

Copilot Rate-Limit Exit Prompt Kit

Start Here

This prompt kit is for developers who already pay for an AI coding assistant and need a calmer way to decide what to do after rate limits, billing changes, or confusing alternative pricing. It does not tell you that one vendor is universally best. It gives you reusable prompts that turn a messy switching decision into concrete notes: what work is blocked, which workflows actually need premium models, which IDE habits matter, what a short trial should prove, and what to say to a client or team before changing tools.

Use it in order if you are blocked this week. Use prompts 01-04 first to document the rate-limit event, recover the current task, and separate urgent work from long-term migration. Then use prompts 05-12 to compare alternatives and plan a 30-minute switch trial. Prompts 13-16 help you communicate the change and validate whether the replacement is actually better for your work.

This is an independent educational product. It is not affiliated with, endorsed by, or sponsored by GitHub, Microsoft, Copilot, Cursor, Windsurf, Continue.dev, Anthropic, Google, DeepSeek, Kimi, or any other vendor. It is not legal, financial, procurement, or employment advice.

The Prompt Kit Framework

1. Capture the incident before choosing a replacement. A rate-limit message feels urgent, but the first useful move is to document exactly what failed: date, plan, model or feature, IDE, task type, deadline impact, and reset message. This keeps the decision anchored in evidence instead of anger.
2. Separate recovery from migration. If a client ticket or release is blocked today, you need a recovery path first: downgrade model, switch task type, use local search, or move one narrow task to another tool. Migration is a second decision. The prompts keep those decisions separate so you do not rebuild your whole workflow while stressed.
3. Build a shortlist from constraints, not hype. The right alternative depends on IDE preference, privacy posture, setup tolerance, budget, model needs, repository size, and whether you need autocomplete, chat, agentic edits, or command-line workflows. The comparison prompts force each option through the same criteria.
4. Run a 30-minute trial with a real task. A pricing page does not prove fit. The kit asks you to choose one representative task, define pass/fail criteria, time the setup, capture friction, and compare output quality using your own repo.
5. Plan a reversible migration. Do not cancel a working subscription until you know how you will recover credentials, extensions, editor settings, client

expectations, and team instructions. The migration prompts produce a reversible checklist and a fallback date.

6. Validate after the switch. A replacement is only better if it reduces blocked work, cost surprise, or workflow friction. The post-switch prompts produce a one-week scorecard, not a vibe-based verdict.

Worked Scenario

Maya is a freelance full-stack developer on a fixed-fee dashboard rebuild. She pays for Copilot Pro+ because she likes staying inside VS Code and because most of her daily use is code chat, refactoring, and small test generation. On Thursday afternoon, she sees a weekly rate-limit message while trying to finish a client bugfix. Her first instinct is to cancel immediately and switch to whichever tool people are recommending on social media.

She starts with Prompt 01 and records the facts: the blocked task is a React table performance bug, the deadline is Friday morning, the affected workflow is chat-assisted debugging, and the reset message points to a future date. Prompt 02 converts the emergency into a recovery plan: use standard autocomplete for small edits, move the explanation request to a second assistant she already has access to, and reserve any premium model calls for final review. She decides not to change the client timeline yet because the immediate task still has a workaround.

Next, Maya runs Prompt 05 to create a shortlist. Her hard constraints are VS Code support, low setup friction, predictable monthly cost, no client code pasted into unknown web forms, and acceptable performance on TypeScript. The prompt narrows her candidates to one hosted IDE assistant, one BYOK extension, and one command-line assistant for deeper refactors. Prompt 06 turns that shortlist into a trial matrix: install time, repo indexing behavior, task success, privacy notes, and cancellation friction.

On Friday morning, she uses Prompt 09 for a 30-minute switch rehearsal. She picks a real but non-urgent task: add tests around a data transformation function. The hosted tool sets up fastest but edits too broadly. The BYOK extension takes longer to configure, yet keeps changes smaller and gives her better control. The command-line assistant is powerful, but she decides it is better for weekend maintenance than client-day hotfixes.

Before changing anything permanently, Maya runs Prompt 13 to draft a client-safe note. She does not mention vendor drama or make promises about savings. She says she is adjusting her AI coding workflow to reduce interruption risk and will keep code review standards unchanged. After one week, Prompt 16 has her compare blocked hours, cost exposure, setup friction, and code quality. Her result is not a universal recommendation: keep Copilot through the month for autocomplete, shift deeper chat tasks to the BYOK extension, and revisit cancellation after the June billing transition once she has real usage data.

Common Mistakes

- Cancelling in the middle of a blocked delivery before documenting what failed.
- Comparing headline monthly prices while ignoring usage caps, paid overage, model limits, setup time, and client privacy requirements.
- Switching every workflow at once instead of testing one representative task.
- Letting social posts choose the tool instead of your IDE, repo, task mix, and budget.
- Using a new assistant on sensitive client code before checking data handling settings and contract obligations.
- Telling a client the new setup will save money or improve speed before collecting real usage data.
- Treating BYOK as free; API usage, context size, retries, and failed experiments still cost money.

Quick FAQ

Is this a recommendation to leave Copilot? No. It helps you decide whether to stay, adjust usage, or test alternatives based on your own work.

Does it include legal or refund advice? No. It deliberately avoids lawsuit, refund-rights, harassment, or complaint escalation instructions.

Can I use it with any coding assistant? Yes. The prompts work with Copilot, editor assistants, command-line coding tools, BYOK extensions, and general AI chat tools.

How long should the first pass take? A focused emergency pass takes about 30-45 minutes. A full comparison and one-week validation takes longer.

What if my team controls tool choice? Use the team communication and scorecard prompts to create a neutral decision packet instead of changing tools unilaterally.

Does this promise lower cost? No. It gives you a structured way to estimate and monitor cost before committing.

Template Walkthrough

Use templates/switch-trial-scorecard.csv after Prompt 09. Before the prompt, your trial note might be vague: Windsurf seemed good, Continue was annoying, maybe cancel Copilot. After the prompt, the row becomes decision-grade: candidate, task, setup minutes, task outcome, friction, privacy note, estimated monthly cost, and keep/test/reject decision. A filled sample row is included so you can copy the scoring style without inventing the format.

Source Currentness

As of May 14, 2026, GitHub documentation states that individual Copilot plans are moving from request-based billing to usage-based billing on June 1, 2026. GitHub's April 27 billing announcement describes the shift from premium-request counting to GitHub AI Credits, and GitHub's May 12 individual-plan update says Pro and Pro+ will receive flex allotments and a new Max plan will be added for sustained high-volume use. GitHub's usage monitoring documentation also says Copilot Pro, Pro+, and student plans have tighter session and weekly limits from April 20, 2026. Community reports in the provided Scout signal describe Pro+ users hitting weekly limits and evaluating alternatives. Treat those community reports as user reports, not verified vendor admissions.

Copilot Rate-Limit Exit Prompt Kit - Prompt Library

Copy each prompt into your preferred assistant and fill in the bullets. Keep vendor names factual, avoid unsupported claims, and use the outputs as decision notes. These prompts are designed for developers who need a reversible stay/switch/hybrid decision, not a vendor argument.

Prompt 01: Rate-limit incident recorder

Category: Current usage audit

Use when: You hit a rate limit, usage warning, or billing surprise and need to turn the event into evidence.

Inputs to gather: exact message shown, plan name, date/time, IDE, feature used, task type, repo sensitivity, deadline impact, screenshots or billing-page notes.

Prompt:

Act as a careful AI coding workflow analyst. I am documenting a Copilot rate-limit, usage, o

Incident facts:

- Tool and plan:
- Date/time and timezone:
- Exact message or behavior:
- Feature affected: chat / autocomplete / agent / code review / other:
- IDE and OS:
- Repository/task affected:
- Deadline or client/team impact:
- Workaround already tried:
- Evidence I can verify:

Produce an incident note with:

1. A one-paragraph factual summary using only the facts above.
2. A table separating verified facts, assumptions, and missing evidence.
3. A severity rating: low / medium / high, with the reason.
4. The next 3 reversible actions for the next 30 minutes.
5. Claims I should avoid making publicly or to clients until I have better evidence.

Worked sample: A freelancer records that chat was limited during a deadline, while autocomplete still worked. The output separates “chat blocked on Thursday” from unsupported claims about all Copilot features being unusable.

Prompt 02: Blocked-work recovery planner

Category: Current usage audit

Use when: A delivery task is blocked today and you need a recovery path before any migration decision.

Inputs to gather: task deadline, affected workflow, available tools, sensitive files, manual fallback, acceptable scope reduction, teammate/client expectations.

Prompt:

Act as a delivery-focused engineering lead. Help me recover from a coding-assistant interruption.

Current delivery:

- Task:
- Deadline:
- What the assistant was helping with:
- What still works:
- What is blocked:
- Tools I already have access to:
- Files or data I must not paste into third-party tools:
- Manual fallback I can execute:

Create a recovery plan with:

1. A "finish the current task" path using the lowest-risk available workflow.
2. A "defer until reset/billing clarity" path for non-urgent work.
3. A "test one alternative safely" path using non-sensitive code.
4. A client/team update sentence only if timeline risk is real.
5. A stop condition that tells me when to pause instead of switching tools under stress.

Worked sample: A React hotfix is due tomorrow. The prompt recommends finishing with autocomplete plus manual debugging, testing an alternative on a toy branch later, and avoiding cancellation until after delivery.

Prompt 03: Premium request usage map

Category: Rate-limit triage

Use when: You need to identify which workflows consume scarce premium usage and which can be moved to lower-cost modes.

Inputs to gather: last 7 days of tasks, model/features used, frequency, value per task, alternatives, whether the task needs premium reasoning.

Prompt:

Act as an AI coding usage auditor. Map my weekly assistant usage into premium-critical and p

Usage notes:

- Plan and billing model I am currently on:
- Tasks I used the assistant for this week:
- Features/models used for each task:
- Tasks that were deadline-critical:
- Tasks that could use autocomplete, search, or a cheaper model:
- Known usage counters or reset windows:

Return:

1. A table with each workflow, frequency, business value, urgency, and likely premium dependence.
2. A "protect premium usage" list for tasks that deserve scarce usage.
3. A "move away from premium" list for tasks that can use cheaper or manual workflows.
4. A monitoring routine I can run every Friday.
5. One sentence explaining the decision in neutral, evidence-based language.

Worked sample: The output identifies architecture review and tricky debugging as premium-critical, while README edits, simple test generation, and syntax cleanup can move to cheaper workflows.

Prompt 04: Model downgrade decision helper

Category: Rate-limit triage

Use when: You are considering a cheaper model or feature mode but do not know what quality risk it creates.

Inputs to gather: task category, acceptable error risk, test coverage, review process, candidate lower-cost modes, examples of past failures.

Prompt:

Act as a senior code reviewer evaluating whether a lower-cost assistant mode is acceptable f

Task:

- Codebase/language:
- Requested assistant job:
- Current high-capability mode:
- Candidate lower-cost mode:
- Test coverage available:
- Human review available:

- Consequence of a bad suggestion:

Give me:

1. A downgrade suitability score from 1-10.
2. The specific failure modes to watch for.
3. A safe prompt I can use with the lower-cost mode.
4. A verification checklist before merging any output.
5. A recommendation: use lower-cost mode, use premium mode, or do manually.

Worked sample: The prompt says doc updates and low-risk unit tests can be downgraded, but auth logic and migrations should stay in stronger review lanes.

Prompt 05: Alternative shortlist builder

Category: Alternative shortlisting

Use when: You need a short list of viable options based on your constraints, not social-media rankings.

Inputs to gather: IDE, OS, languages, repo size, privacy constraints, budget ceiling, setup tolerance, must-have features, current assistant pain.

Prompt:

Act as a neutral AI coding tool evaluator. Build a shortlist from my constraints. Do not de

Constraints:

- IDE/OS:
- Languages/frameworks:
- Repo size and sensitivity:
- Must-have features:
- Nice-to-have features:
- Monthly budget ceiling:
- Setup tolerance: low / medium / high:
- Tools I refuse to use:
- Tools I already pay for:

Produce:

1. A shortlist of 3 option types: stay/adjust current setup, hosted alternative, BYOK or lo
2. A fit score for each option type with assumptions stated.
3. Questions I must answer before choosing a specific vendor.
4. A one-task trial plan for each option.
5. A recommendation for which option to test first and why.

Worked sample: A VS Code freelancer gets a shortlist that keeps Copilot for autocomplete, tests one BYOK extension on a non-sensitive branch, and reserves command-line agents for weekend refactors.

Prompt 06: Vendor-fit comparison matrix

Category: Alternative shortlisting

Use when: You have candidate tools and need a side-by-side decision matrix.

Inputs to gather: candidate names, prices, model access, IDE support, data handling, setup time, billing model, cancellation friction, known limits.

Prompt:

Act as a procurement-minded developer. Compare these AI coding assistant candidates with ca

Candidates:

- Candidate A:
- Candidate B:
- Candidate C:

My criteria:

- Required IDE:
- Required language support:
- Data/privacy constraints:
- Monthly budget:
- Billing predictability needs:
- Team/client requirements:
- Setup time limit:

Create:

1. A comparison matrix with price model, setup friction, IDE fit, data-handling questions, u
2. A list of claims that need source verification before purchase.
3. A "best for" and "avoid if" line for each candidate.
4. The single trial task I should run across all candidates.
5. A decision rule for stay / switch / hybrid.

Worked sample: The matrix flags that cheap-looking options can become expensive with API usage, while hosted tools may win on setup speed but lose on data-control questions.

Prompt 07: BYOK readiness checker

Category: BYOK setup planning

Use when: You are considering bring-your-own-key tools and need to know if you are ready for token/API cost management.

Inputs to gather: API providers, key-management practice, budget alerts, repo sensitivity, model preferences, logging requirements, expected usage.

Prompt:

Act as a practical BYOK setup reviewer for a solo or small-team developer.

Setup facts:

- Candidate BYOK tool:
- API provider(s):
- Monthly API budget:
- Where keys will be stored:
- Who can access keys:
- Repos/tasks to test:
- Data logging or retention concerns:
- Existing budget alerts:

Return:

1. A BYOK readiness score from 1-10.
2. Required guardrails before using real client code.
3. Budget-alert settings I should configure.
4. A first safe test that uses non-sensitive code.
5. A decision: proceed, postpone, or use a hosted alternative.

Worked sample: The prompt recommends setting provider budget alerts and testing on a sample repo before putting client code through a new extension.

Prompt 08: Local/client-code risk reviewer

Category: BYOK setup planning

Use when: You need to decide whether a candidate assistant workflow is acceptable for sensitive or client-owned code.

Inputs to gather: client contract constraints, repo sensitivity, data sent to vendor, logs, retention settings, approval process, alternative local/manual path.

Prompt:

Act as a cautious engineering risk reviewer. I need a non-legal, operational review of whether

Context:

- Client/team constraints I know:
- Repository sensitivity:
- Data the assistant may receive:
- Tool/vendor:
- Logging/retention settings I can verify:
- Human review process:
- Local/manual alternative:

Produce:

1. A risk register with risk, evidence, mitigation, and owner.
2. Questions to ask the client/team before use, written neutrally.

3. A safer test plan using synthetic or non-sensitive code.
4. Red flags that should stop the trial.
5. A short decision memo that avoids legal conclusions.

Worked sample: For a healthcare-adjacent client repo, the prompt recommends synthetic tests first and a client approval question before any code is pasted into a new service.

Prompt 09: 30-minute IDE switch rehearsal

Category: IDE migration

Use when: You want a fast trial that tests real workflow fit without committing to a new tool.

Inputs to gather: candidate tool, representative task, branch name, setup timer, pass/fail criteria, rollback steps, notes template.

Prompt:

Act as a migration coach. Design a 30-minute coding assistant switch rehearsal that is reversible.

Trial facts:

- Current IDE/tool:
- Candidate tool:
- Test repository/task:
- Files allowed in the trial:
- Success criteria:
- Hard stop time:
- Rollback steps:
- Metrics to capture:

Give me:

1. A minute-by-minute trial script for 30 minutes.
2. Setup checks before touching the repo.
3. The exact task prompt to give the candidate assistant.
4. A scorecard with setup friction, edit quality, privacy confidence, speed, and rollback ease.
5. A final decision rule: keep testing, reject, or use only for limited tasks.

Worked sample: The trial uses a non-critical test-generation task and rejects a tool that edits broadly without explaining changes.

Prompt 10: Editor settings migration checklist

Category: IDE migration

Use when: You are moving between IDE extensions or assistants and need to preserve settings, shortcuts, and team norms.

Inputs to gather: current extensions, settings sync, keybindings, ignored files, telemetry preferences, lint/test commands, rollback plan.

Prompt:

Act as a developer-experience migration planner. Help me move assistant settings without bre

Current setup:

- IDE:
- Current assistant extension(s):
- Candidate extension/tool:
- Important keybindings:
- Workspace settings:
- Files/folders assistant should ignore:
- Test/lint commands:
- Rollback plan:

Create:

1. A pre-migration backup checklist.
2. A settings mapping table: current setting, candidate equivalent, unknowns.
3. A privacy/telemetry settings checklist to verify.
4. A rollback checklist if the candidate tool is noisy or slow.
5. A teammate note explaining the change without vendor drama.

Worked sample: The prompt catches that an extension indexes the whole workspace by default and adds an ignore-list check before the trial.

Prompt 11: True monthly cost estimator

Category: Cost comparison

Use when: You need to estimate total monthly cost across subscription, AI credits, API usage, Actions minutes, and time spent babysitting tools.

Inputs to gather: subscription prices, expected usage, API pricing pages, current plan credits, private repo code review usage, setup hours, failed-run rate.

Prompt:

Act as a conservative cost estimator for AI coding workflows. Use ranges when exact pricing

Known costs:

- Current subscription:
- Candidate subscription(s):
- Included credits/requests:
- Expected API usage:
- Code review or CI minutes affected:
- Setup/maintenance hours:
- Budget ceiling:

Return:

1. A monthly cost table with low/base/high scenarios.
2. The assumptions that drive the high scenario.
3. Usage patterns that could create surprise spend.
4. Monitoring steps for the first 14 days.
5. A recommendation: stay, switch, hybrid, or wait for more billing data.

Worked sample: The estimate shows that a cheaper BYOK setup can exceed the current plan if large-context retries and agent loops are not capped.

Prompt 12: Keep-adjust-replace decision prompt

Category: Cost comparison

Use when: You need a final choice after testing, but you want to avoid an emotional all-or-nothing switch.

Inputs to gather: incident notes, trial results, cost estimates, privacy risks, productivity evidence, team/client constraints, deadline calendar.

Prompt:

Act as a calm decision facilitator. Help me choose between keeping, adjusting, replacing, or

Evidence:

- Incident summary:
- Current plan/tool value:
- Trial results:
- Cost estimate:
- Privacy/data concerns:
- Deadlines in the next 30 days:
- Team/client constraints:
- Cancellation or renewal date:

Produce:

1. A decision table for keep / adjust / replace / hybrid.
2. The strongest evidence for and against each path.
3. A recommended path for the next 14 days.
4. A fallback if the recommended path fails.
5. A one-sentence decision I can save in my notes.

Worked sample: The output recommends a hybrid path: keep the current tool for autocomplete through a deadline and test a BYOK workflow for deeper chat on non-sensitive tasks.

Prompt 13: Client-safe workflow update note

Category: Team/client communication

Use when: A client or stakeholder may notice a process change and you need a professional, low-drama explanation.

Inputs to gather: project status, whether timeline changes, quality controls, tool-change reason, what will not change, review process.

Prompt:

Act as a technical account manager. Draft a client-safe note about an internal AI coding wor

Facts:

- Project:
- Whether timeline changes:
- What workflow is changing:
- Why it is changing, stated neutrally:
- What quality controls stay in place:
- What the client needs to do:
- Claims to avoid:

Write:

1. A 75-word client note.
2. A shorter Slack-style version.
3. A version for "no timeline impact."
4. A version for "minor timeline risk."
5. A list of phrases to avoid because they sound like excuses or vendor blame.

Worked sample: The note says the developer is reducing workflow interruption risk while keeping tests and code review unchanged, without mentioning social-media drama.

Prompt 14: Team trial proposal prompt

Category: Team/client communication

Use when: Tool choice is team-controlled and you need approval for a structured trial.

Inputs to gather: team policy, candidate tool, data boundaries, repo to test, budget cap, success metrics, trial length, reviewer.

Prompt:

Act as an engineering manager drafting a lightweight internal trial proposal.

Trial proposal facts:

- Current approved tool:
- Candidate workflow:
- Reason for trial:
- Data boundaries:
- Test repository/task:

- Trial duration:
- Budget cap:
- Success metrics:
- Reviewer/approver:

Create:

1. A one-page trial proposal.
2. A risk and mitigation table.
3. A budget and usage cap section.
4. A success/fail decision rule.
5. A message asking for approval without pressuring the team.

Worked sample: The proposal asks to test a BYOK extension on an internal sample repo for one week, with a hard API budget cap and no client code.

Prompt 15: One-week replacement scorecard

Category: Post-switch validation

Use when: You tested a replacement or hybrid setup and need evidence instead of impressions.

Inputs to gather: one week of tasks, blocked hours, setup time, assistant errors, monthly spend, output quality notes, rollback incidents.

Prompt:

Act as an evidence-based workflow reviewer. Evaluate one week of my AI coding assistant trial.

One-week data:

- Candidate workflow:
- Tasks completed:
- Blocked hours:
- Setup/maintenance time:
- Output quality notes:
- Failed or risky suggestions:
- Approximate spend:
- Privacy or client concerns:
- Times I returned to the old tool:

Return:

1. A scorecard with speed, quality, predictability, privacy confidence, cost control, and de
2. The best evidence that the trial helped.
3. The best evidence that it hurt.
4. A recommendation for the next 30 days.
5. Metrics to keep tracking if I continue.

Worked sample: The scorecard shows fewer blocked chat sessions but more setup friction, so the recommendation is a hybrid workflow rather than a full

replacement.

Prompt 16: Final stay-switch-hybrid memo

Category: Post-switch validation

Use when: You need a final memo for yourself, a client, or a team before changing subscriptions or standard workflow.

Inputs to gather: incident record, usage map, trial scorecard, cost estimate, data-handling decision, deadlines, approval constraints.

Prompt:

Act as a senior engineering advisor. Write a final stay/switch/hybrid memo from the evidence

Evidence:

- Original incident:
- Current tool value:
- Premium usage map:
- Alternative trial result:
- Cost estimate:
- Data/privacy decision:
- Upcoming deadlines:
- Team/client approval constraints:

Write:

1. An executive summary in 5 bullets.
2. The chosen path: stay / adjust / switch / hybrid.
3. Why this path is reversible.
4. The next 7 days of actions.
5. The revisit date and metrics that would change the decision.

Worked sample: The memo keeps Copilot for current client work, moves exploratory refactors to a capped BYOK setup, and sets a revisit date after real usage-based billing data is visible.

Source Currentness Notes

Checked May 14, 2026.

Official GitHub sources used for customer-facing factual claims:

- <https://docs.github.com/en/copilot/concepts/copilot-billing/about-billing-for-individual-copilot-plans>
- <https://docs.github.com/en/copilot/get-started/plans>
- <https://docs.github.com/en/copilot/how-tos/manage-and-track-spending/monitor-premium-requests>

- <https://docs.github.com/en/copilot/how-tos/manage-and-track-spending/prepare-for-your-move-to-usage-based-billing>
- <https://github.blog/news-insights/company-news/github-copilot-is-moving-to-usage-based-billing/>
- <https://github.blog/news-insights/company-news/github-copilot-individual-plans-introducing-flex-allotments-in-pro-and-pro-and-a-new-max-plan/>

Scout signal sources used as demand evidence, treated as community/user reports rather than vendor-confirmed facts:

- `/home/agentops/.openclaw/workspaces/scout-cloud/signals/2026-05-14_copilot-proplus-weekly-rate-lockout-39-mo-unusable-byok-exodus-buyer-intent.json`
- `/home/agentops/.openclaw/workspaces/scout-cloud/signals/2026-05-14_ai-coding-tool-pricing-comparison-shopping-explosion-buyer-intent.json`

Adrian May 14 first-pass currentness note: the May 12 GitHub individual-plan update adds Pro/Pro+ flex allotments and a Max plan starting with the June 1 usage-based billing transition. Customer-facing copy should not imply the April 27 AI Credits announcement is the last official plan update.

Language rules applied: no guaranteed savings, no guaranteed productivity, no universal best alternative, no affiliation or endorsement claims, no legal or refund-rights advice.

Copilot Rate-Limit Exit Prompt Kit

This ZIP contains:

- `copilot-rate-limit-exit-prompt-kit.pdf` - polished reading copy
- `product-guide.md` - start-here notes and framework
- `prompt-library.md` - 16 copy-paste prompts
- `source-notes.md` - currentness notes and source URLs
- `templates/prompt-index.csv`
- `templates/switch-trial-scorecard.csv`
- `templates/cost-comparison-worksheet.csv`
- `templates/client-team-communication.md`

Start with Prompt 01 if you are blocked today. Start with Prompt 05 if you are comparison-shopping before June 1, 2026. This is an independent educational prompt kit, not legal, financial, procurement, or employment advice.

Support: hello@withjz.com

Refund policy: 14 days for accidental duplicate purchase or unusable file delivery.