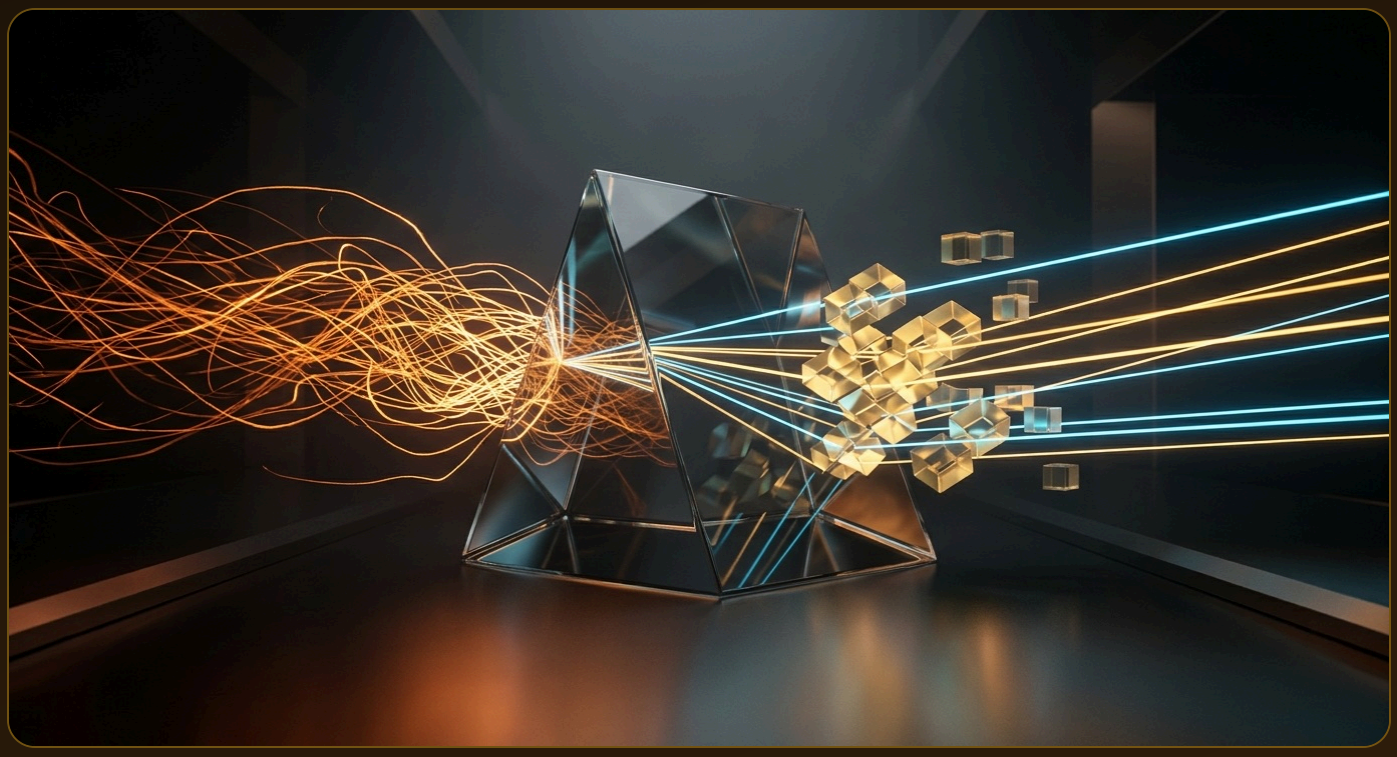
AI AGENT ENGINEERING SERIES • V1.1 EDITION

Mastering Spec-Driven Dev & Context Engineering

The complete visual handbook to ending agent thrash: architecting lean cross-vendor context files, progressive disclosure layers, executable `Spec → Plan → Tasks` workflows, and rock-solid session hygiene.

One Canon (`AGENTS.md`) | `CLAUDE.md` & `.mdc` Adapters | 4-Layer Progressive Disclosure | `Spec → Plan → Tasks` | Verifiable Acceptance Checks | Compaction & Handoff Hygiene



One Canon, Thin Adapters

Maintain one crisp `AGENTS.md` shared across Claude Code, Cursor, Antigravity, Copilot, Gemini CLI, and Codex.

60–75% Leaner Context

Eliminate attention dilution with a 4-layer stack of path-scoped globs, trigger pointers, and on-demand skills.

Proof Over Vibe-Prompting

Replace fragile mega-prompts with durable repo specs and runnable acceptance checks that any fresh agent can verify.

Why Agents Thrash & The Core Mental Model

“The spec is the source of truth; the prompt is just the delivery truck.” When an AI coding agent loops, forgets conventions, or ships plausible-looking code that fails review, typing a longer, sterner chat prompt is the most expensive trap in modern software engineering.

Quick Navigation

1. Why Agents Thrash & Mental Model p. 2
2. 5-Min Audit & Cross-Vendor Matrix p. 3
3. Annotated “One Canon” Repo Tour p. 4
4. 4-Layer Progressive Disclosure p. 5
5. High-Signal `AGENTS.md` & JIT Triggers p. 6
6. Workflow: `Spec → Plan → Tasks` p. 7
7. Verifiable Criteria & Context Rot p. 8
8. 5 Copy-Paste Context & Spec Recipes p. 9
9. Team Scale, Monorepos & Git Hygiene p. 10
10. End-to-End Synergy Walkthrough p. 11
11. Troubleshooting & Pre-Flight Check p. 12
Core Principle: Durable Files Beat Chat All

Why Longer Prompts Lose (And Durable Files Compound)

A chat prompt is ephemeral: it lives in one window, for one agent session, invisible to teammates, and evaporates on compaction. Durable repository files (**Context Files** as the employee handbook, and **Specs** as the work order) compound across every future session, subagent, and engineer on your team.

Architecture: “One Canon, Thin Adapters” + Executable Work Orders



The Whole Philosophy in Three Sentences

1. Write down what you repeat: If you correct an agent twice, put that rule in a repo file, not your next chat message. **2. Pay for context only when needed:** Load deep reference manuals just in time instead of taxing every turn. **3. Make “done” executable:** Phrase acceptance criteria as runnable CLI checks so the machine verifies its own work.

5-Minute Repo Audit & Cross-Vendor Context Matrix

Every major AI coding tool reads persistent Markdown context from your repository, yet each uses different filenames, glob frontmatter, and load mechanics. Run this 5-minute audit before writing a single new rule.

Prerequisites & The 5-Minute Readiness Audit

1 CLI Verifiability

Confirm build, types, and tests run from one shell command (`pnpm check`, `pytest`, `cargo test`). Without CLI checks, agents only guess.

2 Line & Fossil Audit

Run `wc -l AGENTS.md CLAUDE.md`. Any root file over `300` lines (or citing renamed scripts) actively degrades accuracy.

3 Behavioral Probe

Ask a fresh session: *"List the exact build/test commands and hard rules loaded, plus source files."* Mismatches expose blind spots.

Cross-Vendor Context File Decision Matrix (2026 Standard)

Ecosystem	Global & Project Root Files	Path-Scoped / On-Demand	Verified Loading Mechanics
Open Canon `AGENTS.md` + `SKILL.md` (LF AAIF)	`./AGENTS.md` at repo root (plain Markdown, ≤300 lines)	Nested `apps/*/AGENTS.md` + portable `SKILL.md`	Read natively by **20+ tools**. Nearest file wins; Codex caps `32 KiB` & skips untrusted repos.
Claude Code Anthropic CLI & IDE	`~/.claude/CLAUDE.md`, `./CLAUDE.md` (or native `AGENTS.md` fallback), `CLAUDE.local.md`	`.claude/rules/*.md` (`paths:` globs) + `MEMORY.md` (200-ln cap)	Concatenates root-down (`@file` expands inline). Audit via `/context` & `/memory`.
Cursor AI Code Editor	`.cursor/rules/*.mdc` + native `AGENTS.md` + Team Rules	4 modes: `alwaysApply`, Auto (`globs:`), Agent (`description:`), Manual	Requires `.mdc` (legacy `.cursorrules` deprecated). Precedence: **Team > Project > User**.
Antigravity & Gemini CLI Google Agent Suite	`~/.gemini/GEMINI.md`, repo `AGENTS.md` / `GEMINI.md`, `.agents/rules/`	`.agents/skills/<name>/SKILL.md` + JIT subtree discovery	`AGENTS.md` native (or set `context.fileName` in `settings.json`); check `/memory show`.
Copilot, Codex & Windsurf GitHub • OpenAI • Codeium	`AGENTS.md` across Copilot Chat/Cloud/CLI, Codex & Windsurf	`.github/instructions/*.instructions.md` (`applyTo` globs)	Copilot also reads `CLAUDE.md` & `copilot-instructions.md`. Prefer `AGENTS.md` over `.windsurfrules`.

Crucial Honesty Rule: Prose Concatenates, Permissions Enforce!

Across tools, Markdown instruction files accumulate in context rather than deterministically overriding one another. If two files say *"use 2 spaces"* and *"use tabs,"* the model flips a coin. De-duplicate prose to a single home, and enforce hard bans (like blocking `rm -rf` or `git push`) via `settings.json` permissions, OS sandboxes, or hooks.

Annotated Tour: The “One Canon” Repository Layout

Here is how a clean, multi-tool repository looks in practice. Every project-wide convention lives once inside `AGENTS.md`, tool adapters stay under 35 lines, and deep subsystem docs or feature specs stay out of memory until needed.



Numbered Architecture Reference

1 **Root Canon (`AGENTS.md`):** The single source of repo-wide commands, 3–7 non-negotiables, and trigger pointers. Never duplicated elsewhere.

2 **Thin Adapter (`CLAUDE.md`):** Pulls in `@AGENTS.md` on line 1 and adds *only* Claude-specific mechanics (plan threshold, compact instructions).

3 **Path-Scoped Glob Rules (`.mdc` / `.claude/rules`):** Attach React, SQL, or API rules *only* when the agent touches matching file globs.

4 **On-Demand Reference (`docs/agent/*.md`):** Deep architecture guides, migration runbooks, and testing handbooks read solely on trigger.

5 **Feature Spec Folder (`docs/specs/<slug>/`):** Holds `spec.md`, `plan.md`, and `tasks.md` under version control alongside your code.

6 **Staleness Circuit-Breaker:** Explicitly authorizes the agent to trust live code over outdated docs—and report the stale line aloud!

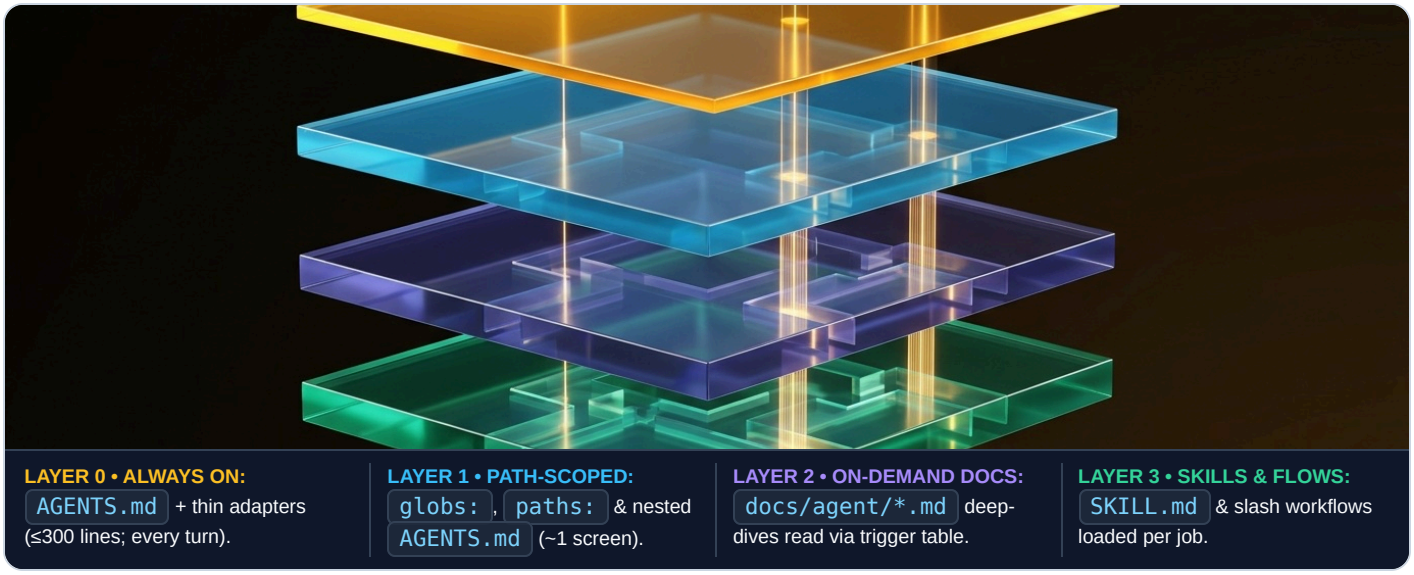
7 **Executable Checklist (`tasks.md`):** Tracks verified progress on disk (`[x]` = verify command exited `0`) so any fresh session can pick up instantly at `T3`.

House Rule of the Layout: One Fact Lives in One Place

The moment you copy a convention from `AGENTS.md` into `CLAUDE.md` or `00-core.mdc`, you schedule a future **Conflict Weave** bug when one copy gets updated and the other rots. Adapters point back to the canon—never restate it!

Progressive Disclosure & The 4-Layer Architecture

Every line in your root context file is re-processed on *every turn of every session*. Bloated 700-line handbooks create three compounding penalties: **attention dilution** (crucial rules get lost in the middle), **silent rule collisions**, and **broken prompt-cache prefixes**.



LAYER 0 • ALWAYS ON: `AGENTS.md` + thin adapters (≤300 lines; every turn).

LAYER 1 • PATH-SCOPED: `globs:`, `paths:` & nested `AGENTS.md` (~1 screen).

LAYER 2 • ON-DEMAND DOCS: `docs/agent/*.md` deep-dives read via trigger table.

LAYER 3 • SKILLS & FLOWS: `SKILL.md` & slash workflows loaded per job.

Before & After: Putting a 740-Line `CLAUDE.md` on a Diet

✗ Before: 740-Line "Omnibus" File

- **Pep-talk filler:** 4 paragraphs of *"You are an expert senior TypeScript architect, write clean code..."*
- **Stale traps:** Lists `pnpm lint` and `pnpm deploy` (renamed months ago—still followed blindly!).
- **Internal contradictions:** Says *"2-space indent"* on line 42 and *"tabs in frontend"* on line 510.
- **Inline manuals:** 90-line DB migration runbook + 120-line testing essay taxing simple CSS edits.
- **Grievance log:** 60 lines of dated complaints (*"In March you used moment.js..."*).

✓ After: 95-Line Layered System

- **Lean Canon (55 lines):** 2-line repo summary, verified commands + runtimes, and 5 crisp non-negotiables.
- **Thin Adapter (40 lines):** `CLAUDE.md` imports `@AGENTS.md` and sets compaction rules.
- **Path Resolution (Layer 1):** `.claude/rules/frontend.md` scopes tab formatting strictly to `apps/web/**/*`.
- **On-Demand Lore (Layer 2):** Runbooks moved to `docs/agent/migrations.md` behind a 5-row pointer table.
- **Result: 60–75% fewer prefix tokens** per turn and zero stale-command misfires!

The Week-One Hire Litmus Test (Keep Cures, Delete Symptoms)

Before adding any line to Layer 0, ask: *"Would a capable new engineer need this in week one on almost every task?"* If it applies to only one folder or workflow, push it to Layer 1, 2, or 3. And replace sprawling incident grievances with positive cures (e.g., *"Dates use Temporal/date-fns; never import moment"*).

Anatomy of a High-Signal `AGENTS.md` & JIT Triggers

Frontier coding models do not need flattery or restatements of standard ESLint rules; they can read your linter config directly. What they cannot guess are your repo's architectural boundaries, command quirks, and where hidden bodies are buried.

The 6-Part High-Signal Context Blueprint (Color-Coded Anatomy)

[1. AUDIT HEADER & MAP] <!-- Verified: 2026-11 • Owner: @platform --> TS monorepo: apps/web, apps/api, packages/core (pure domain, zero I/O).

[2. EXACT COMMANDS + TIMING] Build: pnpm build (root only) • Check: pnpm check • Unit: pnpm test (~40s; do not abort early).

[3. RANKED NON-NEGOTIABLES] 1) DB queries ONLY in packages/core/src/repos/*. 2) Routes MUST call requireRole(). 3) Money = Decimal, never number.

[4. STALENESS BREAKER] If this file contradicts live code or package.json, trust the code, state the mismatch, and propose an edit here.

[5. TRIGGER POINTERS] Touching exports → read docs/agent/export.md FIRST • Editing DB schema → read docs/agent/migrations.md FIRST.

[6. COMPACT SURVIVAL] On /compact, preserve: active spec path, verified [x] vs pending [ ] tasks, and any session-specific "never do X" rules.

Scoping Rules: From Global User Defaults Down to Subtrees

Scope Tier	Example Paths	What Belongs Here (And What Never Does)
Global User Scope	`~/.claude/CLAUDE.md` `~/.gemini/GEMINI.md`	Your personal communication style, OS shell preferences, and cross-project git habits. **Never** put repo-specific rules here where teammates lack them.
Project Root Canon	`./AGENTS.md` + thin `./CLAUDE.md`	Repo-wide build/test commands, architectural boundaries, security invariants, and the Layer-2 trigger index. Checked into Git.
Personal Local	`./CLAUDE.local.md`	Your local sandbox URLs, test user credentials pointers, or WIP experiments for this repo. Always add to `.gitignore`.
Subtree / Glob	`apps/web/AGENTS.md` `.cursor/rules/*.mdc` (`globs:`)	*"Extra law for this neighborhood only"* (e.g., TanStack Query + colocated `.test.tsx` inside `apps/web`). Never re-state root policy.

Making Just-in-Time Pointers & Skills Fire Reliably

A Write Triggers, Not Bibliographies

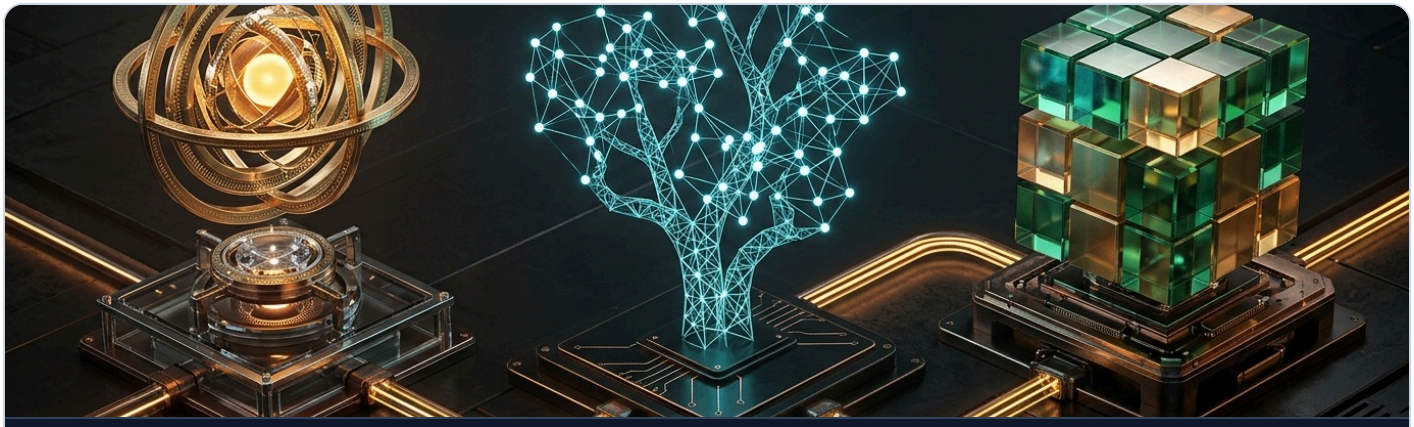
Agents skip passive lists like *"See docs/agent/ for helpful guides."* Write explicit **IF → THEN** conditions: `If your task touches billing or webhooks → read docs/agent/stripe-webhooks.md before planning.`

B Graduate Workflows to `SKILL.md`

Now an LF AAIF open standard (`agentskills.io`) across Antigravity, Claude, Cursor, Codex & Copilot: unlike inline `@file` imports, only YAML `description` loads at boot; full SOPs load strictly on match!

The Spec-Driven Workflow: `Spec → Plan → Tasks`

When you jump straight from a vague chat prompt to code generation, the agent silently guesses scope, auth, formatting, and scale. Splitting work into three small Markdown files shifts your human effort to the highest-leverage 15 minutes: reviewing the route *before* wheels roll.



1. spec.md (WHAT & WHY):
Goals, explicit Non-Goals, requirements & runnable Acceptance Criteria.

2. plan.md (TECHNICAL HOW):
File-by-file map, edge cases owned, test strategy & rejected alternatives.

3. tasks.md (EXECUTION UNITS):
Dependency-ordered steps (`T1..Tn`), each paired with an exact `Verify:` command.

Plain Markdown vs. Spec Toolkits: Choosing Your Machinery

Approach / Toolkit	Core Commands & Artifact Flow	Best Fit & Trade-Offs
Plain Markdown Zero-Install Baseline	`docs/specs/<slug>/{spec,plan,tasks}.md`	**Start here!** Zero dependencies, zero lock-in, works identically across every CLI, IDE, and cloud agent today.
GitHub Spec Kit (`v1.4.4`) `specify init` / `self upgrade`	`/speckit.constitution → .specify → .clarify → .plan → .tasks → .implement → .converge`	Install via `uv tool install specify-cli` ; supports 40+ agents, reasoning-effort mapping, and `T-Shirt Sizing Gate` , `DocGuard` & `Assure` extensions.
OpenSpec (OPSX) `@fission-ai/openspec` v2.1+	`/opsx:explore → /opsx:propose → /opsx:apply → /opsx:verify → /opsx:archive`	`openspec init --scope global|project` & `openspec update` ; brownfield delta specs (`ADDED / MODIFIED / REMOVED`).
Native Agent Modes Antigravity • Claude • Gemini	`/grill-me` + `/plan` Artifacts, Claude Plan Mode, Gemini CLI `/spec:create`	Built right into your daily harness—pair them with a committed `docs/specs/` file so plans survive beyond the chat!

Honest Sizing Rule: When Lazy One-Shot Prompts Win vs. When Specs Win

Skip the spec for disposable spikes, throwaway scripts, or visual exploration where blast radius is zero. **Always write a spec** when work touches auth, money, database schemas, public APIs, or spans multiple sessions/teammates. Never build production-grade features with demo-grade one-liners!

Verifiable Acceptance Criteria & Stopping Context Rot

Two habits separate effortless agent sessions from late-afternoon meltdowns: phrasing every requirement so the agent can *prove* it mechanically, and managing your context window before "lost in the middle" attention poverty sets in.

1. Rewriting Vague Wishes Into Agent-Verifiable Criteria

Vague Wish (Unverifiable Vibes)	Executable Criterion (`GIVEN / WHEN / THEN` + Named Check)
"Export should be fast & scale well"	**AC-1:** GIVEN `10,000` seeded rows WHEN `GET /export` runs THEN completes in `<5s` with RSS delta `<150MB` • `Verified by: pnpm test -- export.perf`
"Only team admins can export"	**AC-2:** GIVEN role `member` WHEN `GET /export` THEN HTTP `403`; GIVEN `admin` THEN `200` + DOM button visible only to admin • `Verified by: export.auth.test.ts`
"Format the CSV cleanly"	**AC-3:** UTF-8, RFC-4180 escaped, header `Date,Merchant,Amount,Category,Owner`, ISO-8601 dates, 2-decimal `Decimal` • `Verified by: export.format.test.ts`
"Handle errors & edge cases"	**AC-4:** Zero-row month returns `200` header-only CSV; malformed `month=05/2026` returns `400` `{"error":"invalid_month"}` • `Verified by: export.edge.test.ts`
"Don't break existing features"	**AC-5:** `pnpm check && pnpm test` both exit `0` with zero previously passing tests deleted or skipped • `Verified by: paste command output tail`

2. Context Window Economics: `/clear` vs. `/compact` vs. Handoffs

`/clear` (Unrelated Switch)

Finished fixing a CSS bug and moving to API auth? Run `/clear` (or open a new chat) immediately. Hauling three finished tasks into task four wastes tokens and breeds cross-task confusion.

Focused `/compact` (Seams)

Never wait for emergency auto-compaction at 95% capacity. Compact right after a task's test turns green, passing a steering focus: `/compact focus on T4 streaming; keep spec path & pending ACs`.

`HANDOFF.md` (Shift Relay)

Pausing mid-feature or switching from a research agent to a fresh builder? Write a 1-page `HANDOFF-<slug>.md` capturing *Verified Done*, *Settled Decisions*, *Landmines*, and *Next Action*.

What Survives Compaction? Externalize State to Disk First!

When a session compacts, chat reasoning, stack traces, and hook outputs get summarized away, and path-scoped rules detach until you touch those files again—while **root `AGENTS.md` / `CLAUDE.md` and disk files (`spec.md`, `plan.md`, `tasks.md`) remain 100% intact**. Always tick `[x]` in `tasks.md` and record tricky findings in `plan.md` *before* running `/compact`!

5 Real-World Copy-Paste Context & Spec Recipes

Notice how short and calm your prompts become once truth lives in repository files. Copy and adapt these five high-leverage prompt recipes in Claude Code, Cursor, Antigravity, Copilot, or Codex CLI.

Recipe & Goal	When to Run It	Copy-Paste Prompt / Action
1. Greenfield Spec → Plan Kickoff	Right after drafting `spec.md`, before writing any code	*"Read `docs/specs/<slug>/spec.md` and follow all `AGENTS.md` pointers. Draft `plan.md` (approach, file changes, edge cases, test map, rejected alternatives) and `tasks.md` (ordered tasks, each with a runnable `Verify:` command). Map every AC to ≥1 test. Write zero implementation code yet."*
2. Bloated `CLAUDE.md` Diet Refactor	Root context file exceeds 300 lines or feels ignored	*"Audit `CLAUDE.md` against live `package.json` and repo paths. Extract universal commands, 5 non-negotiables, and a conditional trigger table into a ≤150-line `AGENTS.md`; move subsystem guides into `docs/agent/*.md`; shrink `CLAUDE.md` to a thin `@AGENTS.md` adapter. List every deleted stale line."*
3. Cross-Vendor Adapter Setup	Mixed team using Claude, Cursor, Antigravity & Copilot	*"Set up One Canon, Thin Adapters: keep `AGENTS.md` as the single source of truth; create `CLAUDE.md` starting with `@AGENTS.md` + compact rules; create `.cursor/rules/00-core.mdc` (`alwaysApply: true`) pointing to `@AGENTS.md`; move React-only rules to `.cursor/rules/10-ui.mdc` with `globs:`."*
4. Legacy Bugfix Spec Gate	Tricky regression in unfamiliar or fragile legacy code	*"Do not patch code yet. Investigate the bug, then create `docs/specs/bug-<id>/spec.md` stating root cause, blast radius, and 3 GIVEN/WHEN/THEN ACs. Task T1 must add a failing reproduction test that exits non-zero today; Task T2 applies the minimal fix and proves full-suite green."*
5. Session Handoff & Clean Verifier	Ending a long session OR auditing finished feature code	**End of shift:** *"Write `HANDOFF-<slug>.md`: verified `[x]` tasks, settled decisions, landmines/failures hit, and exact next command."* **Fresh-session audit:** *"Verify code against `docs/specs/<slug>/spec.md`. Run every AC check verbatim, paste output tails, and report PASS/FAIL without editing code."*

Secret Weapon: Verification Without Authorship
Always run Recipe 5's **Clean Verifier** in a brand-new session! An implementation agent carries subtle confirmation bias toward shortcuts it took 40 turns ago; a fresh verifier has zero ego in the code and judges strictly against `spec.md`.

The Second-Correction Reflex
Whenever you catch yourself typing *"No, remember we use Decimal for money"* for the second time, pause 20 seconds: run `/learn` or add one bullet to `AGENTS.md` so no human on your team ever pays that tuition again.

Team Conventions, Monorepo Scopes & Git Hygiene

Context files are executable code: they run on a probabilistic LLM interpreter on every developer's laptop. Treating them with real software engineering discipline prevents three teammates' agents from pulling your codebase in three incompatible directions.

1. Git Hygiene: What to Commit vs. What to Gitignore

✓ Commit to Git (Shared Team Truth)

- Root `AGENTS.md` and subtree `packages/*/AGENTS.md`
- Thin adapters: `CLAUDE.md`, `.cursor/rules/*.mdc`, `.claude/rules/*.md`
- Shared playbooks & skills: `docs/agent/*.md`, `.agents/skills/`
- Feature work orders: `docs/specs/<YYYY-MM-slug>/{spec,plan,tasks}.md`

⊘ Gitignore (Personal / Ephemeral)

- `CLAUDE.local.md` & `.claude/settings.local.json`
- Personal voice/editor preferences (keep in `~/.claude` or `~/.gemini`)
- Auto-memory scratchpads (`~/.claude/projects/*/memory/`)
- Temporary `HANDOFF-*.md` relay notes (fold into spec changelog & delete on merge!)

2. Monorepo Conflict Policy & Ownership (Paste Into Root Canon)

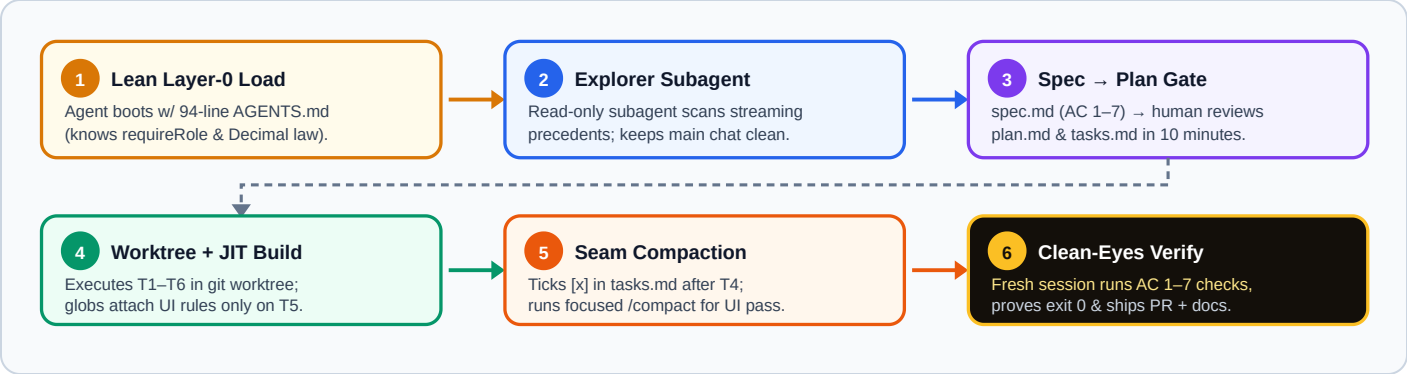
```
# CODEOWNERS — protect shared agent instructions like production code
/AGENTS.md          @acme/platform-leads
/docs/agent/        @acme/platform-leads
/.cursor/rules/     @acme/platform-leads

# Conflict Policy (include inside /AGENTS.md for deterministic tie-breaking)
If loaded instructions conflict: (1) direct human prompt in this session wins;
(2) else rule scoped closest to the edited file path wins (e.g. apps/web/AGENTS.md);
(3) else root /AGENTS.md wins. Always name conflicting sources in your reply.
```

3. Four Rituals That Eliminate Context Drift Forever

Governance Ritual	How High-Performing Engineering Teams Run It
1. Ride-Along PR Checkbox	Add one checkbox to `.github/pull_request_template.md`: *"If this PR renames scripts, moves paths, or changes conventions, I updated `AGENTS.md` / `docs/agent/*` in this same commit."*
2. `check-context.sh` in CI	Run a 20-line shell check on every PR verifying: (a) every backticked repo path in `AGENTS.md` exists on disk, (b) every cited `pnpm <script>` exists in `package.json`, and (c) root `AGENTS.md` stays `≤350` lines.
3. Point at Living Config	Never hardcode *"We use Node 20.11"* in prose (guaranteed future lie). Write *"Runtime versions are pinned in `.nvmrc` and `package.json`—read them directly."*
4. Quarterly Delete Audit	Every 90 days, check `<!-- Last verified -->` headers and delete rules that models now handle natively or linters enforce automatically. Deletion is maintenance!

End-to-End Synergy: One Feature, Zero Thrashing

See how every layer connects on a real feature—shipping a streaming `10,000`-row **Team Expense CSV Export** without security leaks, memory timeouts, or mid-session amnesia.



Turn-by-Turn Feature Execution Walkthrough

Session / Turn	Action & Context Loaded	What This Prevents
Turn 1: Align & Plan	Approve `spec.md`; run Recipe 1. Agent reads `AGENTS.md` trigger → pulls `docs/agent/export.md` → outputs `plan.md` + `tasks.md` (`T1..T7`).	Catches inline-SQL temptation & client-disconnect cursor leak during a 10-minute Markdown review instead of post-code surgery.
Turn 2: Backend Slice (`T1–T4`)	*"Execute T1–T4 in tasks.md."* Agent builds pure RFC-4180 serializer, cursor stream, `requireRole("admin")` route & 10k-row perf test.	First naive buffer attempt fails **AC-3** (`<150MB` memory check)—the agent catches its own mistake immediately and switches to streaming!
Turn 3: Seam + UI Slice (`T5–T6`)	Agent marks `[x] T1–T4` in `tasks.md`; you run `/compact focus on T5–T6 web UI`. Touching `TeamExpenses.tsx` auto-attaches `10-web.mdc`.	Backend test logs are compacted away cleanly while frontend TanStack Query & Playwright visibility rules attach right on cue.
Turn 4: Fresh Audit (`T7`)	Open a **brand-new session** (or subagent) and run Recipe 5's Clean Verifier against `spec.md` + `pnpm check && pnpm test`.	Produces an objective 7-for-7 PASS receipt ready to paste directly into your GitHub Pull Request description.

Subagent Isolation Multiplier: Keep Exploration Out of Implementation

When investigating an unfamiliar subsystem, delegate discovery to a read-only **Research Subagent** (in Antigravity, Claude Code, or Codex) that writes a 40-line summary to `docs/agent/notes/`. Its 50,000 tokens of grep dead-ends never pollute your builder agent's context window!

Troubleshooting Clinic & Pre-Flight Checklist

Keep this final page beside your editor. Almost every case of “the model got dumber today” traces back to one of seven structural context or specification bugs, each with a 10-second fix.

Top 7 Context & Spec Failure Modes (And 10-Second Structural Fixes)

Symptom You Observe	Root Cause & Immediate Structural Fix
1. Agent ignores a rule you definitely wrote down	Rule is buried inside a 600-line file (attention dilution) or contradicted by another file. Trim Layer 0 under 300 lines, inspect loaded files via `/context` or `/memory`, and move must-never-fail bans to `settings.json` / hooks.
2. Agent re-litigates settled choices after `/compact`	Decisions lived only in chat history, which gets summarized away. Record settled choices in `plan.md` or `HANDOFF.md` *before* compacting, and pass a focus instruction: `/compact focus on <task>`.
3. Agent edits `spec.md` to match its buggy code	Missing spec governance boundary. Add to `AGENTS.md`: “*`spec.md` is read-only after approval; never weaken ACs without human sign-off,*” and verify in a fresh session.
4. Agent runs dead commands (`pnpm lint` vs `check`)	**Stale Override** anti-pattern—obediently following an outdated line. Delete dead lines, add the Staleness Circuit-Breaker to `AGENTS.md`, and run `check-context.sh` in CI.
5. Teammate's agent behaves differently on same branch	Hidden collision with personal `~/.claude/CLAUDE.md`, `~/.gemini/GEMINI.md`, or Cursor User Rules. Keep global files strictly personal and promote team conventions into committed `AGENTS.md` via PR.
6. Sharp for 20 minutes, then sloppy & repetitive	Context accumulation (“lost in the middle”). Save Verified `[x]` state to `tasks.md`, run `/clear`, and launch a clean window with a 3-line pointer prompt to the next unchecked task.
7. Path-scoped rule never attaches when editing	Used `.md` instead of required `.mdc` inside `.cursor/rules/`, or glob typo in `globs:` / `paths:` YAML frontmatter. Fix frontmatter and verify with a fresh-session behavioral probe.

6-Point Pre-Flight Checklist: Before Kicking Off Any Serious Feature

☐ **1. One Lean Canon:** Root `AGENTS.md` is ≤300 lines with exact verified commands and thin tool adapters.

☐ **2. Trigger Pointers Ready:** Specialist runbooks live in `docs/agent/*.md` or `SKILL.md` behind explicit IF → THEN triggers.

☐ **3. Plan Reviewed First:** Reviewed `plan.md` & `tasks.md` before allowing multi-file code edits.

☐ **4. Runnable ACs:** Every requirement in `spec.md` names a CLI command or test file that exits `0`.

☐ **5. Clean Context Window:** Ran `/clear` from prior work; plan to `/compact` proactively at green task seams.

☐ **6. Fresh-Eyes Audit:** Scheduled a clean-session verifier pass + context ride-along check before opening PR!