE THE ONES I COULD NOT RIG

When keyword recall was measured against a set of adversarial queries, that set was deliberately authored by **two independent non-Claude models**, so the system could not over-fit to a test it wrote itself. Keyword-only recall capped at 64 to 66%, and the specific failures that survived - a word with two meanings, a synonym the index missed - were exactly what justified adding optional local embeddings. A self-graded system flatters itself; this one is checked by outsiders on purpose. The decay half-lives got the same treatment: they were derived from the real re-access pattern of a 6,483-experience store, not guessed. The first guesses were 3 to 21 times too long.

The system that audits itself

There is a second kind of evidence, and it is the one I find most convincing. The instrumentation does not only measure speed and recall - it catches the system quietly failing. Over its life it has found, in itself, a memory that reported perfect relevance for everything, a completion gate that passed every claim unread, a promotion loop that promoted nothing, a health score pinned at zero by stale alarms, and an analytics pipeline blinded for weeks by a single string-versus-object mismatch.

THE HONESTY PAYS FOR ITSELF

At least **eight subsystems** were caught silently doing nothing while looking perfectly healthy - every one found by the system's own measurement, not by a user noticing something wrong. That is the return on building the instrumentation instead of trusting the green checkmarks: the parts that quietly break get caught before they mislead the next decision. A system that measures itself honestly is worth more than one that performs well on the day you look at it.

>_ ***A system that only claims to remember, but silently forgets, is worse than one that admits it has no memory.***

the rule that shaped everything

Finale

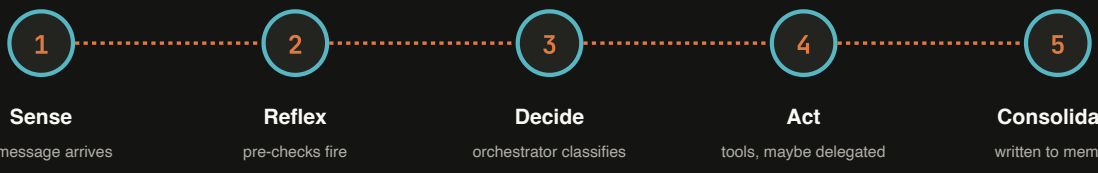
One Message, End to End

A Day in the Life

> The reflex arc as a whole: a stimulus enters through a sensory pathway, passes a spinal reflex check and a cortical decision, triggers motor output - sometimes delegated to specialist limbs - and closes the loop by writing the experience into long-term memory before the next stimulus arrives. The sense, integrate, act, consolidate cycle a nervous system runs on every input, replayed on every single turn.

40+	6	1	0
REFLEX FIRINGS PER TURN	WIRING PATHWAYS	STRICT EXIT GATE	MASTER SWITCH

Now watch one message travel through everything. This is not a component - it is the wiring between all of them, the fixed sequence every message passes through before, during and after the AI acts.



Before you even finish typing, the waking reflexes have already run: memory loaded, caches warmed, drift checked. Your message arrives and a chain of pre-checks screens it - is it urgent, should it be delegated, does it need memory recalled first. The orchestrator classifies it and decides: answer directly, hand off to a specialist, or run tools. Every tool call is wrapped by a safety check before and a logging-and-registration step after. As the AI moves to stop, the one strict gate in the whole system checks whether any done or fixed claim is actually backed by evidence. Then the closing reflexes save what was learned, distill it into permanent memory, and stage it for the next waking.

HELD TOGETHER BY WIRING, NOT A SWITCHBOARD

A single turn can pass through **40 or more** discrete reflex firings, nearly all of them invisible unless something breaks. And the whole thing is held together not by one master switch but by six independent pathways - explicit links, keyword routes, trigger phrases, the memory index, in-file mentions, and filesystem links. A component can be invisible on one pathway and fully load-bearing on another. That redundancy is the point: it is why one silent bug - a mismatch between a string and an object in how tool responses were read - could quietly blind the system's own analytics for weeks while everything else kept running. The body is resilient precisely because no single wire carries everything.

That is the anatomy. Reflexes that fire before thought, instincts that never need deciding, specialists recruited on demand, a memory that consolidates overnight, synapses that strengthen with use, and an immune system that keeps catching the parts that quietly stop working. Not a pile of files. An organism - one built, deliberately, to still be alive at the start of the next session.

Appendix - the system itself, named

The narrative showed the organs. This is the cell count: the 511 parts that operate the system itself, each with a one-line purpose where a clean one exists. Deliberately left out are the tools that merely run *on* the system to do outside work - content and social commands, media generation, external-model bridges - and the historical record (plans, handoffs, ledgers). This is the machinery that keeps the organism alive, not what it produces.

Commands 95

`/adr` - Create a new Architecture Decision Record (ADR)

`/archive`

`/audit-fix`

`/audit-report`

`/audit-security`

`/auto-model`

`/auto-run`

`/auto-validate`

`/autoevolve` - Autonomous optimization loop for LLM artifacts

`/cancel-ralph`

`/compact-stuff` - Distill session knowledge to clipboard for compact

`/compose-agent`

`/consolidate` - Detect and merge clustered experiences into consolidated

`/context`

`/context-stats`

`/create-agent`

`/create-command`

`/create-hook`

`/create-prompt`

`/create-skill`

`/create-system`

`/debug`

`/decide`

`/delegation-dashboard` - Show delegation performance: trait fitness scores

`/desk` - Decision Desk - ranked list of only the decisions that need Robin, with inline act…

`/evolving-audit`

`/fable` - Switch to Fable 5 (Anthropic's most capable model, top tier)

`/fitness-calibrate` - Manually log fitness events or override scores via the ledger

`/fitness-log` - View recent fitness events from the cognitive fitness ledger

`/fitness-report` - Show current cognitive fitness scores across lens, trait, and delegation systems

`/full-audit`

`/gamma-decide` - Action pending Gamma Photons - ship/park/drop each Annihilation insight

`/gemma-bulk` - Run bulk Gemma operations locally (JSON extract, rewrite, classify, translate) …

`/haiku` - Switch to Haiku (fast & cheap)

`/health-dashboard` - Quick visual health overview

`/ideaforge` - Repeatable discovery loop - ecosystem ingest + concept transfer into vetted…

`/inbox-process`

`/integrity-check` - Quick integrity validation without fixes

`/interview-plan` - Interview a plan using framework-aware questions across 3 depth tiers

`/inventory-report`

`/knowledge-add`

`/knowledge-refresh`

`/knowledge-search`

`/learning-review`

`/lens-calibrate` – Customize Quantum Lens configuration - enable/disable lenses, adjust depth…

`/loop-until-done`

`/map` – Map an external framework onto a target problem via the structure-mapping + TRIZ…

`/memory-stats`

`/mutation-stats` – Mutation Stats - Trait Mutation Health Check

`/opus` – Switch to Opus (maximum quality)

`/orchestrate`

`/orchestrate-cycle` – Run one orchestrator cycle (COLLECT → EVALUATE → ACT → LOG)

`/orchestrate-stop`

`/plan-new` – Create a new plan using the Universal Planning Framework

`/plan-refine` – Autonomously stress-test and harden a plan through 6 adversarial perspectives

`/plan-review` – Check plan quality against Universal Planning Framework standards

`/preferences`

`/priority-board` – Show the auto-generated Priority Board for system-structure open work

`/project-add`

`/project-analyze`

`/prompt-quality` – Scan and score all system prompts for quality

`/quantum-full` – Complete perception-to-solution pipeline: Quantum Lens deconstruction + Solution…

`/quantum-lens` – Run a radical multi-perspective deconstruction analysis using quantum-inspired…

`/quantum-review` – Review past Quantum Lens analyses - find patterns, track lens effectiveness…

`/quantum-solve` – Engineer concrete solutions from insights - evaluate repos, solve problems, break…

`/quick-audit`

`/recall`

`/refresh` – Refresh a system-structure/<slug>/ subsystem map - snapshot to history, re-map, run…

`/remember`

`/review`

`/review-plan`

`/rules-review`

`/run-prompt`

`/run-workflow` – Run Workflow

`/scenario`

`/scenario-create`

`/scenario-edit`

`/scenario-list`

`/security-review`

`/session-end`

`/skill-report` – Generate skill usage insights and recommendations from usage analytics

`/sonnet` – Switch to Sonnet (balanced performance)

`/sparring`

`/steward`

`/struct-query` – Structural code query

`/sync-all`

`/system-audit`

`/system-health`

`/template-sync` – Synchronizes generic content from Evolving to

`/think`

`/tool-map` – Generate visual Tool-Map of the Evolving System

`/update-check`

`/verify-grounding` – Ground a claim against an independent source (external for FACTS, an independent…

`/verify-plan`

`/whats-next`

Scripts

135

`anchor-ff-sync` — anchor-ff-sync.py - SessionStart hook: keep the live Evolving anchor current.…

`audit-delegation-gap-pollution` — Phase 7 (2026-05-18 onward) - audit delegation-gaps.jsonl pollution rate over time.…

`auto-adr-coverage-audit` — Read-only audit script: flag architectural decisions that lack a corresponding ADR.…

`autoevolve-scorer` — AutoEvolve Scorer - Deterministic evaluation of optimization targets. Supports…

`backfill-artifact-registration` — Backfill Artifact Registration. Scans scope paths, finds artifacts the registration…

`backfill-edge-provenance` — Backfill edge provenance fields on . Adds to each edge: - provenance: "human" |…

`backfill-experience-projects` — Backfill project_scope for existing experiences. Reads sessions-archive.json to map…

`baseline-generator` — baseline-generator: produce AUDIT-BASELINE-YYYY-MM-DD.md mechanically. Reads `…

`bayesian-experience-update` — Bayesian Experience Update: project-aware Beta-Bernoulli on effective_relevance.…

`benchmark-stop-chain` — Stop chain latency baseline analyzer. Reads cc-inspector's hook-timings.log and…

`build-code-index` — Build Code Structure Index (CSI) - Python symbol extraction via ast module.…

`build-delegation-test-cases` — Build AutoEvolve test cases for delegation-config from real signal sources. Two…

`build-dependency-map` — Build dependency map for the Evolving system. Analyzes file relationships to create…

`burn-candidates` — burn-candidates.py - rank heavy, high-value work for the burn window (Phase 1).…

`cache_writer` — Atomic-write utility for `` and similar JSON state files. Background: ~7 hooks…

`calibration_risk_precision` — calibration_risk_precision.py - precision + volume harness for the calibration-risk…

`check-count-method-parity` — CI gate: count-method parity between full-sync.sh and sync-counts.sh. THE INVARIANT…

`check-shared-write-discipline` — Wave-1 Phase 5: CI write-discipline gate for shared graph-state files. THE…

`coactivation-aggregator` — Co-Activation Aggregator — Computes pairwise node co-activation from…

`completion_linter` — Completion Linter - user-facing-text style gate (CVH Phase B). Pure-regex, no LLM.…

`compute-centrality` — Compute Centrality Scores — Katz Centrality on weighted, filtered Knowledge Graph.…

`compute-hot-pairs` — Pre-compute top N coactivation pairs for fast delegation lookup. Reads: Writes: R

`compute-load-paths` — compute-load-paths.py — Multiplex Interlayer-Gap Emitter (report-only). The…

`dashboard-context` — Context-window stats for /context-stats. Renders the final ASCII box itself…

`dashboard-delegation` — Delegation observatory for /delegation-dashboard. Renders the final ASCII box…

`dashboard-fitness` — Cognitive-fitness report for /fitness-report. Renders the deterministic report…

`dashboard-health` — Health overview for /health-dashboard. Renders the final ASCII box itself (--format…

`dashboard-memory` — Memory stats for /memory-stats. Renders the final report itself (--format ascii…

`dashboard-mutation` — Mutation stats for /mutation-stats. Renders the final ASCII box itself (--format…

`dashboard-scenario` — Scenario registry list for /scenario-list. Renders the finished scenario table as…

`dashboard-skill` — Skill/command usage report for /skill-report. Renders the deterministic sections…

`dedup-detection-index` — Deduplicate and normalize .. WHY: the detection-index auto-generator emits every…

`dedup-graph-node-twins` — Merge duplicate-path knowledge-graph node twins into one canonical node. Audit…

`dedup-knowledge-nodes` — Deduplicate duplicate-id rows in . WHY: knowledge-nodes.json accumulated…

`delegation-drift-scanner` — delegation-drift-scanner.py - producer-side regression early-warning. Reads recent…

`delegation-false-positive-eval` — Delegation False-Positive Evaluation Harness Phase 1 empirical validation sprint…

`delegation-gaps-report` — delegation-gaps-report.py - Classify delegation-gap entries by confidence. Source…

`desk-aggregator` — Decision Desk aggregator (Workstream H / H1 of the Autonomy & Harmonization…

`desk_bridge` - Desk auto-resolution bridge (observe-first) - the /desk -> /autonom rubric.…

`doctype_classifier` - Document-type pre-classifier (CVH Phase D) - deterministic, no LLM. Closes the ONE…

`drain-artifact-registration-queue` - Drain into Kairn via CLI. Each line is a kairn_add request produced by…

`dryrun_delegation_enforcer` - Dry-run capture of format_delegation_instruction output. Used by the subagent-cache…

`effort-inheritance-injector` - effort-inheritance-injector.py - Bug [5] closer. Bridges the effort_routing…

`evolve` - evolve - launch an isolated Claude Code session in its own git worktree

`evolving_audit_findings` - evolving_audit_findings.py — parse an /evolving-audit --quick report and emit…

`experience_score_writer` - Experience Score Writer (Fix 1 - QL+SE+Autorun Flywheel Repair). Updates…

`failure_prior` - Failure-Price MVP - durable failure_event lib (Phase 1). A decaying, evidence-gated…

`failure_prior_aggregate` - Failure-Price MVP - decaying, evidence-gated aggregator (Phase 2). Turns the…

`fetch-cc-changelog` - Refresh the local snapshot of the Claude Code official CHANGELOG

`fitness-calibrate-revert`

`fix-knowledge-links` - Validate and report broken links in . Scans all markdown links and checks if…

`fix-router-drifted-refs` - One-shot: repoint/remove DRIFTED non-artifact refs in context-router.json. WHY…

`flag-polluted-gap-rows` - Phase 5 (2026-05-18) - flag polluted rows in delegation-gaps.jsonl with…

`flag-pre-sister-tracker-fix-multi` - Flag pre-sister-tracker-fix rows in cognitive-fitness.jsonl AND…

`flag-pre-sister-tracker-fix-rows` - Flag pre-sister-tracker-fix rows in delegation-outcome-tracker invocations ledger…

`generate-json-summaries` - Generate JSON summaries for all MD files in the Evolving system. Creates compact…

`generate-usage-report` - generate-usage-report.py - Generate weekly usage insights Reads usage history and…

`grabbelkiste-deposit` - Grabbelkiste pending-tray deposit: the code-enforced curation valve (SC4). WHY THIS…

`grabbelkiste-next-domain` - Grabbelkiste rotation resolver: pick the next cross-domain hunt domain. WHY THIS…

`graph-integrity-restorer`

`graph-orphan-edge-monitor` - Graph Orphan Edge Monitor - READ-ONLY observer for vs . This script ONLY OBSERVES …

`handoff_indexer` - handoff_indexer: scan _handoffs/*.md and build a relevance-ranked index. Two…

`hook-timeout-scanner` - hook-timeout-scanner.py - Signal 2 BLIND closer for Bug [2]. Scans the marker…

`hook_payload` - Normalise Claude Code's PostToolUse `tool_response` across tool kinds. CC does not…

`hook_telemetry` - hook_telemetry - shared invocation/latency/crash ledger for hooks. Closes 3 of 6…

`ideaforge_vet_gate` - ideaforge VET-high HARD gate for rule-file-targeting candidates. Plan: . Closes the…

`jsonl-ledger-rotator` - jsonl-ledger-rotator.py - size/age-triggered rotator for append-only JSONL ledgers.…

`kairn-backup` - Kairn DB Daily Backup Script

`kairn-backup-freshness-scanner` - kairn-backup-freshness-scanner.py - Task #2092 backup-freshness consumer. Stats the…

`kairn-drain-freshness-scanner` - kairn-drain-freshness-scanner.py - Bug [11] freshness-consumer closer. Reads the…

`kairn-query-latency-scanner` - kairn-query-latency-scanner.py - Task #2092 latency-ledger producer+consumer. Runs…

`kairn-recall-bench` - Kairn recall-QUALITY benchmark over a frozen store snapshot. / task #2355: Kairn…

`kairn-snapshot-intraday` - Kairn DB Intraday Snapshot Script (30-minute cadence)

`kairn_promotion_observer` - Kairn Promotion/Decay Observability (Evolving Self-* Closure Program, Phase 5 SC6).…

`lock_telemetry` - lock_telemetry - cross-hook lock contention ledger. Closes Signal 5 (cross-hook…

`mapping_consistency_guard` - Mapping-operator verdict/rationale consistency guard - the "C5 guard". WHY THIS…

`mcp-bridge-worker` - MCP-Bridge Worker - Wave-1 self-driving loop BUILD #0. Drains the two MCP-blocked…

`measure-rule-tokens` - Measure token count of Core Rules. Calculates approximate token count for all rules…

`metacognitive-synthesizer` - Metacognitive Synthesizer Aggregates 3 data sources into a unified metacognitive…

`partition-edges` - partition-edges.py - Partition edges.json by type for faster loading Creates…

`phase0_5_latency_harness` - Phase 0.5 Validation Sprint - Latency Timing Harness. Measures three hot-path…

`phase2_smoke_test` - Phase 2 Smoke Test - Handlers MVP Creates a synthetic artifact in a non-conflicting…

`phase8-cache-aggregator` - phase8-cache-aggregator.py - Subagent + parent-session cache stability around Task…

`prune-dead-router-routes` - Prune / repoint dead `artifact-*` refs in . WHY: the artifact-registration-enforcer…

`qr-scan` - QR Registration Health Scanner (qr-scan.py) Cross-references all components in…

`quota-pace` - quota-pace.py - weekly Max-quota pace + burn-window signal (Phase 0). Reads `~/.`…

`rebuild-experience-index` - Rebuild experience index from filesystem. Globs *.json (top-level only, excludes…

`recalc-fitness` - Cognitive Fitness Recalculator Reads cognitive-fitness.jsonl ledger, computes…

`recall-failure-probe` - Recall Failure Probe — PostToolUse hook that logs zero-result kn_recall queries.…

`refresh-helper` - refresh-helper: deterministic operations for /refresh <slug> slash command. The…

`regenerate-detection-index` - Regenerate detection-index.json from command and skill files. Scans .md files from…

`repair-node-names` - repair-node-names.py - mop-up for cold-path node-name corruption. WHY: full-sync.sh…

`repair-routes` - Repair broken routes in context-router.json by fixing node IDs and file paths

`resilience_monitor` - resilience_monitor.py - Observe-only Resilience Monitor (Monster-Run PR-1) Three…

`route-entropy` - Route-Entropy: Shannon Entropy over node-type distribution per context-router…

`route_uptake_telemetry` - route_uptake_telemetry - observe-mode writer for external-tool route-suggestion…

`rule_miner` - Rule-miner validator + reporter (CVH Phase E) - REPORT ONLY. Consumes the…

`run_evolving_audit` - run_evolving_audit.py — the launchd + manual entrypoint that makes the dormant…

`safe-merge` - safe-merge.sh — enforce "CI green before merge" without GitHub required checks

`saturation-sentinel` - saturation-sentinel.py - turn signal-saturation into a detector. THE INSIGHT (/ …

`sc9-correction-rate` - sc9-correction-rate.py - Corrected SC9 measurement for Opus 4.7 migration plan.…

`scan-command-preexec` - scan-command-preexec.py — static gate for the un-hookable `!`-prefix class.…

`scan-explorations` - Retroactive prompt injection scanner for knowledge files. Scans all…

`scan-parallel-sessions-enrichment` - scan-parallel-sessions-enrichment.py - Archive-derived race detection enrich. Phase…

`self-heal` - self-heal.py - Self-Healing Engine for Evolving System Automatically repairs: …

`session-forensics` - session-forensics.py - CLI reconstructing what one session did. Phase 6 of…

`session-kairn-inventory` - session-kairn-inventory.py - prefill the /session-end Step 0a Kairn inventory.…

`session_attribution` - session_attribution.py - shared session-id resolution + row/doc attribution. Lifts…

`session_kairn_extract` - session_kairn_extract.py - pure extraction of Kairn saves from a CC transcript.…

`skill-drift-detector` - Skill Drift Detector - Epidemiological skill health monitoring. Reads…

`spaced-rep-collector` - Spaced Repetition Collector Collects due items for Spaced Repetition Review from: …

`status-aggregator` - status-aggregator: periodic regeneration of system-structure/ dashboards. Two…

`steward_actuator` - Steward Actuator - Phase 3 of the Evolving Self-* Closure Program. Consumes `` and…

`steward_reaper` - Steward Reaper - Phase 3 of the Evolving Self-* Closure Program. Reads ``, locates…

`steward_self_heal` - Steward Self-Heal orchestrator - Phase 7 wiring of the Evolving Self-* Closure…

`sync-counts`

`synthesis-detector` - Synthesis Detector — Finds coactivation clusters and writes synthesis candidates.…

`synthesis-spine` - synthesis-spine.py -- Synthesis Spine producer (Phase 2). Implements G1-G6…

`synthesis_closer` - Synthesis LEARN-closer - Phase 4 of the Evolving Self-* Closure Program. Closes the…

`test_coordination` - Tests for quantum coordination primitives (system-pulse, active-intents, hot-pairs)

`test_coordination_integration` - Integration tests for coordination primitives. Runs the actual…

`test_saturation_sentinel` - Tests for saturation-sentinel.py (TDD). The Saturation Sentinel turns the QL…

`thinking-recall-fitness-feed` - thinking-recall-fitness-feed.py - Mine archived thinking-recall snapshots. Phase 6…

`tmp-archive-rotator` - tmp-archive-rotator.py - monthly cold rotator for the session archive. Phase 7…

`tmp-archiver` - tmp-archiver.py - SessionStart hook: archive session-scoped /tmp state. Phase 3…

`tmp-lifecycle-scanner` - tmp-lifecycle-scanner.py - Signal 6 BLIND closer for Bug [2]. Surveys `/tmp` for…

`tool_fit_precision` - tool_fit_precision.py - precision + volume harness for the Right-Cases Tool-Fit…

`toolbox` - Framework Toolbox - persistent, never-delete framework/principle library. One entry…

`trim-index-json` - Write content to file atomically via temp file + rename

`trim-project-progress` - Trim one project file; returns number of archived entries

`update-system-inventory`

`v2_bridge_spike` - AutoEvolve V2 Bridge Spike - delegation-gaps.jsonl -> cognitive-fitness.jsonl…

`v2_runner_helpers` - AutoEvolve V2 Runner Helpers - CLI hooks the autoevolve-optimizer agent calls via…

`validate-routes` - Validate context-router.json routes for broken links. Checks all 71 routes in the…

`voice-out` - voice-out.sh - control the Local Voice Bridge output side (Stop-hook TTS)

Rules 66

`README` - Claude Code Rules

`_template`

`artifact-registration` - Artifact Registration

`auto-archival`

`auto-enhancement` - Auto-Enhancement Thinking Pattern

`auto-learning`

`auto-rule-generation` - Auto Rule Generation

`autonomy-classifier` - Autonomy Classifier

`clear-dont-compact` - Handoff + Compact, Not /clear

`command-creation`

`context-budget-awareness`

`context-efficiency` - Context Efficiency System

`context-optimization` - Context Optimization Rule

`correction-learning-feedback` - Correction Learning Feedback

`cross-reference-sync` - Cross-Reference Sync

`delegation`

`delegation-extended` - Delegation Extended

`domain-memory-bootup` - Domain Memory Bootup Ritual

`domain-memory-bootup-advanced` - Domain Memory Bootup - Advanced Patterns

`dsv-principle` - DSV Principle (Decompose - Suspend - Validate)

`evidence-before-claims` - Evidence Before Claims

`experience-suggest` - Experience Auto-Suggest Rule

`explicit-identity` - Explicit Identity Across Boundaries

`explore-index-check` - Explore Index Check

`failure-recovery` - Failure Recovery

`hook-block-response` - Hook Block Response

`idempotent-redundancy` - Idempotent Redundancy

`index-at-creation` - Index at Creation Time

`index-auto-sync` - Index Auto-Sync Rule

`kairn-workarounds` - Kairn Workarounds

`knowledge-linking`

`knowledge-refresh-cycle` - Knowledge Refresh Cycle

`language-frame-discipline`

`memory-decay` - Memory Decay Rule

`metacognitive-orchestrator`

`mid-session-kairn` - Mid-Session Kairn (Project Delta)

`no-reference-only` - No Reference-Only Rule

`observe-before-editing` - Observe Before Editing

`orchestrator-extended` - Metacognitive Orchestrator - Extended

`plan-archival`

`plan-execution` - Plan Execution Rules

`plan-execution-enforcement` - Plan Execution Enforcement

`plan-execution-workflow` - Plan Execution Workflow

`plan-mode-upf-bridge` - Plan Mode - UPF Bridge

`plan-pipeline-automation` - Plan Pipeline Automation

`plan-quality-activation` - Plan Quality Activation

`post-todo-review` - Post-Todo Review

`proactive-behavior` - Proactive Behavior

`proactive-doc-sync` - Proactive Documentation Sync

`prompt-injection-awareness` - Prompt Injection Awareness

`quick-dsv`

`relevance-extraction` - Relevance Extraction Rule

`scenario-agents`

`self-evolving-rule-generation` - Self-Evolving Rule Generation

`session-evaluation` - Session Evaluation Rule

`stop-chain-ordering-contract`

`sub-agent-delegation` - Sub-Agent Delegation

`swarm-orchestration` - Swarm Orchestration

`tier-system` - Security Tier System

`token-efficiency` - Token Efficiency Rule

`token-sustainability` - Token Sustainability

`trait-mutation` - Trait Mutation on Failure

`trait-system`

`trivial-prompt-noise-suppression`

`update-monitor` - Update Monitor

`workflow-detection`

Hooks 65

`artifact-registration-enforcer` - Artifact Registration Enforcer (PostToolUse[Write]). Real-time complement to…

`auto-archival` - Auto-Archival Hook Central orchestrator for automated data cleanup and archival.…

`auto-cross-reference`

`auto-mode-auditor` - auto-mode-auditor.py - PermissionDenied event logger (permanent audit trail)…

`auto-run-gate-keeper-observer`

`auto-sync-validator` - Auto-Sync Validator - Stop Hook Validates system state at session end. Logs results…

`autonom-notice` - Autonom SessionStart Notice - Phase 4, part (b). STAGED - NOT REGISTERED. TO APPLY…

`autonom-spine-guard` - autonom-spine-guard.py - LIVE PreToolUse guard…

`calibration-risk-suggester` - Calibration-Risk Suggester (Stop hook, OBSERVE-MODE-FIRST). Trigger: at a `Stop`…

`code-index-startup` - SessionStart hook: Auto-rebuild code-index.json if missing or stale. Checks: 1.…

`code-index-updater` - PostToolUse hook: Incrementally update code-index.json after Write/Edit on .py…

`code-review-reminder` - PostToolUse Hook: Code Review Reminder Triggers after Write/Edit on code files to…

`config-change-audit` - ConfigChange hook - Audit logging for configuration changes. Logs all configuration…

`content-scanner` - Content Scanner - Prompt Injection Defense for External Web Content Post

`context-warning` - Context Warning Hook - PreToolUse (v6 - Handoff + Compact Loop)

`correction-detector` - Correction Detector Hook - Self-Evolving System (v2) Detects user corrections in…

`deferred-classification-injector` - Deferred-Classification Injector (UserPromptSubmit hook). Trigger: Robin's prompt…

`delegation-enforcer` - Delegation Enforcer Hook (Multi-Event) Supports multiple hook events: …

`delegation-outcome-tracker` - Delegation Outcome Tracker (V2 Phase 0.5). Bridges UserPromptSubmit -> Stop event…

`desk-badge` - Decision Desk badge - SessionStart push (Workstream H3, STAGED). Emits ONE line…

`experience-access-tracker` - Experience Access Tracker - PostToolUse:Read hook Tracks when experience files are…

`forced-verify-stop-gate` - forced-verify-stop-gate.py - LIVE Stop hook for the EPT Stop-gate. INPUT FIELD…

`graph-orphan-surface` - graph-orphan-surface: SessionStart READ-ONLY consumer for the broken-edge monitor.…

`handoff-surface` - handoff-surface: SessionStart READ-ONLY consumer for the handoff index. Surfaces…

`instructions-loaded-tracker` - InstructionsLoaded Tracker - logs which rules are loaded per session. v3.1: Fixed…

`integration-matrix-checker` - Integration Matrix Checker Hook (SessionStart Event) Detects SYSTEM file edits in…

`kairn-call-tracker` - kairn-call-tracker.py — PostToolUse hook for `mcp__kairn__*` tools. Records the…

`kairn-first-enforcer` - kairn-first-enforcer.py — UserPromptSubmit hook. Detects domain-knowledge keywords…

`kairn-sync-enforcer` - Kairn Sync Enforcer Hook (SessionStart Event) Checks for pending Kairn sync…

`kairn-sync-staging` - Kairn Sync Staging Hook (Stop Event) Scans for new/modified knowledge files since…

`lazy-deferral-blocker`

`ledger-auto-save`

`lens-fitness-drainer` - Lens Fitness Drainer (Phase 3.5 producer pipeline). Drains per-session lens fitness…

`notification-sound` - Notification Sound Hook - Debug Version

`plan-rot-detector` - Plan-Rot Detector Hook (SessionStart Event) Checks if the active plan is stale by…

`post-compact-state-save` - PostCompact State Save Hook (v1.0)

`post-tool-tracker` - post-tool-tracker.py - Unified PostToolUse tracker (v1.0) Created: 2026-03-28…

`precompact-extract` - PreCompact Knowledge Extraction Hook (Layer 3 Safety Net) Fires before autocompact.…

`ralph-loop-stop`

`rate-limit-watchdog` - Rate Limit Watchdog for Claude Code Sessions

`recurring-task-checker` - Recurring Task Checker Hook (SessionStart Event) Scans *.json for recurring_tasks…

`recurring-task-completer` - Recurring Task Completer Hook (Stop Event) Counterpart to recurring-task-checker.py…

`sanitizer` - Agent Output Sanitizer - Cross-Session Injection Defense Scans sub-agent output fo

`security-tier-check` - Security Tier Check Hook for Claude Code PreToolUse hook that checks Bash commands…

`session-end-cleanup`

`session-journal` - Session Journal Hook (PostToolUse) Automatically logs significant tool actions to a…

`session-summary`

`steward-checker` - SessionStart hook: consolidated steward checker. Date-routed dispatch across three…

`steward-pending-surface` - Steward Pending Surface - SessionStart read-only consumer (Phase 7-partial).…

`stop-failure-handler` - StopFailure Hook (v1.0)

`subagent-router` - SubagentStop Router Hook for Claude Code Automatically routes subagent outputs to…

`synthesis-approval-surface` - Synthesis Approval Surface - SessionStart read-only consumer (Phase 4). Surfaces…

`synthesis-spine-scheduler` - synthesis-spine-scheduler.py - - SessionStart producer for the Synthesis Spine (task…

`system-structure-stale-marker` - system-structure-stale-marker: PostToolUse hook for freshness contract. When…

`task-completion-gate` - TaskCompleted hook - Quality gate for agent team tasks. Validates that code-related…

`teammate-idle-check` - TeammateIdle hook - Prevents premature teammate idle. Checks if there are unclaimed…

`test-content-scanner` - Integration tests for content-scanner.py PostToolUse hook

`test_thinking_recall_dual_throttle` - Tests for the Phase 13.B dual-throttle gate in thinking-recall.py. These tests…

`test_thinking_recall_sec001` - SEC-001 regression tests for thinking-recall.py path-boundary checks. Locks in the…

`test_verify_before_change` - Tests for verify-before-change.py (task #2205). The hook is a…

`thinking-recall` - Thinking Recall - Mid-Stream Memory Injection Hook PreToolUse hook that reads…

`tmux-hint` - tmux-hint.py - Suggest tmux for long-running dev servers Hook: PreToolUse (Bash)…

`usage-tracker` - usage-tracker.py - Track all tool usage with detailed analytics Hook: PostToolUse…

`verify-before-change` - verify-before-change.py - PreToolUse[Edit|Write] advisory (task #2205). Before an…

`voice-out-speak` - voice-out-speak.py - Stop hook: speak the main turn's reply via local pocket-tts.…

Patterns 59

`8-block-profile-system`

`README` - Patterns

`agentic-maturity-model-pattern`

`ai-content-generation-pipeline`

`auto-prompt-enhancement-pattern`

`blackboard-pattern`

`browser-automation-pattern`

`checkpoint-validation-pattern`

`compact-errors-pattern`

`comprehensive-audit-pattern`

`confidence-scoring-pattern`

`context-window-ownership-pattern`

`delegation-request-pattern`

`ensemble-pattern`

`explicit-identity-pattern`

`four-bucket-context-pattern`

`freemium-saas-gates`

`gdpr-compliant-saas`

`hook-development-reference`

`hook-patterns-library`

`idempotent-redundancy-pattern`

`intake-gate-pattern`

`interview-before-implement-pattern`

`iterative-retrieval-pattern`

`learning-mode-pattern`

`lock-methodology-pattern`

`manager-pattern-multi-agent-systems`

`memory-architecture-pattern`

`meta-component-creation-pattern`

`metacognitive-pattern`

`ml-project-structure-pattern`

`multi-agent-orchestration`

`multi-source-aggregation-pattern`

`multi-strategy-retrieval-pattern`

`observation-compression-pattern`

`parallel-agent-dispatch-pattern`

`pev-pattern`

`pkm-digital-brain-pattern`

`progressive-disclosure-pattern`

`ralph-wiggum-loop-pattern`

`react-pattern`

`recursive-research-pattern`

`reflection-pattern`

`research-confidence-scoring`

`research-orchestration-pattern`

`resume-strategies-pattern`

`safety-hooks-framework-pattern`

`security-review-pattern`

`self-assessment-rubric-pattern`

`self-improving-rules-pattern`

`spaced-repetition-pattern`

`sub-agent-delegation-pattern`

`superposition-reasoning-pattern`

`system-generation-pattern`

`systematic-debugging-pattern`

`task-decomposition-pipeline`

`technical-code-review-pattern`

`tree-of-thoughts-pattern`

universal-planning-framework

Agents 52

agent-factory - Dynamic agent composition from trait system

api-routes-auditor-agent - API endpoint coverage, REST practices, error responses

architecture-auditor-agent - System architecture, design patterns, scalability assessment

audit-coordinator-agent - Orchestrates all audit phases, manages checkpoints, synthesizes results

audit-intake-agent - Project profiling, tech stack detection, agent selection

audit-reporter-agent - Synthesizes audit findings into comprehensive reports

automation-setup-agent

business-logic-auditor-agent - Gap-to-goal analysis, feature completeness, business rules

code-quality-auditor-agent - Code quality analysis, tech debt, testing practices

code-reviewer-agent

codebase-analyzer-agent - External codebase analysis with context persistence and multi-agent orchestration

compatibility-checker-agent

content-anonymizer-agent - Transforms personalized content into generic templates

context-manager-agent - Context sharing, persistence, and coordination across multi-agent systems

cross-reference-checker-agent - Validates consistency between ALL Master-Docs, _stats.json and actual structure.…

cross-reference-fixer-agent - Fixes cross-reference inconsistencies between master docs, _stats.json, and actual…

debugger - Debugging specialist for errors, test failures, and unexpected behavior. Use…

dependency-checker-agent - Checks finding dependencies - tool mutex, naming collisions, pattern overlap…

dependency-fixer-agent - Repairs broken registrations, creates missing edges, syncs counts. Use after…

detection-index-checker-agent - Validates Command + Skill Detection Index - Keywords, Coverage, Context-Router…

detection-index-fixer-agent - Fixes detection index issues - missing entries, orphan commands/skills, keyword…

doc-sync-agent

documentation-auditor-agent - Documentation completeness, README quality, API docs

file-hygiene-auditor-agent

file-hygiene-fixer-agent - Fixes file hygiene issues - duplicates, oversized files, stale backups, legacy…

github-repo-analyzer-agent - Dual-System Repository Analyzer with SYSTEM-MAP integration and contextual findings

graph-node-creator-agent - Creates Knowledge Graph nodes from file metadata

graph-orphan-detector-agent - Detects files without corresponding Knowledge Graph nodes

harmony-checker-agent - Checks style consistency with dynamic domain expert via trait system. Use for…

integration-orchestrator-agent - Coordinates fully automated integration of findings with 4 specialized sub-agents.…

knowledge-graph-fixer-agent - Repairs knowledge graph issues - orphan edges, missing nodes, count mismatches…

knowledge-graph-validator-agent - Validates Knowledge Graph Integrity - Nodes, Edges, Orphans, Legacy Status…

memory-cleanup-fixer-agent - Cleans up domain memory - stale backups, expired experiences, progress overflow…

memory-schema-validator-agent - Validates Domain Memory Schema - Progress, Failures, Experiences, Backups…

performance-auditor-agent - Performance bottlenecks, resource usage, optimization

planner - Autonomous plan quality reviewer using Universal Planning Framework

privacy-scanner-agent - Scans files for sensitive/personal content - before sync AND validates entire…

prompt-auditor-agent - AI prompt quality, model usage, token efficiency

remediation-orchestrator-agent - Executes remediation plan with semi-auto workflow, git commits, verification, and…

`research-analyst-agent` - Multi-source research and validation agent with confidence scoring

`security-auditor-agent` - OWASP-class vulnerabilities, injection (SQL/command/prompt), auth/authz flaws…

`silent-failure-hunter` - Audits a code change for silent failures, swallowed errors, overly-broad or empty…

`system-analyzer-agent`

`system-architect-agent` - Designs architecture for generated systems - agent roles, model tiering, knowledge…

`system-generator-agent` - Generates all files for new systems from architecture specs - templates, agents…

`system-validator-agent` - Validates generated systems for completeness and correctness - structure…

`team-composer-agent` - Analyzes audit findings and dynamically composes optimal fixer team via trait-based…

`template-diff-agent` - Intelligent diff analysis between source and target repositories

`template-inventory-agent` - Analyzes Evolving-Template repo and creates inventory comparison with source

`tool-inventory-agent` - Discovers tools, finds orphans, generates Tool-Map

`whats-next` - Whats Next

`workflow-optimizer` - Use this agent for optimizing human-agent collaboration workflows and analyzing…

References 31

`FRAMEWORKS` - Thinking Frameworks & Methodologies

`PATTERN-SELECTOR` - Pattern Selector

`anthropic-skill-best-practices` - Anthropic Skill Best Practices Reference

`audit-hidden-dependencies` - Audit Hidden Dependencies Reference

`audit-tmp-predictable-path-2026-05-21` - Audit: /tmp Predictable-Path Pattern (Statusline-Bug Class)

`awesome-claude-resources` - Awesome Claude Resources

`brainstorming`

`calibration-risk-trigger` - Calibration-Risk Trigger (operator reference)

`cc-wf-studio-reference` - CC Workflow Studio Reference

`claude-behavior-deltas-2026-07` - Claude Behavior & Platform Deltas (2026-07)

`claude-code-changelog` - Changelog

`claude-code-plugins-guide` - Claude Code Plugins Guide

`claude-code-system-prompts` - Claude Code System Prompts Reference

`claude-config-ab-tests-2026-07` - Claude Config A/B Tests (2026-07)

`claude-skills-generator` - Claude Skills Generator

`evolve-command` - `evolve` - isolated Claude Code session per worktree

`gemma-best-practices`

`grounding-primitive` - External Grounding Verification Primitive

`index`

`just-bash-reference` - just-bash Reference

`kairn-tools-reference` - Kairn MCP Tools Reference

`mcp-tools-reference` - MCP Tools Reference

`mega-template`

`quarantine-audit-2026-06` - Quarantine Audit - Untrusted-Content Pipelines (2026-06)

`synthesis-contract` - Synthesis Spine - Precision Contract

`systematic-debugging`

`telemetry-premise-ledger-map` - Telemetry Premise -> Ledger Map

`test-driven-development`

`token-economics` - Token Economics Reference

`tool-comparison-matrix`

`writing-and-executing-plans`

Skills 8

`adr` - Architecture Decision Record system

`backlog-triage` - Audit open issues and PRs across solo-dev GitHub repos, classify with 6…

`careful` - Destructive-operation guards for current session

`freeze` - Freeze file modifications outside a directory

`inventory-report` - Complete tool inventory report generation

`prompt-hardening` - Post-creation quality assurance for prompts

`qcr-refine` - Apply QCR-lite refinement (Compaction + Annihilation + Ice-9) to any content …

`quantum-deconstruction` - Multi-perspective deconstruction with quantum lenses