LUXE BUSINESS BACKEND™ · FOUNDER FIRE CODES™

THE AI FIRE CODES

Turn any recurring business fire into a playbook — with AI doing the heavy lifting.

BEFORE YOU READ

© 2026 TIFFANY LOPEZ · LUXE BUSINESS BACKEND™ · ALL RIGHTS RESERVED.

This guide is part of The AI Fire Codes for Founders and is licensed to the original purchaser for use inside their own business. That license covers you, your team, and the playbooks you build for your own company.

It does not cover resale, redistribution, republishing, teaching this material as your own, or rebuilding this system — in whole or in part — as a product, program, template, prompt pack, or AI tool for anyone else. The Fire Codes Method™, the Founder Fire Codes™ framework, the 9 Dependency Types, and the Whole-Picture Handoff™ are the intellectual property of Tiffany Lopez and Luxe Business Backend™.

The examples in this product are composite scenarios built to teach the method. They are not real client stories, and any resemblance to a specific company or person is coincidental. Nothing in this guide is legal, financial, or HR advice. You know your business; use your judgment.

Questions about licensing or team use: reach out through the address on your receipt.

WHAT'S INSIDE

12 The Last Word

From fighting fires to writing codes.

01 READ THIS FIRST

The nine playbooks in this product line cover the nine classic fires — the ones I see in almost every founder-run business. But your business has its own special ones.

Maybe it's the vendor who only responds when you personally email them. Maybe it's the retreat that eats your whole March because nobody else can build the run-of-show. Maybe it's a fire so specific to your business that no off-the-shelf playbook was ever going to cover it.

That used to be the limit of a product like this. I could hand you nine finished playbooks, and the tenth problem was yours to figure out alone.

Not anymore. This guide hands you the method I used to build the nine — the same six steps, in order, with nothing held back. Once you have the method, the count stops mattering. Fire number ten, fire number thirty, the weird one nobody else has: they all go through the same machine.

1	2	3	4	5	6
CAPTURE THE INCIDENT	NAME THE FIRE	TRACE THE DEPENDENCY	DRAFT THE RESPONSE	DRAFT THE PREVENTION	ROLL OUT & REVIEW

WHAT THE AI IS HERE FOR

Let me set the terms before we start, because this is the part people get wrong in both directions.

AI is here for the drafting. Not the judgment.

The slowest part of building a playbook was never the thinking — you already know your business. The slow part is the writing: turning an incident into a response plan, turning a dependency into prevention structures, turning "somebody should own this" into an actual briefing document. That's hours of drafting work, and it's exactly the kind of work a well-instructed AI does fast and does well.

What AI cannot do is know whether the draft is true. It doesn't know that your ops lead is three weeks into the role. It doesn't know that a $500 threshold is generous in your business and reckless in someone else's. It doesn't know which client would be insulted by a templated note. You know those things. So the deal, all the way through this system, is simple: AI drafts, you decide. Every output in this method passes through your hands before it touches your team.

6	2	1
STEPS — CAPTURE TO RETIRED	CHECKS BEFORE ANYTHING TOUCHES YOUR TEAM	DECIDER ON EVERY DRAFT: YOU

> **THE STANDING RULE**
>
> If you wouldn't sign your name to it, it doesn't ship. AI-drafted and founder-approved is the standard. AI-drafted and unread is how you end up with a playbook that embarrasses you in front of your own team.

WHAT'S IN THE SYSTEM

Your order includes seven pieces. Here's the map so nothing gets lost in a downloads folder:

- **This method guide** — the six steps, taught in full, with a worked example running through all of them.
- **The Fire Codes GPT** — a custom AI assistant trained on the method, the nine dependency types, and the playbook anatomy. Your access link ships separately with the product.
- **The Nine-Dependency Prevention & Solution Guide** — the full taxonomy: how each dependency type sounds, why it exists, and what replaces you in each one.
- **The Rapid Implementation Guide** — your first playbook, built and live, in seven days.
- **Fire Codes in Action** — three completed example playbooks, built with this method, start to finish.
- **The Incident-to-Protocol Worksheet** — the capture-to-playbook worksheet, ready to print or copy.
- **The Fire Codes Prompt Pack** — complete copy-paste prompts for founders who'd rather run the method through their own AI tool.

Read this guide first. Then open the Rapid Implementation Guide and build your first one. The rest of the library is there when you need it.

WHO THIS ASSUMES YOU ARE

This system is written for a founder with a real business: clients, revenue, a team — even a small one — and operations that actually move. If you're solo, most of the method still works, but the transfer steps assume there's someone to transfer to. It also assumes you're not trying to

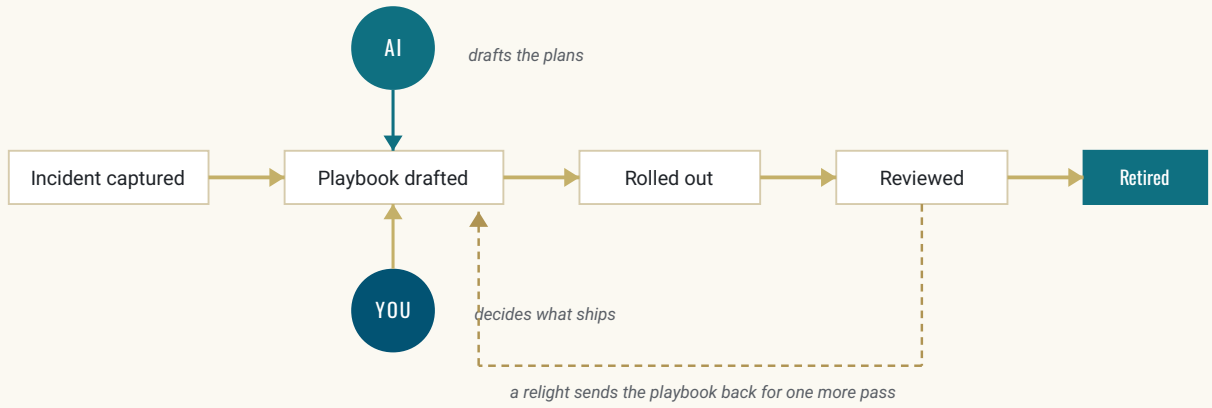
disappear from your business. You're trying to stop being its bottleneck — the approver, fixer, expert, and fallback for everything with a deadline. Those are different projects, and this one's better.

One thing I want you to hear before the first step: the fires in your business are not evidence that you built badly. They're evidence that you built fast — and that the backend grew slower than the front. Every fire in this method traces to a structural cause, and structure can be built. If you've been reading your recurring problems as a verdict on you or your team, put that down now. It was never true, and it's been making the problems harder to see clearly.

HOW THE METHOD MOVES

The six steps run in a deliberate order: three diagnostic steps, two drafting steps, one live step. Capture, name, and trace happen before any AI touches anything — because a draft built on a vague memory of the problem is a confident essay about the wrong thing. Then the two drafting steps run fast, because the AI is doing the assembly while you do the judging. Then rollout, which is slow on purpose, because that's where paper meets your actual Tuesday. Founders who struggle with this method almost always struggle in the same spot: they want to start at step four, because drafting feels like progress and diagnosing feels like homework. Resist. The diagnostic steps take under an hour combined, and they're what make the fast steps land.

THE FIRE CODES LOOP — FROM INCIDENT TO RETIRED FIRE



AI sits in the middle as the drafting engine; you sit under it as the decider. The loop runs until the fire stops relighting — then it retires, by name.

THE TIME BUDGET

Since you're the person with no spare capacity — that's why you're here — let's put numbers on what this actually costs you. A full playbook, first time through: about three hours of founder time, spread across a week or two. Capture and naming, twenty minutes each. The trace, twenty. The drafting sessions with the AI, an hour to ninety minutes across a sitting or two. The checks and hand-edits, thirty. The owner briefing and the rollout announcement, an hour combined. After the first one, the whole cycle compresses — founders report their third playbook taking half the time of their first, because the captures are already warm and the AI conversation has a rhythm.

Against that three hours, set the fire's price tag. Most recurring fires in an established business cost that much every two or three occurrences — some cost it weekly. This is the rare kind of business investment where the payback period is measured in incidents, not quarters. And if any single week of this method asks you for more than the budget above, something's wrong with how the week is scoped, not with you. Shrink the fire, not your sleep.

02 STEP ONE — CAPTURE THE INCIDENT

The method starts with a receipt, not a memory.

Founders are unreliable narrators of their own workload. Not dishonest — habituated. You've normalized your interruptions so thoroughly that by Friday you genuinely cannot reconstruct what pulled you out of deep work on Tuesday. You remember that the week was heavy. You cannot remember why, in any detail an AI — or a coach, or your own future self — could actually work with.

That's why step one happens while the fire is still warm. The next time a recurring problem lands on you, you take four minutes and write it down before you fix it, or immediately after. Not a journal entry. A capture. Four questions, answered plainly.

HOW TO CAPTURE

1 WHAT HAPPENED

The event, in plain words, with specifics. Not "the launch was stressful" — "the cart-open email wasn't written at 9pm the night before it was due, so I wrote it myself from 9:40 to 11:15." Dates, times, and numbers where you have them. Specifics are what make every later step work.

2 WHO BROUGHT IT TO YOU

Name and role. This tells you where the fire enters your day — and later, it usually tells you who's closest to owning the fix. If nobody brought it to you because you caught it yourself, write that down too. Self-caught fires point at a different kind of dependency than delivered ones.

3 WHAT YOU DID

Your actual response, step by step. This is the part founders skip, and it's the most valuable material in the whole capture — because whatever you did is a rough draft of the process nobody else has. You didn't just "handle it." You checked something, decided something, wrote something, in an order. Get the order down.

4 WHAT IT INTERRUPTED

What you were supposed to be doing instead, and what happened to that work. This is the cost line. A fire that eats a free Tuesday afternoon and a fire that eats your sales calls are different emergencies, and you'll use this later to decide which fires get playbooks first.

Add one more line if you can: **how many times this has happened before**, best guess. Even "at least four launches running" is useful. You're not building a court case; you're building enough of a record that pattern-spotting gets easy.

WHAT GOOD LOOKS LIKE

A good capture is short, specific, and slightly annoying to reread. If it's vague enough to describe anyone's business, it's too vague. If it names the person, the task, the time it ate, and the work it displaced, it's ready. One paragraph to half a page. Done in under five minutes, while the details are still sharp.

Keep your captures in one place — a note, a doc, a channel, wherever you'll actually put them. Three captures of the same fire, side by side, will teach you more about your business than a quarter of reflection.

> ★ FIRE MARSHAL'S NOTE
>
> Make the capture frictionless before you need it: put the four questions in a phone note template, or talk them into a voice memo on the walk between meetings and let AI transcribe. A capture you can do with your thumbs at 9pm is a capture that actually happens.

WHERE THIS GOES SIDEWAYS

Three failure patterns show up in step one, and all three are avoidable.

The capture becomes a vent. You sit down to record what happened and end up writing three paragraphs about how tired you are of it happening. Understandable — and useless downstream, because the AI can't draft a response plan from frustration. If you need to vent, vent, then delete it and write the four answers. The feelings are legitimate; they're just not receipts.

The capture waits for the weekend. "I'll write it up properly on Friday" is how captures die. By Friday you've lost the times, the exact wording, the order you did things in — the specifics that make every later step sharp. A rough capture written in the moment beats a polished one written from memory, every time. This is the same principle that makes a live tally beat a recollection: receipts over recall, always.

You capture the wrong layer. Founders often record the resolution instead of the interruption — "handled the client issue" instead of "client emailed me directly at 7am because she didn't trust the

account manager's answer, and I spent 40 minutes re-answering something my team had already answered correctly." The fire isn't that a problem existed. The fire is that it needed *you*. Aim the capture at the moment the thing landed on your desk, and write down why it couldn't land anywhere else. That "why" is what step three runs on.

WHY THIS COMES FIRST

Every later step runs on this raw material. The AI drafts from it. The dependency trace reads from it. The prevention plan is checked against it. A weak capture doesn't ruin the method — but it makes every step after this one slower and mushier. Four good minutes here buy you hours later.

WORKED EXAMPLE · THE TUESDAY SCRAMBLE

Through this whole guide, we'll follow one fire from capture to retirement so you can see the method end to end. It's a composite example — a pattern I've seen in plenty of businesses, not any one client's story.

The business: a course company, low seven figures, team of five. The fire: every launch, the emails get written the night before they send — by the founder — because only the founder can write them. Her capture, written at 11:20 on a Tuesday night: *"Cart-open email due 8am tomorrow. Asked Dana (marketing coordinator) for a draft Friday; what came back Monday didn't sound like us and undersold the bonus, so I rewrote it from scratch, 9:40–11:15pm. Dana brought it to me Monday 4pm asking if it was okay. It interrupted nothing on the calendar — it ate my night. This has happened every launch for at least the last five: I write or rewrite every email the night before it sends."*

03 STEP TWO — NAME THE FIRE

A problem with a name can be assigned, tracked, and retired. A problem without one just keeps happening to you.

Step two does two jobs: it decides whether this incident deserves a playbook at all, and if it does, it gives the fire a name your whole team will recognize on sight.

PATTERN OR ONE-OFF

Not every incident is a fire. A vendor missing one deadline in three years is weather. A vendor missing every third deadline is climate. Playbooks are for climate.

1 RUN THE PATTERN TEST

Has this happened three or more times? Can you predict roughly when it will happen again? If someone on your team heard the story, would they say "oh, that again"? Two yeses means it's a pattern. Write a playbook. One yes or none — log the capture and move on. Genuinely new problems get solved once, not systematized.

2 NAME IT

Give it a short, vivid, slightly unflattering name. "The Tuesday Scramble." "The Renewal Panic." "The Where's-the-File Hour." The name should make your team smirk in recognition. Avoid corporate labels — "Email Production Workflow Gap" names a document; "The Tuesday Scramble" names an enemy.

3 PUT NUMBERS ON IT

Two figures, rough is fine: how often it happens, and what it costs per occurrence. Cost in hours is the floor. If you can also see displaced revenue work, missed deadlines, or a team member's stalled week in there, note it. You're not building a spreadsheet — you're deciding whether this fire is worth two hours of playbook-building, and giving yourself a baseline so you'll know later whether the fix worked.

WHAT GOOD LOOKS LIKE

One line, and everyone in your company could finish it: name, frequency, cost. *"The Tuesday Scramble — every launch, roughly six times a year, two to four founder-hours per email, times five*

emails a launch." Say that sentence out loud. Notice that it's already more actionable than "launches are exhausting" — which is what you would have said last month.

Naming does something else too: it moves the problem outside of you. "I always end up writing the emails" sounds like a personality trait. "The Tuesday Scramble relit" sounds like a system fault — which is what it actually is. Your team can't help you fix a personality trait. They can absolutely help you fix a system fault.

> **WATCH FOR THIS**
>
> Founders resist naming fires that flatter them. The fire where you swoop in and save the launch feels like being needed. Name it anyway — the fires that feel best are usually the ones costing the most, because nobody's motivated to put them out.

WHERE THIS GOES SIDEWAYS

Naming the category instead of the fire. "Communication issues" is not a fire; it's a shrug with a label. A fire happens at a specific moment, in a specific process, and lands on a specific desk. If your name could apply to half the businesses in your industry, zoom in until it couldn't. "The Broken Handoff Between Sales and Delivery, Every Third Week of the Month" — now that's fightable.

Pricing it in feelings. "It's exhausting" is real and unbudgetable. Push through to hours and occurrences even when the estimate feels crude, because the crude number does two jobs the feeling can't: it tells you whether this fire outranks the other three on your list, and it gives step six a baseline to beat. Without a baseline, your 30-day review is two people exchanging impressions.

Skipping the pattern test because you're annoyed. The freshest fire always feels like the biggest one. But a spectacular one-off — the vendor who imploded once, the client from a bad dream — doesn't earn a playbook; it earns a story at dinner. Run the test even when you're sure. Especially when you're sure. Two minutes of honesty here saves you two hours of systematizing something that was never going to happen again.

One more thing about names, because it sounds cosmetic and isn't: the name is what lets your team join the fight. Nobody on your team can safely say "you're the bottleneck on launch emails" to your face. Every one of them can say "looks like the Tuesday Scramble is starting early this cycle." The name gives the problem a third-party address — something you and your team can stand on the same side of and look at together. That's the moment the fire starts belonging to the company instead of to you.

WORKED EXAMPLE · THE TUESDAY SCRAMBLE

Pattern test: five-plus launches running, predictable to the week, and when the founder mentioned it at a team meeting, two people laughed before she finished the sentence. Pattern, confirmed.

The name stuck immediately: **The Tuesday Scramble** — because launch emails always seemed to come due on Tuesdays, and the scramble always started after dinner.

The numbers: six launches a year, five emails a launch, two to four founder-hours per email. Somewhere between 60 and 120 founder-hours a year — spent at night, during the highest-revenue weeks on the calendar. Written down in one line and pinned to the top of her captures doc.

04 STEP THREE — TRACE THE DEPENDENCY

The fire is what you see. The dependency is what keeps relighting it.

Every recurring fire in a founder-run business burns on the same fuel: something the business needs is wired through you personally. Your judgment, your knowledge, your standards, your relationships, your push. Until you find which wire it is, every fix is a guess — and most guesses treat the symptom. You hire someone, buy a tool, run a training, and the fire relights, because the wire is still there.

Founder dependency isn't one disease. It's nine, and each one has its own cure. This is the diagnostic step, and it's the one place in the method where I want you to slow down, because a wrong trace sends every step after it in the wrong direction.

THE NINE TYPES, AND THE SYMPTOM TEST

Each dependency type has a tell — a sentence that comes out of a founder's mouth when that wire is live. Read the incident you captured, then read these nine sentences and ask: which one is this fire saying?

DEPENDENCY TYPE	THE SYMPTOM TEST
Direction & Decision	"The team checks with me before deciding or moving forward."
Revenue	"It only works when I'm the one selling, launching, or being the face."
Knowledge	"They ask me how to do it, or I'm the only one who knows the context."
Execution & Delivery	"It doesn't move unless I push it."
Quality & Risk	"They do the work, but I have to review it, fix it, or catch what they missed."
Coordination	"I'm the one connecting people and keeping everyone on the same page."
Relationship & Brand	"The client only trusts me with it."
Culture	"It gets done the way I'd do it only when I'm watching."
Systems & Infrastructure	"If I were unreachable for a week, things physically couldn't happen."

HOW TO TRACE

1 REREAD THE CAPTURE, NOT YOUR MEMORY

Work from what you wrote in step one. The capture says what actually happened; your memory says what usually annoys you. Those aren't always the same fire.

2 FIND THE PRIMARY

Ask: at the moment this fire needed me, what exactly was I supplying? A decision? An answer? A push? A standard nobody wrote down? A relationship nobody else holds? The thing you supplied is the primary dependency. Match it to the symptom test that fits best.

3 FIND THE SECONDARY

One fire usually carries one primary and one secondary type. Ask: if the primary were fixed tomorrow, what would still be leaking? That's your secondary. A launch-email fire that's primarily Knowledge (only you can write them) often carries Revenue underneath (money only moves when you're the voice). Note both — the response plan targets the primary; the prevention plan usually has to handle both.

4 CHECK IT AGAINST THE PERSON, NOT THE PROCESS

Dependencies are typed per process, not per founder. Your onboarding can be a Knowledge problem while your sales calls are a Relationship problem — same founder, different wiring. Don't let last month's diagnosis color this one.

WHAT GOOD LOOKS LIKE

A completed trace is two lines: *"Primary: Knowledge — I'm the only one who can write these. Secondary: Revenue — the launch voice is wired to me personally."* If you're torn between two types for the primary, that's normal — pick the one whose remedy you'd bet on, and note the runner-up. The Nine-Dependency Guide in this system gives you the full anatomy of each type when you want to go deeper than the symptom test.

> ★ FIRE MARSHAL'S NOTE
>
> Run the trace twice — once by you, once by the person who keeps bringing you the fire, separately. If you land on different types, neither of you is wrong: you found the two ends of the wire, and that conversation is usually worth more than the first draft it feeds.

> **THE ROOT CAUSE RULE**
>
> Whichever type you land on, the cause is structural, never personal. Your team checks with you because nobody defined decision authority — not because they're timid. They ask you how because the knowledge was never extracted — not because they're lazy. Trace the structure. Fixing people is a miserable hobby; fixing structure actually works.

WHEN THE TRACE IS HARD

Some fires trace themselves — the symptom sentence practically reads itself off your capture. Others sit stubbornly between two or three types, and it's worth knowing the common confusions.

Knowledge vs. Quality. Both end with you fixing things. The difference is what was missing when the work was done: if your team didn't have the information or context to do it right, that's Knowledge. If they had the information but no written definition of what "right" looks like, that's Quality. Ask: could they have described what a great version of this looks like before starting? If yes, but the work still needed your catch — Quality. If no — Knowledge.

Direction & Decision vs. Execution & Delivery. Both look like work waiting on you. But waiting for your *permission* is a Decision problem, and waiting for your *push* is an Execution problem. The tell: when you finally engage, do you make a call, or do you just apply pressure? A call means the decision architecture is missing. Pressure means the accountability loop is.

Relationship & Brand vs. Revenue. Both show up in sales and renewals. If prospects and clients want the company's result but only through you, trace whether the "through you" is about trust (they'd take a worse deal to keep you personally — Relationship) or about capability (nobody else can actually run the sales conversation — Revenue). Most founder-led sales fires carry both; the question is which one is primary, because the trust fix and the extraction fix are different projects.

When you're genuinely stuck, run the counterfactual test on each candidate: imagine that dependency fully fixed — the decision rights written, the knowledge extracted, the trust transferred — and replay the incident. Would the fire still have happened? The type whose fix would have prevented the incident is your primary. And if two fixes both would have prevented it, congratulations: you found a primary and a genuine secondary, and your prevention plan just got its scope.

WORKED EXAMPLE · THE TUESDAY SCRAMBLE

What was the founder supplying at 9:40pm? Not a decision, not a push — the emails were getting written, just badly. She was supplying what only she had: the voice, the offer logic, what each email in the sequence is supposed to do, which claims are fair game and which aren't. Nobody ever extracted any of that. **Primary: Knowledge.**

And if the knowledge transferred tomorrow? The launch calendar would still bottleneck on her availability to approve, and the audience relationship would still route through her name alone. **Secondary: Revenue.**

Trace, written down: *Tuesday Scramble — primary Knowledge, secondary Revenue. The fix is extraction and a written email standard, not "Dana tries harder."*

05 STEP FOUR — DRAFT THE RESPONSE PLAN

This is where the AI earns its keep.

You now have a captured incident, a named fire, and a traced dependency. In the old world, the next move was a blank page: sit down and write out how this fire gets handled from now on. That blank page is where most founders' systemizing dies — not for lack of knowledge, but for lack of a free evening.

So don't start from blank. Feed everything you've built to the Fire Codes GPT — the capture, the name, the numbers, the trace — and ask it for a response plan draft. It knows the method, it knows the nine types, and it knows what a finished playbook looks like. What it doesn't know is your business, which is exactly what your capture supplies.

WHAT A RESPONSE PLAN IS

The response plan is the this-week half of a playbook. It answers one question: the next time this fire starts, what happens — and who does it — so that it gets handled without routing through you? It's not the permanent fix; that's step five. It's the protocol that stops the bleeding while the permanent fix takes hold. A good response plan has steps in order, one name on each step, and a clear line for when something escalates — and to whom, which should almost never be you.

HOW TO DRAFT WITH AI

1 FEED IT THE WHOLE TRAIL

Paste the capture, the fire's name and numbers, and the dependency trace into the GPT. Then tell it what it can't infer: who's on the team, what tools you run on, and any hard constraints ("Dana is part-time," "we use ConvertKit," "nothing sends without a second set of eyes"). Thin input gets you a generic draft. Rich input gets you a draft that's eighty percent right.

2 ASK FOR THE THIS-WEEK VERSION

Tell it: draft the response plan — the steps my team runs the next time this fire starts, without me in the loop. Ask for numbered steps, an owner per step, and an escalation line. If it hands you a strategy essay instead of a protocol, say so and make it tighten up.

3 PUSH BACK ON THE DRAFT

First drafts come back too polite and too generic. Push: "Step 3 assumes we have a copywriter; we don't — redo it with Dana owning drafts." "What happens if the owner is out sick that day?" "Tighten this to fit on one page." The GPT revises fast and doesn't get defensive. Three rounds of pushback usually beats an hour of solo writing.

4 EDIT BY HAND UNTIL IT'S TRUE

Then take the draft and make it yours: real names, real thresholds, real tools, sentences you'd actually say. The AI's draft is scaffolding. The judgment calls — who owns what, what escalates, what's good enough — are yours, and they're the part your team will actually feel.

WHAT GOOD LOOKS LIKE

One page or less. Numbered steps a new hire could follow cold. A name on every step — a role name at minimum, a person's name once you assign it. An escalation line that names a condition and a person. And this test above all: read it and ask, *where am I in this plan?* If the honest answer is "approving one thing at a defined checkpoint" or "nowhere," it's good. If you're load-bearing in three steps, the plan just reinstalled the dependency with better formatting.

> ★ FIRE MARSHAL'S NOTE
>
> Keep a prompts file. Every time a pushback line earns its keep — “define complex or cut it,” “one page, verbs first” — save it. By playbook three you’ll open with your six best lines and skip two rounds of revision entirely.

> DRAFT FAST, DECIDE SLOW
>
> Let the AI produce in minutes what used to take you an evening — then spend your saved time on the part it can't do: deciding whether the plan is true. Ten minutes of founder judgment on a fast draft beats two hours of founder drafting every single time.

WHERE THIS GOES SIDEWAYS

Accepting the first draft because it sounds professional. AI drafts arrive wearing a suit. Numbered steps, confident verbs, clean formatting — everything about the presentation says "done." Read past the tailoring. The most dangerous drafts are the plausible ones, because plausible-but-wrong ships, and true-but-messy gets fixed. If you catch yourself approving a plan

because it reads well rather than because you checked it against your actual Tuesday, stop and reread it as the skeptic who has to live with it.

Building the response plan for the ideal team. The draft assigns a step to "the project coordinator" — a role you don't have — or gives your newest hire a judgment call they're six months from being ready for. The AI staffs plans with the team the plan deserves; you have to staff it with the team you employ. If the honest answer is "nobody can own this step yet," that's not a flaw in the method — that's the method telling you the response plan has a hiring or training prerequisite. Write that down; it's real information.

Letting the response plan try to be the prevention plan. These are different documents with different jobs. The response plan assumes the fire will start again and gets it handled without you. The prevention plan works on making it not start. When a draft mixes them — response steps that quietly include "rebuild the whole workflow" — the plan becomes too heavy to run during an actual fire. Keep response light enough to execute on a bad day. The deep fixes have their own step, and it's next.

> **WORKED EXAMPLE · THE TUESDAY SCRAMBLE**
>
> The founder pasted her capture, the name, the numbers, and the trace into the GPT, added her roster (Dana, marketing coordinator; Priya, ops lead) and her stack, and asked for the response plan. Three rounds of pushback later, the plan fit on one page:
>
> 1. Five business days before any send date, Dana drafts the email using the launch email standard (built in step five) and the last launch's sequence as reference. 2. Dana self-checks against the standard's checklist before it goes anywhere. 3. Priya reviews for offer accuracy — price, dates, bonus terms — against the launch doc. 4. The founder gets one 15-minute review window, 48 hours before send, for voice only. She may edit lines; she may not rewrite from scratch. 5. If a draft misses the standard twice, it escalates to Priya — not the founder — to diagnose whether the standard or the brief was unclear.
>
> Her hand-edits: she cut the AI's suggested "founder final sign-off on all emails" step entirely — it recreated the fire — and she set the review window at 48 hours because 24 had night-before written all over it.

06 STEP FIVE — DRAFT THE PREVENTION PLAN

The response plan handles the fire. The prevention plan removes the fuel.

Prevention is where the dependency actually gets replaced — where the thing you've been personally supplying gets built into structure that supplies it instead. This is the half of the playbook that makes the fire stop recurring rather than just stop hurting.

THE REPLACEMENT STRUCTURES

Whatever the dependency type, prevention comes down to five kinds of structure. Your fire won't need all five, but you should check each one against your trace:

- **Ownership** — one name accountable for the outcome, not the task list. "The team handles it" means you handle it.
- **Boundaries** — what the owner decides alone, in writing. Thresholds, authority, the gray areas pre-decided.
- **Escalation rules** — what goes up, when, and to whom. An escalation path that ends at you is the dependency wearing a costume.
- **Standards** — what good looks like and what unacceptable looks like, with real examples of both. Standards without examples are just vibes.
- **Triggers** — the conditions that start the process, written as facts a receiver can watch for. You've been the company's alarm system; this is the alarm, externalized.

THE WHOLE-PICTURE LAYERS

When you hand the rebuilt process to its owner, the handoff carries seven layers — this is the Whole-Picture Handoff™ format, and it exists because delegation historically failed by shipping layer one and keeping the other six in your head:

- **What** — the outcome, with a definition of done you could check on a Friday afternoon.
- **When** — the triggers, scheduled and conditional, that tell the owner it's time.
- **Why** — the context: what this connects to, what breaks if it's done wrong.
- **How** — steps, tools, examples. Enough that nobody has to call and ask what you meant.
- **Standard** — great and unacceptable, side by side, with real examples.

- **Judgment** — what the owner decides alone, what escalates, how to think about gray areas.
- **Voice** (where relevant) — how we sound, what we never say, how we deliver bad news.

HOW TO DRAFT WITH AI

1 ASK FOR PREVENTION STRUCTURES AGAINST THE TRACE

Same conversation, next move: "Now draft the prevention plan. Primary is Knowledge, secondary is Revenue — what structures replace me?" A well-briefed AI will propose ownership, standards, and triggers matched to the dependency type. Make it get specific to your capture, not generic to the type.

2 BUILD THE HANDOFF LAYERS

Have it draft all seven layers for the process owner. The What, When, and How it can draft credibly from your capture. The Standard, Judgment, and Voice layers it can only scaffold — it will hand you placeholders and its best guesses, and those guesses are exactly where your hand-editing matters most.

3 RUN THE FOUNDER INVOLVEMENT CHECK

Read the full prevention draft and ask: does anything still route through me that shouldn't? Every point where you appear should survive one question — am I here because only I can do this, or because the draft was being polite to my ego? Chapter 10 gives you the full checklist.

4 RUN THE ESCALATION CHECK

Every step has an owner; every failure mode has a path; the path doesn't dead-end at you. If the plan's answer to "what if it goes wrong?" is "the founder steps in," send it back for another round. Also in chapter 10.

WHAT GOOD LOOKS LIKE

A finished prevention plan reads like an operating change, not a wish. Structures named. An owner named. Layers drafted. And when you read it, you should feel a small, specific discomfort — the feeling of a thing you've always personally held being written down where anyone can run it. That feeling is the point. Mildly uncomfortable and finished beats perfectly comfortable and still in your head.

> **DEPTH OVER COVERAGE**
>
> Don't try to prevent every fire at once. One fire, fully rebuilt — ownership, boundaries, standards, triggers, handoff — teaches your team how this works and buys you real hours back. Three fires half-prevented buys you three new ways to be disappointed.

WHERE THIS GOES SIDEWAYS

Standards without examples. This is the most common gap in AI-drafted prevention plans, because the AI literally cannot supply your examples — it's never seen your work. A standard that says "emails should match our brand voice" changes nothing. A standard with three real lines that nailed the voice and three that missed it teaches the voice to someone who's never heard you explain it. Every standard in your prevention plan needs real examples of great and unacceptable, pulled from your actual archive. Budget twenty minutes for the pulling; it's the highest-leverage twenty minutes in the step.

Ownership assigned but authority withheld. The plan names an owner, and then every meaningful decision inside the process still requires a check-in. That's a title transfer with you still holding the steering wheel. If you're not ready to give someone the authority a process needs, that's worth knowing — but solve it honestly (train them toward it, stage the authority in phases, pick someone else) instead of shipping a plan that pretends.

Triggers that still need you to notice. Read every trigger in the draft and ask who detects it. "When a client seems unhappy" requires a person with your instincts watching — which is the alarm-system dependency wearing new clothes. Push every trigger toward something detectable without judgment: a date, a count, a threshold crossed, a form submitted. Where genuine pattern-recognition is required, do the extraction: what did you notice, the last five times, right before you stepped in? Whatever you were noticing, write it as a signal the owner can watch for. Your instincts are transferable. They just have to be named first.

WORKED EXAMPLE · THE TUESDAY SCRAMBLE

The prevention plan the founder landed on, after two rounds with the GPT and a hand-edit pass:

Extraction: one 90-minute recorded session where she walked through the last three launches' email sequences — what each email does, how she decides the angle, what she always checks before send. The GPT turned the transcript into a launch email standard: sequence anatomy, voice rules with six real example lines (three great, three "never again"), claim boundaries, and a pre-send checklist.

Ownership: Dana owns launch email production, brief to scheduled send. **Boundaries:** Dana decides angles and subject lines alone; price, dates, and bonus terms come only from the launch doc. **Trigger:** the email calendar generates automatically when a launch date is set — no email is ever due in less than five days. **Escalation:** misses-the-standard twice goes to Priya; anything legal or claims-adjacent goes to the founder, which she kept deliberately — that one is genuinely hers.

Voice layer: she rewrote the GPT's guesses herself, line by line. Nobody drafts her voice rules but her. That took forty minutes and was the best forty minutes in the whole build.

07 STEP SIX — ROLL OUT AND REVIEW

A playbook nobody rolled out is a document. The fire doesn't care about documents.

Everything so far happened on paper. Step six is where the business actually changes — and it's the step founders shortcut most, because by now the problem feels solved. It isn't. A fire is out when it stops relighting, and you only know that by watching.

HOW TO ROLL OUT

1 ASSIGN THE OWNER — FORMALLY

One name on the whole playbook, told directly, with the full handoff package. Not "hey, can you kind of take this over" in a hallway. The owner reads all seven layers, asks their questions, and explicitly confirms they've got it. If they push back on a piece of the plan, listen — the person about to run a process sees holes the person escaping it can't.

2 COMMUNICATE THE CHANGE TO EVERYONE IT TOUCHES

The team hears what changed, who owns it now, and — this matters — what to do instead of coming to you. "Launch email questions go to Dana now" has to be said out loud, by you, or the old wiring quietly wins. Chapter 11 covers the rollout messages in full, including the ones AI should draft and the one rule about who delivers them.

3 WATCH FOR RELIGHTS

A relight is any occurrence of the fire after rollout — the question that still comes to you, the deadline that still slides, the draft that still lands in your lap. Log every one, same four-minute capture as step one. Relights aren't failure; they're feedback with an address on it. Each one tells you exactly which layer of the package was thin.

4 REVIEW ON A CADENCE, THEN RETIRE

Two weeks in: short review with the owner. What ran without you? What relit, and why? Fix the package — not the person — and keep watching. When the process has run clean without you through two or three real occurrences, retire the fire: mark the playbook stable, thank the owner where the team can hear it, and take the fire off your worry list on purpose.

THE REVIEW AGENDA

The two-week review works best as a short, fixed agenda — fifteen minutes, owner talking, you mostly listening. Five questions, same every time. What ran without me since rollout? (Let them tell you the wins first; they earned the floor.) What relit, and what does the log say about why? Which part of the package turned out to be wrong, missing, or harder to use than it looked? What decision did you make inside your boundaries that you want me to know about — not approve, know about? And what do you need — from the document, the team, or me — to make the next two weeks cleaner? Then one move that matters more than the whole agenda: every fix that comes out of the conversation goes into the playbook before the meeting ends, with the owner doing the typing. The document stays theirs, the improvements carry their fingerprints, and the review ends with the playbook better than it started — which is the entire point of reviewing. Resist adding your own agenda items about adjacent processes; scope creep in reviews is how a fifteen-minute check-in becomes a meeting nobody wants to attend twice.

WHAT GOOD LOOKS LIKE

The honest markers: the fire's trigger event happened — a launch ran, a renewal came due, a client escalated — and you found out afterward, or not at all. The owner made a judgment call inside their boundaries without checking first. A relight got logged and traced to a thin layer, and the fix went into the document instead of your task list. And the number you wrote down in step two moved: fewer occurrences, fewer founder-hours, or both.

> ### THE URGE TO TAKE IT BACK
>
> Somewhere in the first month, the process will wobble and every instinct you have will say "faster if I just do it." That urge is the dependency asking for its job back. Fix the package, brief the owner, and sit on your hands. The second clean run is where the release becomes real.

WHERE THIS GOES SIDEWAYS

The quiet rollout. The playbook gets finished, the owner gets a link, and the team finds out by inference. Three weeks later the old questions are still coming to you, because as far as anyone knows, you're still the address. A change nobody announced is a change that didn't happen. The rollout meeting isn't ceremony — it's the mechanism by which the old wiring actually gets cut, in public, where the whole team can update at once.

Reading relights as verdicts. First relight, and the little voice says "see, it doesn't work, they can't do it." Wrong reading. A new process wobbling on its first real occurrence is a new process behaving normally. The verdict question isn't "did it wobble?" — it's "did the wobble get fixed in the document, and did the next run go cleaner?" A playbook that improves through its relights is succeeding. The only genuinely bad sign is a relight that gets fixed by you doing the work, silently, with no entry in the log — because that's not a relight, that's a relapse.

Skipping the retirement. Founders finish the hard part and forget the last move: telling the team the fire is out. Retirement matters more than it looks like it does. It marks the win publicly, credits the owner where everyone can hear it, and teaches your team that this method finishes things — which is exactly the reputation the method needs the next time you hand someone a playbook. A fire retired by name at a team meeting does more for your systems culture than a quarter of speeches about ownership.

WORKED EXAMPLE · THE TUESDAY SCRAMBLE — RETIRED

Rollout: the founder briefed Dana with the full package on a Thursday; Dana came back with four questions, two of which improved the standard. At Monday's team meeting the founder announced the change herself and said the sentence out loud: "Launch email questions go to Dana."

Launch one under the new plan: four of five emails ran clean. One relight — a bonus deadline was wrong in a draft, caught at Priya's check. Traced to a thin How layer: the launch doc's location wasn't in the package. Fixed in ten minutes.

Launch two: five for five. The founder's total email involvement was two 15-minute review windows, at her desk, in daylight. At the next team meeting she retired the Tuesday Scramble by name — and started a capture on the next fire that same week, because the method was warm and the machine was hers now.

08 WORKING WITH THE FIRE CODES GPT

The Fire Codes GPT is a drafting partner that already speaks this language. Here's how to get real work out of it.

WHAT IT KNOWS

The GPT ships pre-loaded with three things: the Fire Codes Method™ (all six steps, in order, including what each step's output should look like), the nine dependency types (symptoms, structural causes, and the replacement structures for each), and the playbook anatomy this product line uses — Scene, what's burning, response plan, prevention plan, ownership, proof it's out. You never have to teach it the framework. You only have to teach it your business.

What it does not know: your team, your tools, your thresholds, your clients, your voice, or anything that happened in your company. Everything specific comes from what you paste in. Which means the quality of what you get out tracks almost exactly with the quality of the capture you feed it — another reason step one earns its four minutes.

> **GETTING ACCESS**
>
> Your access link for the Fire Codes GPT ships separately with this product — check your purchase delivery for the setup instructions. If you'd rather run the method through your own AI tool instead, the Fire Codes Prompt Pack in your order covers you; more on that at the end of this chapter.

THE CONVERSATION PATTERN

Working with the GPT is a conversation, not a vending machine. The pattern that produces good playbooks:

1 GIVE IT THE INCIDENT

Open with your capture — the full four-question version — plus the fire's name, the numbers, and your trace if you've done it. Add your roster and stack. The more receipts you hand it, the less it invents.

2 ANSWER ITS QUESTIONS

It's built to ask before it drafts — who touches this process, what's the real trigger, what's been tried. Answer honestly, including the unflattering parts. "I've taken this back from two different people" is diagnostic gold; don't dress it up.

3 PUSH BACK ON DRAFTS

Treat the first draft as an opening offer. Tell it what's wrong plainly: "we don't have that role," "that threshold is too loose," "step four still routes through me — redo it." It revises without sulking, which makes it a better drafting partner than most humans on a deadline.

4 ASK FOR TIGHTER LANGUAGE

AI drafts run wordy. Once the substance is right, ask it to cut: "fit this on one page," "make every step start with a verb," "cut anything a new hire wouldn't need." Tight documents get used. Long ones get skimmed once and abandoned.

CONVERSATION HYGIENE

A few working habits that keep sessions productive. Run one conversation per fire — the GPT holds your context within a session, and a fire's capture, trace, drafts, and revisions belong in one thread where each step can see the last. Don't make it hold your whole business at once; a conversation juggling three fires drafts all three worse. When you come back to a fire after a break, re-paste the current playbook version rather than trusting the thread's memory of it — your hand-edits happened outside the conversation, and the GPT doesn't know about them until you show it. Date your playbook versions in your own doc, because "which draft was the good one" is a real question three weeks later. And know when to stop: the GPT will happily revise forever, and there's a version of this work where polishing drafts becomes a productive-feeling way to avoid rolling anything out. Three or four rounds is usually the ceiling of useful. If round five is still moving words around, the document is done and the founder is stalling — ship it to step six.

WHAT TO ALWAYS EDIT BY HAND

Some things never ship straight from the machine, no matter how good the draft looks:

- **Names and assignments.** The AI will guess at who should own what. Ownership is a leadership decision, and your team can tell when one was made by a robot.
- **Thresholds.** Dollar amounts, timelines, approval limits. The GPT proposes plausible defaults; only you know what's actually safe in your business.

- **Anything client-facing.** Every word a client might read gets rewritten in your hand. A client can smell a template through the screen.
- **The Voice layer.** How your company talks is the one thing an AI can only ever approximate. Draft it yourself, from real examples of your own writing.

WHEN THE GPT GETS IT WRONG

It will, sometimes, and the failure modes are worth naming so you catch them fast. **Confident invention:** a detail you never provided, stated as fact — a tool, a role, a number. This is why the quality checklist below starts with traceability; anything you can't trace to your own input gets challenged or cut. **The generic drift:** three revisions in, the plan starts sounding like it was written for a mid-size software company instead of your business — usually a sign the conversation has run long and thin. Re-anchor it: paste your capture back in and say "ground the next revision in this incident specifically." **The agreeable spiral:** you push back, it capitulates instantly — even when your pushback was wrong. The GPT optimizes for cooperation, not truth, so don't treat its agreement as validation. If a structural question actually matters — should this escalate to the ops lead or the program director? — make the case both ways yourself, or ask it to argue the other side before you decide. And **the scope balloon:** asked for a response plan, it delivers a change-management initiative with phases. Cut it back without mercy. None of these failure modes make the tool unusable; they make it a drafting partner with known habits — which is exactly how you should treat it. You wouldn't fire a talented copywriter for needing an editor. You'd just never skip the edit.

QUALITY CHECKS BEFORE YOU TRUST AN OUTPUT

Run every draft through this before it goes anywhere near your team:

- [ ] Every fact in the draft traces back to something I actually told it — nothing invented, no made-up names, tools, or numbers.
- [ ] Every step has an owner, and none of the owners are fictional roles we don't employ.
- [ ] The escalation path exists and doesn't dead-end at me.
- [ ] The thresholds are ones I chose, not ones it guessed.
- [ ] I've read the whole thing, out loud or close to it — not skimmed the headings.
- [ ] I'd be comfortable if my ops lead knew which parts the AI wrote, because I've made every part true.

WHAT A GOOD EXCHANGE LOOKS LIKE

Here's the shape of a working session, so you can recognize when yours is on track. Notice the founder isn't writing anything — she's supplying facts and rendering verdicts, which is the correct division of labor.

> **THE PATTERN IN MINIATURE**
>
> *Founder: "Here's my capture, trace, and roster. [Pastes all three.] Draft the response plan."*
>
> *GPT: "Before I draft — three questions. Who currently notices when this fire starts? What has the client success manager already tried? Is there a dollar amount above which you'd genuinely want to be consulted?"*
>
> *Founder: "I notice it, which is part of the problem. She's tried reminder emails. And yes — anything over $10K in annual value, but only as an FYI, not an approval."*
>
> *GPT: [Drafts a five-step plan.]*
>
> *Founder: "Step 2 gives her a script — good — but step 4 says 'loop in the founder for complex cases.' Define complex or cut it. Also this is a page and a half; make it a page."*
>
> *GPT: [Revises: 'complex' becomes two named conditions routed to the ops lead, plan tightens to one page.]*
>
> *Founder: "Better. Now the prevention plan, same rules."*

Two things make that exchange work. The founder answered the clarifying questions with facts, including the mildly embarrassing one — "I notice it, which is part of the problem" is exactly the kind of admission that sharpens a draft. And she treated vagueness as a defect to be flagged, not politeness to be accepted. "Define complex or cut it" is the single most useful sentence in AI-assisted playbook drafting. Use it liberally.

IF YOU'D RATHER USE YOUR OWN AI

Some founders already live in a particular AI tool and want the method there. The Fire Codes Prompt Pack in your order exists for exactly this: a master system prompt that teaches any capable AI assistant the method, the nine types, and the playbook anatomy, plus task prompts for each step. You'll do a little more setup than with the GPT, and you should expect to push back on drafts a bit harder — but the method runs the same. The machine matters less than the material you feed it and the judgment you apply on the way out.

09 AI-ASSISTED CUSTOMIZATION OF THE NINE

Before you build anything new, squeeze what's already built.

If the fire burning you this month is one of the nine classics, don't run the full method from scratch — the playbook already exists. The fastest path in this whole system is customization: feed the GPT one of the nine finished playbooks plus your specifics, and get back a version with your team's names on it, your thresholds in it, and your tools wired through it. What takes a session with the method takes about twenty minutes this way.

The move: paste in (or reference) the playbook, then give the GPT your roster with roles, your tool stack, your relevant thresholds, and any capture notes you have on how this fire shows up in your business. Ask it to return the same playbook, customized — same structure, your reality. Then hand-check the parts that carry judgment.

A customization session runs in three moves. First, the feed: the playbook plus a tight paragraph about your business — what you sell, at what price point, to whom — then the roster, the stack, and any capture notes on how this fire actually shows up for you. Second, the ask: "return this same playbook, same structure, customized to my reality — and mark anything you had to guess." That marking instruction matters; it converts silent guesses into visible questions. Third, the hand-check: go straight to the judgment-bearing parts — assignments, thresholds, client-facing language — and verify each one against the table below.

Where customization earns its keep is the distance between the template's assumptions and your reality. The Approval Pileup playbook assumes decisions worth defining thresholds for; if your version of the pileup is creative approvals rather than spending approvals, say so in the feed and watch the whole playbook re-aim itself. The nine playbooks are good defaults. Your specifics are what make them installable.

Here's the per-fire summary — what to feed the AI for each of the nine, and what to verify yourself before rollout:

FIRE	WHAT TO TELL THE AI	WHAT TO CHECK BY HAND
01 · The Approval Pileup	What currently waits on your approval, your roster, and rough comfort levels by decision type.	Every dollar and authority threshold. These are the playbook — don't ship a robot's guess.

FIRE	WHAT TO TELL THE AI	WHAT TO CHECK BY HAND
02 · The Information Black Hole	The questions you answer most, where knowledge currently lives, who asks whom.	The answers themselves. Verify every extracted fact before it becomes the official record.
03 · The Delegation Boomerang	What you've handed off that came back, to whom, and what was missing when it returned.	The Standard and Judgment layers of the rebuilt handoff. Those are yours to define.
04 · The Rework Loop	The work you keep fixing, real examples of great and unacceptable if you have them.	The examples in the standard. Real ones from your business, never AI-invented ones.
05 · The Broken Handoff	The handoff that keeps dropping, the roles on each side, where it breaks.	The trigger definitions — the exact conditions that move work from one desk to the next.
06 · The Founder Trust Trap	Which clients or accounts insist on you, who could credibly take each relationship.	Every client-facing word, and the pace of the trust transfer. Clients feel a rushed one.
07 · The Revenue Drop-Off	Your sales process as it actually runs, what only happens when you sell, your offer details.	Pricing language, claims, and anything a prospect will read or hear. All yours.
08 · The Accountability Vacuum	What slides when you're not watching, your last five interventions and what each was about.	The values-as-behaviors language. The culture code has to sound like you, not like HR.
09 · The No-Backup-Plan Breakdown	Your tools, who holds which logins and accesses, what physically stops if you vanish.	The access list itself — verify against real accounts, and never paste live credentials into any AI.

CUSTOMIZE OR BUILD FRESH?

The honest decision rule: read the classic playbook's Scene section. If it describes your fire — not perfectly, but recognizably — customize, because eighty percent of the structure transfers and you'll be live in a day. If you find yourself explaining to the GPT why your situation is "kind of like the Rework Loop except actually not," stop — that sentence means you're forcing a template onto

a different fire, and forced templates produce playbooks nobody follows because they solve a neighboring problem. Run the method from scratch instead; it costs you one extra session and buys you a playbook aimed at the thing that's actually burning. The middle case — a genuinely custom fire that shares a dependency type with one of the nine — gets the hybrid: run the full method, but hand the GPT the matching classic playbook as reference material so its prevention structures start from proven ground. That's the move the example book uses in all three of its builds, and it's usually the best of both.

> **ONE WARNING**
>
> Customization is fast, which makes it tempting to customize all nine in a weekend and call your business systematized. Resist. A customized playbook still needs step six — rollout, relight-watching, review — and rollouts compete for your team's attention. One live and stable beats nine customized and sitting in a folder.

10 THE TWO CHECKS

Every AI-drafted playbook passes two checks before it touches your team. No exceptions, including the playbooks you're proudest of.

These exist because AI drafts fail in two predictable ways. First: trained on a world of documents where founders approve everything, it quietly writes you back into the process you're trying to exit. Second: it assigns steps without finishing the thought — owners with no backup, failure modes with no path. Both failure modes are invisible on a skim and expensive in production. So you check for them explicitly, every time.

CHECK ONE — THE FOUNDER INVOLVEMENT CHECK

Question on the table: *does anything in this playbook still route through me that shouldn't?* Read the full draft — response and prevention — and run this:

- [ ] I've highlighted every step, review, approval, and escalation where I appear, and counted them.
- [ ] For each appearance, I can say why it must be me — a genuine legal, financial, or strategic reason, not comfort, habit, or the AI being deferential.
- [ ] No step requires my availability for the process to keep moving — nothing waits on my calendar, my inbox, or my mood.
- [ ] Any approval I've kept has a deadline and a default — if I don't respond by the defined time, the plan says what happens, and it isn't "everything stops."
- [ ] I am not the trigger. The process starts from a written condition or calendar event, not from me noticing and saying "go."
- [ ] The Judgment layer gives the owner real gray-area authority — the plan doesn't quietly define every gray area as "ask the founder."

Anything that fails goes back to the GPT with one instruction: remove me from this step and rebuild it so it works without me. If it genuinely can't be rebuilt that way, fine — but that's a decision you made on purpose, on the record, not a default the draft slipped past you.

CHECK TWO — THE ESCALATION CHECK

Question on the table: *does every step have an owner, and does every failure path lead somewhere that isn't me?*

- [ ] Every step in the response plan has exactly one owner — a real person or a real role we currently employ. No "the team." No committees.
- [ ] Every owner has the authority the step requires — nobody's assigned a call they aren't actually allowed to make.
- [ ] Each foreseeable failure — owner out sick, deadline slips, quality misses, tool breaks — has a named next move.
- [ ] Escalations name a person and a condition: what goes up, when, to whom. "Escalate if needed" is not a path; it's a shrug.
- [ ] The first escalation stop is not me. Someone competent stands between every problem and my phone.
- [ ] Anything that does reach me arrives defined — the plan says what I decide and what I'm handed to decide it with, so my involvement is a decision, not a rescue.

Run both checks on paper, pen in hand, not as a mental exercise while scrolling. They take ten minutes together. Every relight you prevent with them would have cost you more.

> ★ FIRE MARSHAL'S NOTE
>
> Outsource check one: hand the draft to its future owner and ask them to highlight every place you appear. They'll catch appearances you've normalized — you read your own name the way you read your own typos, which is to say not at all.

WHAT A FAILED CHECK LOOKS LIKE

A quick anatomy, because failed checks are subtle by design — the draft never says "route everything through the founder." It says things like this:

- "The founder reviews the final version before send" — on a process that runs weekly. That's a standing appointment with your old job.
- "Escalate significant issues to leadership" — where "significant" is undefined and "leadership" is a one-person category. Two vague words hiding one dependency.

- "The owner keeps the founder informed of progress" — harmless-sounding, until "informed" becomes daily check-ins that are approval-seeking with better manners.
- "In urgent cases, contact the founder directly" — with "urgent" left to the judgment of whoever's nervous. Every case is urgent to someone nervous.

Each of these survives a skim and fails the check. And each has the same repair: replace the vague word with a condition, and replace yourself with a name. "Reviews the final version" becomes a defined category — first run of a new format, anything with legal exposure — with everything else shipping on the owner's authority. "Significant issues" becomes two named conditions routed to the ops lead. That's the whole craft of the two checks: hunting soft words that quietly point at you, and hardening them until they point somewhere else.

> **WHY THIS IS THE LINE**
>
> These two checks are the difference between AI making you faster and AI making you busier. A fast draft that reinstalls you as the approver just built you a prettier version of the problem you paid to solve.

11 TEAM ROLLOUT SUPPORT

The playbook changes your business the day your team hears about it — from you.

Rollout runs on three messages, and AI is genuinely good at drafting all three. What it can't supply is the weight. A process change announced well lands as leadership; announced badly — or worse, discovered via a shared doc nobody mentioned — it lands as one more thing the founder is doing to the team instead of with them.

THE THREE MESSAGES

1 THE ROLLOUT MESSAGE — TO THE WHOLE TEAM

What's changing, why now, who owns it, and what to do instead of the old way. Feed the GPT the finished playbook and ask for a short, plain announcement; tell it the tone you want and make it cut the corporate filler. Then edit until it sounds like you on a good day. The one sentence to keep no matter what: the redirect — "questions about X now go to [owner]."

2 THE OWNER BRIEFING — TO THE PERSON TAKING IT ON

Longer and warmer than the announcement. What they now own, the full handoff package, the boundaries of their authority, what escalates and to whom, and what success looks like at thirty days. Ask the GPT to draft it from the playbook, then personalize it — why you chose them, what you've seen in their work. That paragraph can't be delegated to a machine, and it's the paragraph they'll remember.

3 THE CLIENT NOTE — WHERE THE CHANGE TOUCHES CLIENTS

If clients will feel the change — a new point of contact, a new review process — they hear it framed as an upgrade, because it is one: "You now have a dedicated person on this, and here's who they are." Have the AI draft the structure, then rewrite every sentence by hand. Client-facing words are always yours; that rule doesn't bend for rollout week.

THE RULE THAT DOESN'T MOVE

AI drafts the messages. The founder delivers the change — personally, out loud, even when every word was machine-drafted and hand-edited. The rollout message gets said in the meeting, not just posted in the channel. The owner gets briefed face to face, not assigned via task manager. Delegating the work is the whole point of this system; delegating the announcement of the work reads as not caring about it, and your team will price the change accordingly. Thirty minutes of

your voice at rollout buys months of the process running without you. That's the best hourly rate in this entire guide.

THE WEEK AFTER

The announcement is a moment; the first week is the test. Three small disciplines carry it. Watch your own reflexes hardest — the person most likely to violate the new playbook in week one is you, absentmindedly answering a question that now belongs to the owner, because answering it has been muscle memory for years. Check in with the owner once, briefly, midweek — not to inspect the work, but to ask one question: "anything in the package turning out to be wrong or missing?" Early package fixes are cheap; the same fix after a public wobble costs the owner confidence. And say nothing corrective in public for the first two runs. If something needs adjusting, it goes through the owner privately and comes out as their adjustment. The team is watching to see whether this handoff is real. Every signal you send in week one is worth ten in week six.

> ★ FIRE MARSHAL'S NOTE
>
> For the first two weeks after rollout, end each day with a thirty-second audit: which questions reached me today, and whose were they? Anything that belonged to an owner goes on the relight log — even the ones you answered by reflex. Especially those.

> AFTER THE ANNOUNCEMENT
>
> Expect two or three people to bring the old questions to you anyway, out of pure habit. Your only job is the redirect, warmly and every time: "That's Dana's now — she's got it." Answer it yourself even once and you've told the team the playbook is decorative.

12 THE LAST WORD

You bought this because your business kept catching the same fires, and you were the only extinguisher in the building.

Now look at what's actually in your hands. Not nine playbooks — a method. The next fire that starts in your business, whatever it is, however specific to you it is, has a path: capture it, name it, trace it, draft the response, draft the prevention, roll it out, watch it, retire it. AI carries the drafting. You carry the judgment. Your team carries the process. That's the whole architecture, and it doesn't wear out with use — it gets faster every time you run it, because every extraction leaves more of your head on paper where your company can use it.

Fighting fires was a job. Writing codes is an asset. Every playbook you retire is a piece of your business that runs on structure instead of on you — and the founder who builds a dozen of those is running a different company than the one who bought this guide.

A year from now, the difference between having read this and having used it will be visible in your calendar. The founder who read it has a well-organized folder and the same Tuesdays. The founder who used it has a shelf of retired fires, a team that captures incidents without being asked, and whole categories of interruption that simply don't reach her anymore — not because she got tougher about boundaries, but because the structure underneath stopped producing the interruptions. Same business, different wiring.

Start with one. This week's fire, whatever it is. The worksheet is in your order, the GPT is waiting, and the Rapid Implementation Guide will walk you through the first build in seven days.

And when you're ready to go past playbooks — to a business that runs without you at the structural level, every system built to completion — that's the work Luxe Business Backend™ does with founders directly.

TIFFANY LOPEZ

Business Systems Architect · Luxe Business Backend™