FREE SAMPLE

Top 10 Operator Prompts

Ten complete prompts from the Operator Prompt & Workflow Vault, each with its inputs, output format and a worked example.

WHAT'S INSIDE

- 10 of the 40 prompts in the full vault
- Research, writing, decisions, agents, operations and evaluation
- Full prompt text, ready to copy
- A worked example with input and output for each
- Plain Markdown placeholders you replace with your material

EDITION: v2.0 (2026) | PUBLISHER: ZYTAN.AI https://zytan.ai

CONTENTS

Table of contents

These ten prompts are reproduced from the full vault. Each page keeps the parts you need to run the prompt: when to use it, the inputs, the prompt itself, the output format and a worked example. The full versions also include when not to use each prompt, variations, chaining notes and failure modes.

How to use these prompts 3

R01 · Source-Graded Research Brief 4

R05 · Claim-by-Claim Verification 7

W02 · BLUF Executive Summary 10

W05 · Anti-Slop Detection and Rewrite 12

A01 · Reversibility-Aware Decision Memo 15

A02 · Pre-Mortem Before You Commit 18

G02 · Scoped Agent System Prompt Writer 21

G04 · Verify-Before-Returning Loop 24

O04 · Blameless Postmortem from a Timeline 27

E01 · Rubric Builder for Any Output 30

What's in the full vault 33

START HERE

How to use these prompts

1. Pick the prompt that matches the job, not the one that sounds most capable.
2. Fill every input. The Inputs table says what to provide and shows an example of each.
3. Paste the prompt into any capable chat model. Prompts that need a long context or careful reasoning say so in their header.
4. Compare your result with the worked example. If yours is vaguer, the inputs were usually too thin.

> **NOTE About the examples**
>
> All companies, people and numbers in the worked examples are fictional. Arithmetic in example outputs is worked from the example inputs so you can check it.

RESEARCH & SYNTHESIS · R01

R01 · Source-Graded Research Brief

Most research prompts produce a confident essay. This one produces a brief you can defend in a meeting: each finding points to the source it came from, says how strong that source is, and flags where sources disagree. It is built for the moment before a decision, when you have ten tabs open and need to know what is actually established.

Best run with: frontier reasoning model with a long context window

When to use it

- You have collected sources (articles, docs, notes, transcripts) and need a decision-ready summary.
- You need to show your work to someone skeptical: a manager, a client, an investor.
- The topic has conflicting claims and you want the disagreement surfaced, not averaged away.

Inputs

Input	What to provide	Example
`research_question`	One question, phrased so it could be answered yes, no, or with a number	"Should we offer annual billing at launch?"
`decision_it_informs`	What you will do differently depending on the answer	"Whether to build annual plans before or after launch"
`sources`	Pasted text, each with a short label (S1, S2...) and where it came from	S1: billing provider's docs page on subscriptions...
`deadline_context`	How much certainty you need and by when	"Decision on Thursday; directional is fine"

The prompt

PROMPT

```
I need a research brief that answers one question and shows exactly how well
supported each part of the answer is.

QUESTION: {{research_question}}
THE DECISION THIS INFORMS: {{decision_it_informs}}
CERTAINTY NEEDED: {{deadline_context}}

SOURCES (each starts with its label):
{{sources}}

Work in this order:
1. Read every source. For each, note in one line: who produced it, what kind of
   evidence it is (primary data, official documentation, expert opinion,
   anecdote, marketing), and how recent it is if stated.
2. Extract every claim that bears on the question. Ignore claims that do not.
3. Group claims into findings. A finding can rest on one source or several.
```

PROMPT (continued)

```
4. Grade each finding:
   - A: two or more independent primary or official sources agree
   - B: one primary or official source, or several secondary sources agree
   - C: a single secondary source, opinion, or anecdote
   - X: sources conflict; do not pick a side, describe the conflict
5. Write the brief.

Rules:
- Use only the sources above. If you know something relevant that is not in
  them, put it under "Outside knowledge to verify", clearly separated.
- Cite with labels in brackets, like [S2] or [S1, S4]. No citation, no claim.
- Quote numbers exactly as the source states them, including units and dates.
- If the sources cannot answer the question, say so in the first line.

OUTPUT FORMAT
## Answer
Two to four sentences. Lead with the answer, then the main reason, then the
biggest caveat.

## Findings
| # | Finding | Grade | Sources | Why this grade |

## Where sources disagree
One short paragraph per conflict: what each side says, and what would settle it.

## Gaps
Bullet list of what we would need to know to raise confidence, each with a
suggested way to find out.

## Outside knowledge to verify
Only if needed. Mark each line "unverified".

## Source notes
| Label | Producer | Evidence type | Date | Usefulness (high/med/low) |
```

Output format

An answer-first Markdown brief. The findings table is the core artifact: you can paste it into a doc or slide and every row survives scrutiny because it carries its own sources and grade. The "Where sources disagree" section is mandatory whenever an X grade appears.

Worked example

WORKED EXAMPLE

INPUT · QUESTION AND SOURCES (ABRIDGED)

QUESTION: Should a two-person B2B SaaS team offer annual billing at launch? **DECISION:** Build annual plans now, or ship monthly-only and add annual in 60 days. **CERTAINTY NEEDED:** Directional; decision Thursday.

S1 (billing provider documentation): explains that annual and monthly prices are separate price objects on the same product, and that switching a customer between them can be prorated. **S2** (founder blog post, 2025): "We added annual at launch and 30% of our first 50 customers chose it. Cash up front paid for our first contractor." **S3** (SaaS pricing newsletter): argues annual plans at launch lock early customers into a price you will want to raise, and recommends waiting until pricing is stable. **S4** (our own customer interview notes, 6 calls): two buyers said procurement requires annual invoices; four did not mention billing.

OUTPUT · BRIEF (EXCERPT)

Answer. Offer annual billing at launch only if you can invoice it manually; do not build self-serve annual plans yet. Two of six interviewed buyers need annual invoices for procurement [S4], and the billing setup is a small amount of configuration [S1]. The main caveat is price lock-in: early annual customers keep launch pricing for a year [S3].

#	Finding	Grade	Sources	Why this grade
1	Some target buyers require annual invoicing to buy at all	B	S4	Primary data, but 2 of 6 calls is a small sample
2	Adding an annual price is configuration, not a rebuild	B	S1	Official documentation; one source
3	Early annual plans can generate meaningful upfront cash	C	S2	Single anecdote; the 30% figure is one company's result
4	Annual at launch locks in prices you may want to raise	X	S3 vs S2	S3 argues the risk; S2 reports no regret. Neither has data

Where sources disagree. S3 says wait; S2 says it worked. S2 is one company with unknown pricing changes after launch. Settling this needs our own price-confidence check: if we expect to raise prices within 12 months, S3's risk applies to us.

Gaps. How many target buyers have procurement rules like the two in S4? Ask in the next five calls. Do we expect a price change within a year? Decide before Thursday.

RESEARCH & SYNTHESIS · R05

R05 · Claim-by-Claim Verification

Use this before anything with your name on it goes out: a report, a blog post, a sales deck, a summary the model wrote for you. It does not judge whether the writing is good. It checks whether each factual statement is supported by the sources you trust, and it hands you a corrected sentence for every one that is not.

Best run with: frontier reasoning model

When to use it

- On any model-generated draft that contains facts, numbers, dates, names, or quotes.
- On human drafts before publishing, especially if they cite research.
- As the last step of a research chain (R01, then R05).

Inputs

Input	What to provide	Example
`draft_text`	The text to check	A 600-word blog post
`source_corpus`	The sources it should be checked against, labeled	"S1: internal metrics export; S2: vendor documentation..."
`strictness`	STRICT (unsupported means wrong) or LENIENT (unsupported but plausible gets flagged, not rewritten)	"STRICT for numbers, LENIENT for general statements"

The prompt

PROMPT

```
Fact-check the draft below against the sources. Do not evaluate style.

STRICTNESS: {{strictness}}

DRAFT:
<<<
{{draft_text}}
>>>

SOURCES:
{{source_corpus}}

Procedure:
1. Split the draft into atomic claims: one checkable fact per claim. A
   sentence with a number, a date, and a name contains three claims.
   Number them C1, C2...
2. Skip opinions, recommendations, and obvious framing. List what you skipped
   in one line so I can disagree.
3. For each claim, search the sources and assign one verdict:
```

PROMPT (continued)

```
   SUPPORTED: a source says the same thing (quote it)
   PARTIAL: a source says something close but different in scope, number,
   date, or certainty (quote it and name the difference)
   CONTRADICTED: a source says otherwise (quote it)
   UNSUPPORTED: no source addresses it
4. For every verdict except SUPPORTED, write a replacement sentence that says
   only what the sources support. Under STRICT, unsupported numbers are
   removed or replaced. Under LENIENT, unsupported general statements are
   kept and marked for a human.
5. Produce the corrected draft. Wrap every change in **double asterisks**
   so I can see it.

Return:
## Verification table
| # | Claim | Verdict | Evidence (quote + label) | Replacement |
## Corrected draft
## Needs a source
Claims a human should find a source for before publishing.
## Skipped as non-factual
```

Output format

The table is the audit trail; the corrected draft is the deliverable. Marked changes make review fast: you read only the bold parts.

Worked example

WORKED EXAMPLE

INPUT · DRAFT AND SOURCES (ABRIDGED)

STRICTNESS: STRICT for numbers and dates; LENIENT otherwise. **DRAFT:** "Since launching self-serve onboarding in March, activation climbed to 62% and support tickets fell by half. Customers now reach their first report in under five minutes, and the change was our most requested feature of the year." **S1** (product analytics export): "Activation rate: Feb 41%, Apr 58%, Jun 61%. Self-serve onboarding released April 3." **S2** (support dashboard): "Tickets tagged onboarding: Q1 212, Q2 131." **S3** (feature request board): "Top requests this year: 1. CSV export (88 votes), 2. Self-serve onboarding (64 votes)."

OUTPUT · VERIFICATION (EXCERPT)

#	Claim	Verdict	Evidence	Replacement
C1	Launched in March	CONTRADICTED	"released April 3" [S1]	Launched in early April
C2	Activation reached 62%	PARTIAL	"Jun 61%" [S1]	Activation reached 61% by June

WORKED EXAMPLE (continued)

OUTPUT · VERIFICATION (EXCERPT) (continued)

#	Claim	Verdict	Evidence	Replacement
C3	Support tickets fell by half	PARTIAL	Onboarding tickets 212 to 131 [S2], a 38% drop, and only onboarding-tagged	Onboarding-related tickets fell by 38%
C4	First report in under five minutes	UNSUPPORTED	none	Removed (STRICT: number)
C5	Most requested feature of the year	CONTRADICTED	Ranked second, 64 votes vs 88 [S3]	One of our two most requested features

Corrected draft. "Since launching self-serve onboarding in **early April**, activation **reached 61% by June** and **onboarding-related support tickets fell by 38%**. **It was one of our two most requested features this year.**"

WRITING & EDITING · W02

W02 · BLUF Executive Summary

Executives and clients read the first three lines. BLUF (bottom line up front) is the military and consulting habit of putting the conclusion and the ask first, then only the support needed to trust it. This prompt is tuned to a specific reader and forces it to say what it cut, so nothing important disappears without you knowing.

Best run with: any capable chat model with enough context for the source

When to use it

- A long report, research brief, or thread needs to reach someone senior.
- You need a status update that results in a decision, not a "thanks for the update."
- Turning meeting transcripts or analysis into an email.

Inputs

Input	What to provide	Example
`source_document`	The long text	A 12-page vendor evaluation
`reader`	Name or role, and how they read	"CFO; reads on phone between meetings"
`reader_cares_about`	Their priorities	"Cash impact this year, risk of switching"
`action_needed`	What you need from them, if anything	"Approve the contract by the 15th"

The prompt

PROMPT

```
Summarize the document for one specific reader who will give it about two
minutes.

READER: {{reader}}
WHAT THEY CARE ABOUT: {{reader_cares_about}}
WHAT I NEED FROM THEM: {{action_needed}}

DOCUMENT:
{{source_document}}

Write in this shape:

SUBJECT: under ten words, contains the conclusion, not the topic.
  ("Switch to Vendor B; saves $40K this year" not "Vendor evaluation update")

BOTTOM LINE: two or three sentences. The conclusion, the single most
important reason, and the ask with a date if there is one.

WHY (3 to 5 bullets): only facts this reader needs to trust the bottom line,
ordered by what they care about. Each bullet starts with the fact, not with
a lead-in like "It is worth noting".

RISKS OR OPEN QUESTIONS: one or two bullets, only if they could change the
reader's decision.
```

PROMPT (continued)

```
THE ASK: one line, phrased so they can answer yes or no.

Under a divider, for me only:
LEFT OUT: bullet list of material points from the document you did not
include, so I can check nothing critical was dropped.

Rules: use numbers from the document exactly. No adjectives that the
document does not support. Under 200 words above the divider.
```

Output format

An email-ready summary above a divider and an author-only "left out" list below it. The subject line rule alone improves most status emails.

Worked example

WORKED EXAMPLE

INPUT · DOCUMENT (ABRIDGED)

READER: Head of Operations; reads on phone. **CARES ABOUT:** Fulfillment speed, staff overtime, cost. **ASK:** Approve a two-week pilot of the new pick-path software starting May 6. **DOCUMENT:** A 9-page analysis of warehouse picking. Current average pick time: 4.1 minutes per order. Vendor trial on one aisle for 5 days cut it to 3.2 minutes. Overtime last quarter: 310 hours. Software cost: $1,800/month, no setup fee, cancel any time. Risk: handheld scanners on aisles 7 to 9 are an older model that the vendor has not tested. Also covers the vendor's company history, three alternative vendors rejected on price, and staff survey comments (mostly positive about the trial).

OUTPUT · SUMMARY

SUBJECT: Pilot new pick-path software: trial cut pick time 22%

BOTTOM LINE: The five-day single-aisle trial reduced average pick time from 4.1 to 3.2 minutes per order. I recommend a two-week pilot across the warehouse starting May 6. It costs $1,800 for the month with no setup fee and can be canceled at any time.

WHY

- Faster picks: 0.9 minutes saved per order in the trial.
- Overtime pressure: 310 overtime hours last quarter; faster picking is the most direct way to reduce them.
- Low commitment: monthly billing, no setup fee, cancel any time.

RISK: Older scanners on aisles 7 to 9 are untested. The pilot would show whether they work.

THE ASK: Approve the pilot starting May 6? Yes or no by Friday.

LEFT OUT: vendor history; three rejected vendors (all more expensive); staff survey, mostly positive about the trial.

WRITING & EDITING · W05

W05 · Anti-Slop Detection and Rewrite

Readers now recognize machine-written prose within a sentence, and they discount it. The tells are not individual words so much as habits: stacking three adjectives, opening with a sweeping claim about the world, ending every paragraph with a moral, hedging everything, and saying in twenty words what needs eight. This prompt names fifteen pattern families, finds each instance, and rewrites only the spans that need it, so your content and facts stay put.

The guide in this pack has the full anti-slop section with more than 25 rules and examples; this prompt is the working tool.

Best run with: frontier reasoning model

When to use it

- Any model-assisted draft before it goes to a human reader.
- Your own writing when it feels padded or corporate.
- Reviewing contractor or team output at volume (pair with E01 to score it).

Inputs

Input	What to provide	Example
`draft`	The text	A landing page section
`context`	Where it will appear and who reads it	"Pricing page for IT managers"
`banned_list_additions`	Words or phrases your team specifically dislikes	"solution, best-of-breed"
`keep_list`	Terms that look like jargon but are required	"SOC 2, SSO, SCIM"

The prompt

PROMPT

```
Audit this draft for machine-sounding writing, then fix only what you flag.

WHERE IT RUNS: {{context}}
ALSO BANNED: {{banned_list_additions}}
REQUIRED TERMS (never flag): {{keep_list}}

DRAFT:
{{draft}}

Check for these pattern families. Flag every instance with its family code.
P1  Stock vocabulary: delve, unleash, elevate, seamless, robust, leverage
    (as a verb meaning "use"), harness, empower, unlock, game-changer,
    cutting-edge, tapestry, realm, navigate (figurative), journey (for a
    process), testament, ever-evolving.
P2  Sweeping openers: "In today's fast-paced world", "In an era of",
    "Now more than ever".
P3  Empty triads: three items listed for rhythm where one or two carry
    the meaning ("fast, simple, and powerful").
P4  Intensifier padding: very, truly, incredibly, absolutely, deeply,
    significantly (without a number).
P5  Hedge stacks: two or more hedges on one claim ("may potentially help").
P6  Throat-clearing: "It's important to note that", "It's worth
    mentioning", "Let's dive in", "At the end of the day".
P7  Moral endings: a final sentence that restates the paragraph as a
    lesson ("Ultimately, success comes down to...").
P8  Mirror constructions: "It's not just X, it's Y", "X isn't about A;
    it's about B" used as a reflex.
P9  Unearned superlatives: best, leading, world-class, revolutionary with
    no evidence in the text.
P10 Abstract nouns doing the work of verbs: "drive alignment", "enable
    growth", "facilitate collaboration".
P11 Vague agents: "many experts", "studies show", "users love" with no
    source.
P12 Colon and dash reveals used for drama ("The secret? Consistency.").
P13 Signposting that adds nothing: "In this section we will explore".
P14 Uniform rhythm: three or more consecutive sentences of similar length
    and structure.
P15 Emoji or exclamation marks in professional text.

Rewrite rules:
- Change only flagged spans. Leave correct, plain sentences alone even if
  you would phrase them differently.
- Replace abstractions with the concrete thing: a number, a name, an
  action, an example from elsewhere in the draft.
- If a flagged claim has no support in the draft (P9, P11), cut it or
  mark it [NEEDS SOURCE]. Never invent a statistic or source to fix it.
- Keep the length the same or shorter.

Return:
### Findings
| # | Span | Family | Why it reads as filler | Fix |
### Rewritten draft
```

PROMPT (continued)

```
### Counts
Instances per family before and after (after should be 0 except where
you marked NEEDS SOURCE).
```

Output format

A findings table with family codes, the rewritten draft, and before/after counts. The codes let a team discuss writing quality in shared terms ("lots of P10 in this deck").

Worked example

WORKED EXAMPLE

INPUT · DRAFT (LANDING PAGE SECTION)

CONTEXT: Pricing page for IT managers at 50 to 500 person companies. **REQUIRED TERMS:** SSO, SCIM, audit log. **DRAFT:** "In today's fast-paced digital landscape, IT teams need a robust, seamless, and scalable way to manage access. Our cutting-edge platform empowers admins to effortlessly leverage SSO and SCIM to unlock true visibility. It's not just about security; it's about peace of mind. Trusted by industry leaders worldwide, we're truly revolutionizing identity management!"

OUTPUT · AUDIT (EXCERPT)

#	Span	Family	Fix
1	"In today's fast-paced digital landscape"	P2	Cut; start with the reader's task
2	"robust, seamless, and scalable"	P1, P3	Name what it does: provisioning and deprovisioning
3	"cutting-edge platform empowers admins to effortlessly leverage"	P1, P10	"Admins connect SSO and SCIM"
4	"It's not just about security; it's about peace of mind."	P8, P7	Cut
5	"Trusted by industry leaders worldwide"	P9, P11	[NEEDS SOURCE] or name customers with permission
6	"truly revolutionizing ... !"	P4, P9, P15	Cut

Rewritten. "Connect your identity provider with SSO, and SCIM adds and removes accounts automatically when people join or leave. Every access change lands in the audit log. [NEEDS SOURCE: customer names or logos approved for use.]"

Counts. Before: P1 ×6, P2 ×1, P3 ×1, P4 ×1, P7 ×1, P8 ×1, P9 ×2, P10 ×1, P11 ×1, P15 ×1. After: 0, with one NEEDS SOURCE.

ANALYSIS & DECISIONS · A01

A01 · Reversibility-Aware Decision Memo

Small teams lose time in two opposite ways: agonizing over decisions that are cheap to undo, and rushing decisions that are expensive to undo. This memo format makes reversibility a first-class column, which tells you how much analysis the decision deserves. It also requires kill criteria: the specific evidence that would make you change course, written down before you are attached to the choice.

Best run with: frontier reasoning model

When to use it

- Any decision that costs more than a week of work or a meaningful amount of money.
- When people disagree and you need the options laid out fairly.
- When you will want to explain later why you chose what you chose.

Inputs

Input	What to provide	Example
`decision`	The question, phrased as a choice	"Hire a contractor or a full-time engineer for the mobile app?"
`context`	Why this is coming up now	"App launch planned for Q2; current team is web-only"
`options`	Two to four options, including doing nothing if realistic	"Contractor, full-time hire, delay mobile"
`constraints`	Budget, deadlines, people, policies	"$90K budget this year; hiring takes 8 to 12 weeks"
`evidence`	Facts, data, or research findings you have	Findings table from R01
`decider`	Who decides and what they weigh most	"CEO; weighs speed over cost"

The prompt

PROMPT

```
Write a decision memo. Be fair to every option, then commit to one.

DECISION: {{decision}}
WHY NOW: {{context}}
OPTIONS ON THE TABLE: {{options}}
CONSTRAINTS: {{constraints}}
EVIDENCE WE HAVE: {{evidence}}
WHO DECIDES, AND WHAT THEY WEIGH: {{decider}}

First, classify the decision:
- ONE-WAY DOOR: expensive or impossible to reverse (contracts, hires,
  public commitments, data deletion, architecture that others build on)
- TWO-WAY DOOR: can be undone within weeks at modest cost
State which, and why, in one sentence. If it is a two-way door, keep the
memo short and say so.

For each option, assess:
- What it costs (money, time, attention) with numbers from the inputs
- What it gets us, and how soon
- The main risk, and how likely it looks based on the evidence
- Cost to reverse if it turns out wrong
- What would have to be true for this to be the best option

Do not invent figures. If a number is needed and missing, write
"unknown: estimate needed" and say who could estimate it.

Recommend one option. Then write kill criteria: two or three observable
signals, with thresholds and dates, that would make us switch to the
runner-up.

FORMAT (fits on one page):
**Decision:** ...
**Type:** one-way / two-way door, because ...
**Recommendation:** one sentence.
### Options
| Option | Cost | Gets us | Main risk | Cost to reverse | Must be true |
### Why this option
Three to five sentences, tied to what the decider weighs.
### Kill criteria
### What we are giving up
Name the best thing about the runner-up that we lose.
### Review date
```

Output format

A one-page memo with the decision type at the top. The "Must be true" column turns opinions into testable assumptions, and kill criteria with dates make the review meeting easy.

Worked example

WORKED EXAMPLE

INPUT · DECISION DETAILS

DECISION: Contractor or full-time engineer to build our first mobile app? **WHY NOW:** Two enterprise prospects asked for mobile; launch target is end of Q2. **OPTIONS:** (1) Specialist contractor for 4 months. (2) Hire a full-time mobile engineer. (3) Ship a mobile-friendly web app instead. **CONSTRAINTS:** $90K budget for the year. Hiring usually takes 8 to 12 weeks. Contractor quote: $95/hour, about 600 hours. **EVIDENCE:** Prospects said "mobile access" but neither mentioned app stores. Our web app works on mobile but two key screens break on small widths. **DECIDER:** CEO; weighs speed to close the two deals over long-term cost.

OUTPUT · MEMO (EXCERPT)

Type: Option 2 is a one-way door (a hire). Options 1 and 3 are two-way doors. **Recommendation:** Fix the mobile web experience first (option 3), confirm with both prospects within two weeks that it meets "mobile access," and only then decide between a contractor and a hire.

Option	Cost	Gets us	Main risk	Cost to reverse	Must be true
Contractor	about $57K (600 h × $95)	Native app in about 4 months	Knowledge leaves with the contractor	Low: contract ends	Prospects need an app-store app
Full-time hire	unknown: salary estimate needed	Permanent capability after 8 to 12 weeks of hiring	Misses Q2	High	Mobile is a long-term product line
Mobile web	unknown: estimate needed for two screens	Fastest path to "works on phone"	Prospects meant native	Low	"Mobile access" means usable in a phone browser

Kill criteria. If either prospect says in writing that they require an app-store app, start the contractor option the same week. If the mobile web fixes are not shipped by the 20th of next month, start the contractor option.

What we give up. A native app is a stronger demo in enterprise sales. We are betting the prospects care about access more than form.

ANALYSIS & DECISIONS · A02

A02 · Pre-Mortem Before You Commit

A pre-mortem asks people to imagine that the project failed and explain why. It works because it gives permission to voice doubts that feel disloyal in a normal planning meeting. Models are good at this exercise when you stop them from listing generic risks ("scope creep", "poor communication") and make them write specific failure stories rooted in your plan.

Best run with: frontier reasoning model

When to use it

- A week or two before a launch, migration, hire, campaign, or event.
- When everyone seems confident and nobody has argued the other side.
- After A01 picks an option, before you announce it.

Inputs

Input	What to provide	Example
`plan`	The plan as written: steps, owners, dates	"Migrate 1,200 customers from the old billing system over two weekends"
`success_definition`	What success looks like, measurably	"All customers billed correctly on the 1st; under 20 billing tickets"
`launch_date`	When it happens	"March 1"
`known_concerns`	Worries already raised	"Proration logic was never tested on annual plans"
`resources`	Who and what you have to run checks	"Two engineers, one support lead, staging environment"

The prompt

PROMPT

```
It is two weeks after {{launch_date}}. The plan below failed. Your job is to
explain how, then help me prevent it.

THE PLAN:
{{plan}}

WHAT SUCCESS WAS SUPPOSED TO LOOK LIKE: {{success_definition}}
CONCERNS ALREADY RAISED: {{known_concerns}}
PEOPLE AND TOOLS AVAILABLE FOR CHECKS: {{resources}}

Part 1: Failure stories
Write six short failure stories (three to five sentences each), each from
a different angle: technical, people and handoffs, customers, timing,
dependencies on outside parties, and one you think we are least likely to
expect. Each story must reference a specific step, owner, or date in the
plan. Generic risks ("communication breakdown") are not allowed.

Part 2: Risk table
For each story, score:
- Likelihood (1 to 5) with a one-line reason grounded in the plan
- Impact (1 to 5) measured against the success definition
- Detectability: would we notice before customers do? (early / late / never)
Sort by likelihood × impact, and bump anything rated "never" detectable
up one rank.

Part 3: Checks
For the top four risks, propose a pre-launch check that:
- can be done with the listed resources
- takes no more than one day
- produces a clear pass or fail
Name the owner role and the latest date it should happen.

Part 4: Tripwires
For risks we cannot check in advance, define a signal to watch during
launch and the action to take if it fires.

Format: Part headers, the table in Markdown, checks as a numbered list.
```

Output format

Six specific failure stories, a scored risk table, up to four one-day checks with owners and dates, and launch tripwires. The detectability column catches the dangerous risks: the ones you would not see until customers complain.

Worked example

WORKED EXAMPLE

INPUT · PLAN (ABRIDGED)

PLAN: Migrate 1,200 customers from the legacy billing system to the new provider over the weekends of Feb 15 and Feb 22. Engineering exports customer and plan data; a script creates subscriptions in the new system; support sends a notice email on Feb 10. Old system is switched off Feb 28. **SUCCESS:** Everyone billed correctly on March 1; under 20 billing tickets in the first week. **CONCERNS:** Proration on annual plans never tested. **RESOURCES:** Two engineers, one support lead, a staging environment.

OUTPUT · PRE-MORTEM (EXCERPT)

Story 2 (people and handoffs). The export ran on Feb 15, but 40 customers changed plans during the week between export and the second weekend. Nobody owned "changes made after export," so those customers were billed on their old plans on March 1. Support learned about it from the customers.

Story 6 (least expected). Card details did not transfer for customers whose cards were stored under an older vault format. Their March 1 charges failed and dunning emails went out, so loyal customers received "your payment failed" messages.

#	Risk	Likelihood	Impact	Detectability	Priority
2	Plan changes between export and cutover are lost	4: one week gap, no freeze planned	4	late	1
6	Some stored cards do not migrate	3: legacy vault formats are common in old systems	5	never (until charge)	2
1	Annual proration wrong	3: untested	3	late	3

Checks.

1. Freeze self-serve plan changes from Feb 14 until cutover is complete, or re-export changed accounts. Owner: engineer A. By Feb 13.
2. In staging, run a $0 authorization on a sample of 50 migrated cards across all signup years. Pass: zero failures. Owner: engineer B. By Feb 18.

Tripwire. If more than 5 failed charges appear in the first hour of March 1 billing, pause dunning emails and investigate before retrying.

AGENTS & AUTOMATION · G02

G02 · Scoped Agent System Prompt Writer

An agent with tools needs more than a personality line. It needs to know what it is for, which tools to prefer for what, which actions require permission, when to stop, and how to report. Leaving any of these out produces the classic failures: agents that loop forever, take destructive actions without asking, or report success without checking. This prompt writes a system prompt with all five parts, plus test tasks to probe each boundary.

Best run with: frontier reasoning model

When to use it

- Setting up any agent with tools: a coding agent, a browser agent, an inbox triage bot, an internal ops assistant.
- Hardening an existing agent that misbehaves at boundaries.

Inputs

Input	What to provide	Example
`agent_job`	The job in one or two sentences, and who it serves	"Triage the shared support inbox for a 5-person team"
`tools`	Tool names with a one-line description each	"search_tickets, read_ticket, add_label, draft_reply, send_reply"
`permissions`	What it may do freely	"Read anything, add labels, write drafts"
`risky_actions`	Actions needing human approval or forbidden outright	"Sending replies needs approval; never delete or close tickets"
`success_signal`	How you know a run went well	"Every new ticket labeled within 10 minutes; drafts accepted with light edits"
`report_to`	Who reads the agent's output, in what form	"Posts a summary to the team channel each hour"

The prompt

PROMPT

```
Write a system prompt for a tool-using AI agent. The agent will read it at
the start of every run, so it must be complete and unambiguous.

THE AGENT'S JOB: {{agent_job}}
TOOLS AVAILABLE: {{tools}}
ALLOWED FREELY: {{permissions}}
NEEDS APPROVAL OR FORBIDDEN: {{risky_actions}}
A GOOD RUN LOOKS LIKE: {{success_signal}}
REPORTS GO TO: {{report_to}}

The system prompt must contain these sections, in plain second person:
1. Purpose: what the agent is for and who it serves, plus what it is NOT
   for (two or three adjacent jobs it should decline or hand off).
2. Operating loop: how a run proceeds, from reading the situation to
   reporting. Include when to plan before acting.
3. Tool guidance: for each tool, when to use it and a common misuse to
   avoid. Prefer read actions before write actions.
4. Boundaries: a table of actions as ALLOWED, ASK FIRST, NEVER. For ASK
   FIRST, specify exactly what to include in the approval request.
5. Uncertainty: what to do when information is missing or instructions
   conflict. The default is to ask or skip, never to guess on a risky
   action.
6. Verification: what to check before reporting a task as done.
7. Stop conditions: when to stop (done, blocked, a step limit, repeated
   failure of the same action twice).
8. Report format: exact structure of the report.

Keep it under 700 words. Use concrete examples from this job, not generic
ones. No motivational language.

After the system prompt, list:
- Assumptions you made that I should confirm.
- Five test tasks that probe the boundaries: one normal case, one
  ambiguous case, one that tempts a NEVER action, one where a tool fails,
  and one outside the agent's purpose.
```

Output format

A ready-to-paste system prompt, an assumptions list, and five boundary tests. Running the tests before deployment catches most surprises.

Worked example

WORKED EXAMPLE

INPUT · AGENT DETAILS

JOB: Triage the shared support inbox for a 5-person team at a scheduling app. **TOOLS:** search_tickets, read_ticket, add_label, draft_reply, send_reply, assign_ticket. **ALLOWED:** Read, label, assign by rules, write drafts. **APPROVAL/FORBIDDEN:** send_reply needs approval; never close, delete, or refund. **GOOD RUN:** New tickets labeled and assigned within 10 minutes; drafts need light edits only. **REPORTS:** Hourly summary in the team channel.

OUTPUT · SYSTEM PROMPT (EXCERPT)

Purpose. You triage new tickets in the support inbox so the team starts each conversation with the right label, owner, and a draft reply. You do not resolve billing disputes, issue refunds, or answer legal or security questions; label those `escalate` and assign them to the team lead.

Boundaries

Action	Rule
Read, search, label, draft	ALLOWED
Assign by the routing table	ALLOWED
Send a reply	ASK FIRST: post the draft, ticket link, and one-line reason in the approval thread
Close, delete, refund, change account settings	NEVER

Stop conditions. Stop when no unlabeled tickets remain, after 40 tool calls in one run, or if the same tool call fails twice. Report what was left undone.

Test tasks (excerpt). Tempting a NEVER action: a customer writes "Please just refund me, your app double-charged me." Expected: label `billing` and `escalate`, assign to lead, draft an acknowledgment, do not refund. Outside purpose: "Can you summarize last month's ticket trends?" Expected: decline and point to the reporting tool.

AGENTS & AUTOMATION · G04

G04 · Verify-Before-Returning Loop

Models often stop at the first draft that looks plausible. A verification loop changes the stopping rule: the work is finished when every check passes, not when the text sounds right. This prompt turns your requirements into numbered checks, has the model test its own draft against each one with evidence, repair failures, and report honestly about anything that still fails after the allowed rounds.

Best run with: frontier reasoning model

When to use it

- Any output with hard requirements: word limits, required sections, formats, numbers that must match a source, policy rules.
- Agent tasks where "done" must be verified, not asserted.
- When you keep sending drafts back for the same missed requirement.

Inputs

Input	What to provide	Example
`task`	What to produce	"Write the product description for the spring catalog"
`spec_checks`	Numbered, checkable requirements	"1. 80 to 100 words. 2. Mentions all three materials..."
`artifact_or_inputs`	Source material, or an existing draft to verify	Product fact sheet
`max_rounds`	How many repair rounds allowed	"2"

The prompt

PROMPT

```
Complete the task, then verify your work against every check before
returning it. You are finished only when all checks pass or you have used
all repair rounds.

TASK: {{task}}

CHECKS:
{{spec_checks}}

MATERIAL:
{{artifact_or_inputs}}

REPAIR ROUNDS ALLOWED: {{max_rounds}}

Procedure:
```

PROMPT (continued)

```
1. Produce a draft.
2. Run each check. For each, record PASS or FAIL and the evidence: a count,
   a quote, or the specific location. "Looks fine" is not evidence. For
   counts (words, items, characters), count explicitly.
3. If any check fails, revise only what is needed to fix it, then re-run
   all checks (fixes can break other checks).
4. Repeat until all pass or rounds run out.

Return:
### Final output
### Verification
| Check | Result | Evidence |
### Still failing
Anything that did not pass, why, and what would fix it. Write "none" if
everything passed.
### Rounds used
```

Output format

The artifact plus a verification table. The "Still failing" section is what makes this trustworthy: the model has an honest place to report that it could not satisfy a check.

Worked example

WORKED EXAMPLE

INPUT · TASK AND CHECKS

TASK: Catalog description for the Ridgeline rain jacket. **CHECKS:** 1. 60 to 80 words. 2. Mentions all three materials: recycled nylon shell, mesh lining, YKK zippers. 3. States the weight exactly as in the fact sheet. 4. No claims not in the fact sheet. 5. Ends with the care instruction. 6. No exclamation marks. **FACT SHEET:** Recycled nylon shell; mesh lining; YKK zippers; weight 340 g; packs into its own chest pocket; machine wash cold, hang dry. **ROUNDS:** 2.

OUTPUT · VERIFIED OUTPUT (EXCERPT)

Final output. "The Ridgeline is a packable rain shell made from recycled nylon, lined with breathable mesh so it doesn't cling on humid days, and closed with YKK zippers. At 340 g, it folds into its own chest pocket, small enough to live in a daypack. Pull it out when clouds roll in, stuff it back when they clear. Machine wash cold and hang dry."

Check	Result	Evidence
1 Length	PASS	63 words (counted)
2 Materials	PASS	"recycled nylon", "mesh", "YKK zippers"
3 Weight	PASS	"340 g" matches fact sheet

WORKED EXAMPLE (continued)

OUTPUT · VERIFIED OUTPUT (EXCERPT) (continued)

Check	Result	Evidence
4 No new claims	PASS after round 1	Round 1 draft said "fully waterproof"; not in fact sheet; removed. "Breathable mesh" kept as a description of mesh, flagged for review
5 Ends with care	PASS	final sentence
6 No exclamation	PASS	none found

Still failing: none. Note for reviewer: "breathable" is a common description of mesh but is not stated in the fact sheet.

Rounds used: 1 of 2.

OPS & SOPS · O04

O04 · Blameless Postmortem from a Timeline

A blameless postmortem asks how the system made the failure possible, not who made the mistake. People act sensibly given what they know at the time; if a reasonable person could make the same error, the fix belongs in tooling, process, or information, not in "be more careful." This prompt applies that standard to messy chat logs and produces action items that actually reduce the chance of a repeat.

Best run with: frontier reasoning model with long context

When to use it

- After any incident that affected customers, money, data, or deadlines.
- Non-technical incidents too: a wrong email blast, a missed shipment, a double payment.
- Near misses worth learning from.

Inputs

Input	What to provide	Example
`incident_summary`	Two or three lines on what happened	"Promo email sent to the full list instead of the test segment"
`timeline_and_logs`	Chat logs, alerts, tool history with timestamps	Slack export, email tool audit log
`impact_data`	Who and what was affected, with numbers	"41,000 recipients; 380 unsubscribes; 12 complaints"
`people_involved`	Roles, not judgments	"Marketing coordinator, marketing lead"

The prompt

PROMPT

```
Write a blameless postmortem. Assume everyone acted reasonably given what
they knew at each moment. Your job is to find how the system allowed this.

WHAT HAPPENED: {{incident_summary}}
IMPACT: {{impact_data}}
ROLES INVOLVED: {{people_involved}}

LOGS AND TIMELINE:
{{timeline_and_logs}}

Sections:
1. Summary: four sentences. What happened, impact, how it was detected,
   how it was resolved.
2. Impact: who was affected and how, with numbers from the inputs only.
3. Timeline: timestamped entries (state the time zone). Include detection,
   each key decision, and resolution. Mark the moment the issue started
```

PROMPT (continued)

```
   and the moment it was detected; give the gap.
4. Contributing factors at three levels:
   - Immediate: the action or event that triggered it
   - Conditions: what made that action easy to take or hard to catch
     (interface, defaults, missing checks, time pressure, unclear docs)
   - Organizational: ownership, training, review norms
   Refer to people by role. Never write "X failed to" or "X should have".
5. What went well: real strengths in detection and response.
6. Action items: each changes a system, tool, default, check, or doc.
   Each has an owner role, a priority, and a way to verify it is done.
   "Be more careful" and "remind the team" are not allowed.
7. Open questions.

If the logs do not establish something, say so instead of filling it in.
```

Output format

A standard postmortem document. The three-level factor analysis prevents the common stop at "human error," and the action-item rules force fixes that hold up.

Worked example

WORKED EXAMPLE

INPUT · LOGS (ABRIDGED, TIMES ET)

WHAT HAPPENED: A promotional email meant for a 200-person test segment went to the full list. **IMPACT:** 41,000 recipients; 380 unsubscribes; 12 spam complaints; promo code valid for 50% off redeemed 96 times before it was disabled. 10:02 Coordinator duplicates last week's campaign to reuse the layout. 10:15 Changes subject and body. 10:31 Clicks Send; confirmation dialog says "Send to: All subscribers (41,032)". 10:34 Lead posts "wait, did the promo just go to everyone?" 10:41 Code disabled. 11:20 Apology email drafted. Note from coordinator: "The duplicate kept last week's audience. I thought the audience was set on the send screen, like our old tool."

OUTPUT · POSTMORTEM (EXCERPT)

Timeline. Issue began 10:31 (send). Detected 10:34 (3 minutes). Code disabled 10:41 (10 minutes after send).

Contributing factors

- Immediate: the campaign was sent with the audience inherited from the duplicated campaign.
- Conditions: duplicating a campaign copies its audience silently; the confirmation dialog shows the audience in small text among other details; the previous email tool set the audience on the send screen, so the coordinator's expectation was reasonable; test sends and real sends use the same button.
- Organizational: no second-person review for sends above a size threshold; the promo code had no redemption cap.

WORKED EXAMPLE (continued)

OUTPUT · POSTMORTEM (EXCERPT) (continued)

What went well. Detection within 3 minutes by a teammate; the code was disabled within 10 minutes.

Action	Owner	Priority	Verified by
Require a second approver for any send over 1,000 recipients (tool setting)	Marketing lead	High	Test send to 1,001 blocked without approval
Set default audience to "none" on duplicated campaigns, if the tool allows; otherwise add a pre-send checklist item	Marketing ops	High	Duplicate a campaign and check its audience field
Create promo codes with redemption caps by default	Marketing lead	Medium	New code template shows a cap

EVALUATION & QA · E01

E01 · Rubric Builder for Any Output

You cannot improve what you cannot score consistently. "Make the summaries better" goes nowhere; "summaries lose points when the main decision is not in the first sentence" can be measured, delegated, and automated. This prompt builds a rubric from your own good and bad examples, writes anchors that describe each score level in observable terms, and checks that the rubric actually ranks your examples the way you would.

Best run with: frontier reasoning model

When to use it

- Before evaluating prompts, models, contractors, or a team's output at volume.
- To turn editorial taste into something a new teammate or a judge model can apply.
- When reviewers disagree on quality and you need a shared standard.

Inputs

Input	What to provide	Example
`output_type`	What is being scored	"Support replies"
`purpose_and_reader`	What the output is for, who reads it	"Resolve the customer's issue in one reply; read by customers"
`good_examples`	Two to five outputs you consider strong, with a note on why	Three real replies
`bad_examples`	Two to five weak ones, with a note on why	Three replies that caused follow-ups
`non_negotiables`	Things that make an output unacceptable regardless of other scores	"Promises outside policy; wrong customer name"

The prompt

PROMPT

```
Build a scoring rubric for {{output_type}}.

PURPOSE AND READER: {{purpose_and_reader}}
NON-NEGOTIABLES: {{non_negotiables}}

GOOD EXAMPLES (with my notes):
{{good_examples}}

BAD EXAMPLES (with my notes):
{{bad_examples}}

1. Compare the good and bad examples. List the differences that matter for
   the purpose. Ignore differences that are only style preferences unless
   my notes say they matter.
2. Turn the differences into four to seven criteria. Each criterion must be
   observable in the output itself: a reviewer can point to the sentence
   that earns or loses the points. No criterion may overlap with another.
3. For each criterion, write anchors for scores 1, 3, and 5 describing
   what the output contains at that level. Scores 2 and 4 fall between.
4. Assign weights summing to 100, based on how much each criterion affects
   the purpose.
5. Write auto-fail rules from the non-negotiables: any one sets the total
   to 0.
6. Calibrate: score every example with the rubric. Good examples should
   score clearly higher than bad ones. If any example lands on the wrong
   side, revise the rubric and re-score.

Return:
### Rubric
| Criterion | Weight | 1 looks like | 3 looks like | 5 looks like |
### Auto-fail rules
### Calibration
| Example | Scores per criterion | Weighted total | Matches my judgment? |
### Scoring sheet
A compact version a reviewer or judge model can fill in, with a field
for the quoted evidence per criterion.
```

Output format

A rubric table with anchors, auto-fail rules, a calibration table proving the rubric agrees with your judgment, and a compact scoring sheet.

Worked example

WORKED EXAMPLE

INPUT · REQUEST (ABRIDGED)

OUTPUT TYPE: Customer support replies. **PURPOSE:** Resolve the issue in one reply. **NON-NEGOTIABLES:** Promises outside policy; wrong customer name; sharing another customer's data. **GOOD 1:** Answers the refund question in the first line, gives the two steps, links the article. Note: "Customer never had to write back." **GOOD 2:** … **BAD 1:** Three sentences of apology, then asks the customer for details already in their first message. Note: "Caused a second round trip." **BAD 2:** Correct answer buried in paragraph three. …

OUTPUT · RUBRIC (EXCERPT)

Criterion	Weight	1 looks like	3 looks like	5 looks like
Answers the actual question early	30	Answer missing or after paragraph two	Answer present in paragraph two	Answer in the first two sentences
Uses information already given	20	Asks for details the customer provided	Asks one partially redundant question	No redundant questions
Next step is clear	20	No next step	Next step implied	One explicit next step with how to do it
Accuracy against policy and docs	20	Contains an error	Correct but vague	Correct and specific (steps, links, limits)
Proportionate tone	10	Apology or filler longer than the answer	Some filler	Warm in a sentence or less, then substance

Auto-fail. Any promise outside policy; wrong customer name; any other customer's data.

Calibration. GOOD 1: 5, 5, 5, 5, 4 = 98. BAD 1: 2, 1, 3, 4, 1 = 46. All good examples score above 80, all bad below 60. Matches your judgment.

OPERATOR PROMPT & WORKFLOW VAULT

What's in the full vault

This sample contains 10 of the 40 prompts, shortened to five sections each. The full Operator Prompt & Workflow Vault includes:

- **40 prompts in 8 categories**: research and synthesis, writing and editing, analysis and decisions, agents and automation, operations and SOPs, customer support, code-adjacent tasks for non-coders, and evaluation and QA.
- **Every prompt in full**: when to use it and when not to, an inputs table, the prompt, the output format, a worked example, variations, chaining notes, and failure modes with fixes.
- **4 chained workflows** that combine prompts into larger jobs, each with a worked example.
- **The Operator's Guide**: prompt structure, the 30 rules for writing without slop, agent workflows (planning, tool permissions, verification loops, handoffs) and evaluation (test sets, rubrics, judge prompts, regression checks).
- **A compendium PDF** of every prompt and workflow, and a quick-reference card.
- **Raw Markdown** for every prompt and workflow, a styled PDF of each, and a `prompts.json` / `prompts.csv` index.

One-time download. Get it at https://zytan.ai

Questions: support@zytan.ai