LUXE BUSINESS BACKEND™ · FOUNDER FIRE CODES™

THE NINE-DEPENDENCY PREVENTION & SOLUTION GUIDE

The nine ways a business gets wired through its founder — and what replaces you in each one.

BEFORE YOU READ

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 and your team.

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

Any examples in this guide are composite scenarios built to teach, not real client stories. Nothing here is legal, financial, or HR advice.

WHAT'S INSIDE

00 How to Use This Guide

Dependency is nine diseases, not one — and it's typed per process, not per founder.

01 Direction & Decision Dependency

"The team checks with me before moving."

02 Revenue Dependency

"It only works when I'm selling or being the face."

03 Knowledge Dependency

"I'm the only one who knows."

04 Execution & Delivery Dependency

"It doesn't move unless I push."

05 Quality & Risk Dependency

"I have to review, fix, catch."

06 Coordination Dependency

"I'm the one connecting people."

07 Relationship & Brand Dependency

"The client only trusts me."

tyle="color:inherit;text-decoration:none;">08 Culture Dependency

"It's done my way only when I'm watching."

09 Systems & Infrastructure Dependency

"If I vanished for a week, things physically couldn't happen."

10 The Master Table

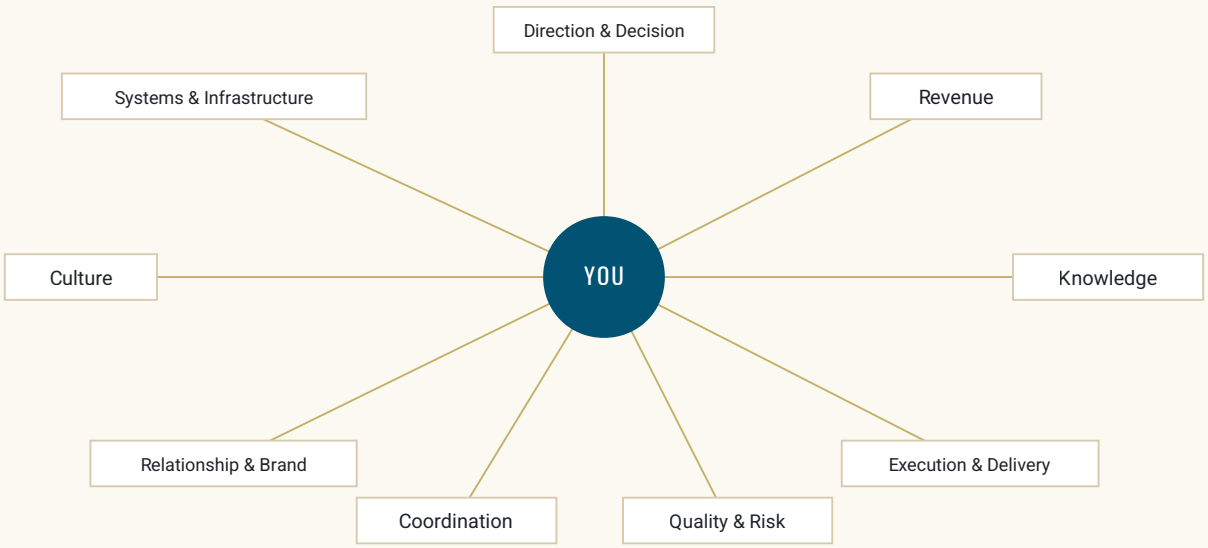
All nine on one page: how each sounds, and what replaces you.

00 HOW TO USE THIS GUIDE

"Everything depends on me" is a complaint. It is not a diagnosis.

Founder dependency isn't one disease. It's nine, and each one has its own symptom signature, its own structural cause, and its own cure. Treat a Knowledge problem with an accountability fix and nothing changes. Treat a Quality problem by hiring a better person and the new person fails the same way the last one did — because the standards still live in your head. The whole point of this taxonomy is to stop guessing.

THE WIRING MAP — NINE WAYS A BUSINESS ROUTES THROUGH ITS FOUNDER



Every recurring fire runs on one or two of these wires. The trace finds which; the chapters that follow cut them — one structural cure per wire.

Two rules govern everything in this guide.

RULE ONE: IT'S TYPED PER PROCESS, NOT PER FOUNDER

You are not "a Knowledge-dependent founder." Nobody is. Dependency attaches to processes, not personalities. The same founder's client onboarding can be a Knowledge problem while her sales conversations are a Relationship problem and her content calendar is an Execution problem. Diagnose the process in front of you, fresh, every time — and don't let last month's finding color this month's trace.

One process also rarely carries just one type. Most carry a primary and a secondary, sometimes a third. Onboarding that's primarily Knowledge ("they ask me where everything lives") often carries Quality underneath ("and when they don't ask, I end up fixing it"). The specific combination is the treatment plan — which structures you build, and in what order.

RULE TWO: THE CAUSE IS STRUCTURAL, NEVER A CHARACTER FLAW

In every chapter of this guide, the "why it exists" section points at missing structure — absent decision architecture, unwritten standards, un-extracted knowledge, a trust transfer nobody designed. Not once does it point at a weak team, a controlling founder, or anybody's personality. That's not politeness; it's accuracy. Structures can be built in weeks. People-fixing projects run forever and mostly fail. If your diagnosis ends in a character judgment, you haven't found the root cause yet. Keep digging.

WHERE THIS FITS IN THE METHOD

This guide is the deep reference behind step three of the Fire Codes Method™ — Trace the Dependency. The method guide gives you the symptom test, one line per type, enough to trace most fires. Come here when the trace is ambiguous, when you want the full anatomy of a type before building prevention, or when you're feeding the Fire Codes GPT and want to check its trace against the source. Each chapter runs the same anatomy: how it sounds, the fires it starts, why it exists, what prevents it from forming, and what replaces you once it's burning. The last two sections are the ones your prevention plans will lean on hardest.

WHAT EACH CHAPTER HANDS YOU

Every chapter runs the same six sections, and each one maps to a different moment in your work. *How It Sounds* is for tracing — the symptom sentence plus the quieter tells that confirm or break a diagnosis. *The Fires It Starts* connects the type to the named fires from the playbook library and describes what custom fires of this type look like, so you can recognize your own weird ones. *Why It Exists* is for your head: the structural cause, stated plainly, because founders fix faster once they stop treating the dependency as a personal failing — theirs or their team's. *Catching It Early* is for the processes that aren't on fire yet — the signals that a dependency is forming and the cheap move that stops it. *The Prevention Structure* tells you what a business where this type can't take hold has built. And *The Solution Moves* is the working section: the sequence that replaces the founder once the type is live, in the order that actually works. When you're building a prevention plan with the Fire Codes GPT, the last two sections are your fact-check — whatever the AI

proposes should rhyme with what's written there, and where it doesn't, trust the chapter and make the AI explain itself.

> **A NOTE ON READING ORDER**
>
> Don't read this cover to cover like a novel. Trace your fire first, then read your chapter — the one whose symptom sentence made you wince. That wince is data.

01 DIRECTION & DECISION DEPENDENCY — NOTHING MOVES UNTIL YOU'VE WEIGHED IN

HOW IT SOUNDS

The signature sentence: **"The team checks with me before deciding or moving forward."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

SLOW BURN | STEADY | **FAST** | FIVE ALARM

- "Quick question — are we okay to…" arrives in your inbox several times a day, and the answer is almost always yes.
- Work pauses politely while people wait for your reply, then sprints to make up the lost time.
- Your team can execute anything you've specified and almost nothing you haven't.
- You come back from three days away to a stack of decisions that were all safe to make without you.
- Nobody ever gets in trouble for asking. People have gotten in trouble for guessing.

THE FIRES IT STARTS

This is the type underneath **Fire 01, The Approval Pileup** — finished work queuing at your desk for sign-off. It also feeds **Fire 03, The Delegation Boomerang**, because a handoff without decision authority isn't a handoff; it's a task on a leash. Custom fires of this type all share one shape: a queue with your name on it. The discount that couldn't be offered until you answered, the project that idled all Thursday waiting on a yes, the vendor decision that aged two weeks in your inbox. If the fire's story includes the word "waiting," look here first.

WHY IT EXISTS

Almost never a weak team. It's absent decision architecture: nobody ever defined what the team can decide alone, what the thresholds are, or what principles govern the gray areas. In that vacuum, escalating everything is the team's only safe move — guessing wrong risks embarrassment or rework, and asking costs only your time, which the structure has silently priced at zero. Smart people respond to that math exactly the way your team is responding to it.

★ FIRE MARSHAL'S NOTE

Keep a yes-log for one week: every approval where your answer was yes with zero changes goes on the list. That list is your first thresholds memo, already written — you're just publishing decisions you've been making on autopilot.

CATCHING IT EARLY

This type announces itself long before the approval queue gets embarrassing, and the earliest signal is a pleasant one: your team starts describing you as "so responsive." Responsiveness feels like leadership, but read it structurally — a team that praises your reply speed is a team whose work is gated on your replies. Other early tells: your first hire's questions never tapered off after onboarding; new team members learn within a month that checking first is the culture; and you've started answering some questions with "good instinct, go with it" — which means the instinct exists and the permission structure doesn't. The cheapest moment to fix this is right there: the first time you catch yourself approving something you'd have approved a hundred times out of a hundred, stop and write the rule instead of the reply. One written threshold at month six prevents a hundred-item queue at year three. Watch your own language too — if you hear yourself saying "just run it by me first, it'll only take a second," understand what you're actually building: a business where everything takes a second of you, forever.

THE PREVENTION STRUCTURE

Decision architecture, written before it's needed: decision rights by role, so every recurring decision type has a named owner. Spending and impact thresholds, so "under $X and reversible" never needs to ask. Escalation criteria that define what actually merits your judgment. And written operating principles — the handful of rules that let people decide gray areas the way you would, without you in the room. Businesses with this architecture simply don't grow approval queues; there's nowhere for the queue to form.

THE SOLUTION MOVES

1 INVENTORY A WEEK OF ASKS

Log every decision that came to you for seven days. Sort into three piles: they could have decided (usually most of it), they needed a rule that didn't exist, and it genuinely needed you.

2 PUBLISH DECISION RIGHTS AND THRESHOLDS

Turn pile one into written authority: who decides what, up to what dollar amount, in which situations — say it out loud to the team, and mean it.

3 WRITE THE OPERATING PRINCIPLES

Turn pile two into rules. Your last twenty decisions have a pattern in them; extract it, write it as principles the team can apply to cases you've never seen.

4 HOLD THE LINE AT THE DOOR

When a delegated decision shows up anyway, redirect it — warmly, every time: "That's yours. What would you do?" Then back their call publicly, especially the imperfect ones. One overruled decision teaches the team more about your real thresholds than any document.

> **Your team isn't asking permission because they can't decide. They're asking because deciding was never made safe.**
>
> FOUNDER FIRE CODES™

02 REVENUE DEPENDENCY — MONEY ONLY MOVES WHEN YOU DO

HOW IT SOUNDS

The signature sentence: **"It only works when I'm the one selling, launching, or being the face."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

- Every meaningful sale in the last year closed on a call you were on.
- Revenue dips on cue whenever you go heads-down on delivery, and recovers when you resurface.
- Someone else has tried taking sales calls, and the close rate told you exactly how that went.
- Launches happen when you have the energy to be the engine, so the calendar bends around your capacity.
- Prospects say some version of "I want to work with *you*" — and the offer doesn't quite work without that sentence.

THE FIRES IT STARTS

This is the type underneath **Fire 07, The Revenue Drop-Off** — the dip that tracks your attention like a shadow. It's also the usual secondary inside **Fire 06, The Founder Trust Trap**, because when clients only trust you, they also only buy from you. Custom fires of this type look like a calendar problem that's secretly a structure problem: the renewal that only closes when you personally call, the launch that can't run in a quarter you're traveling, the flagship offer nobody else can explain at a dinner party, the pipeline that empties whenever you stop posting.

WHY IT EXISTS

The sales process was never separated from the salesperson. Your instincts — what you say when a prospect hesitates, which stories you tell for which objection, how you frame the price — produced revenue from day one, so nobody ever wrote them down; the business just kept renting them from your calendar. Meanwhile the offer's messaging grew up assuming a live founder attached: it converts because you explain it, which means the explanation is the product's missing user manual. None of that is a team failure. It's an extraction that never happened.

CATCHING IT EARLY

The early warnings live in your calendar and your metrics, months before the dependency feels like a problem. Watch for the correlation first: put your revenue by week next to your visibility by week — the launches you fronted, the calls you took, the content you personally posted — and see how tightly they track. A healthy business shows some slack in that line; a Revenue-dependent one moves in lockstep. Second signal: the sentence "we should really get someone else doing sales" has been said in your business more than twice, and each time it dissolved into "after this launch." Third: your best delivery quarter was your worst pipeline quarter, or vice versa — the seesaw that means one engine is powering two machines. The early move is small and specific: before you need it, start recording your own sales conversations and stockpiling the raw material. Even if nobody else sells for another year, the transcripts turn a future six-month extraction project into a six-week one. The founders who unwind this dependency painfully are the ones who started documenting the day they stopped being able to keep up. Start while you still can.

THE PREVENTION STRUCTURE

A documented sales process that exists apart from any one person — stages, follow-up cadence, proposal standards. Offer messaging that converts without a live founder explanation: the page, the deck, and the email sequence carry the full argument. Case-study and objection libraries extracted from your actual instincts, so your best answers are an asset, not a performance. And at least one non-founder sales pathway — a person, a funnel, or both — producing revenue on purpose, so the day the business needs it, it isn't an experiment.

THE SOLUTION MOVES

1 EXTRACT THE INSTINCTS

Record your next five sales conversations (with consent). Pull out what you actually say: the discovery questions, the objection answers, the price framing, the stories. That transcript pile is the raw sales system.

2 BUILD THE LIBRARIES

Turn the extraction into an objection library and a case-study library — your best answer to each recurring hesitation, written, with the real examples that make it land.

3 FIX THE MESSAGING GAP

Rewrite the offer's core assets until a warm prospect could reach "yes" without meeting you. The test: what do you always end up explaining live? That explanation belongs on the page.

4 OPEN ONE NON-FOUNDER PATHWAY

Pick one: a trained closer running your documented process, or a self-serve path for the offer that doesn't need a call. Start it small, measure it honestly, and let it be worse than you at first — a pathway at 70% of your close rate that runs without you beats 100% that only runs on your calendar.

> **THE HONEST TIMELINE**
>
> Revenue is usually the slowest dependency to unwind, because trust and skill both transfer through reps, not memos. Start before you're desperate. The worst time to build a non-founder sales pathway is the quarter you finally burn out on selling.

03 KNOWLEDGE DEPENDENCY — YOU ARE THE COMPANY'S ONLY SEARCH ENGINE

HOW IT SOUNDS

The signature sentence: **"They ask me how to do it, or I'm the only one who knows the context."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

SLOW BURN | STEADY | FAST | FIVE ALARM

- The same five questions arrive on a loop — different asker, same answer, forever.
- "Where does the…" and "what's our…" messages follow you onto vacation.
- Client history lives in your memory: what was promised, what was tried, why the thing is the way it is.
- Work stalls politely while people wait for an answer only you have.
- Training a new hire means weeks of your calendar, because you are the onboarding materials.

THE FIRES IT STARTS

This is the type underneath **Fire 02, The Information Black Hole**. It also quietly feeds **Fire 03, The Delegation Boomerang** — handed-off work comes back wrong when the context needed to do it right never left your head — and it's a frequent primary in launch and delivery fires of every flavor. Custom fires of this type have a tell: the incident story contains a question mark. Somebody asked you something, or should have and guessed instead. The night-before rewrite, the wrong-version deliverable, the "I didn't know we'd promised that" client call — all Knowledge fires wearing different outfits.

WHY IT EXISTS

The knowledge was never extracted — and the standard advice for fixing that is designed to fail. "Document your processes" assigns homework to the one person with no capacity for homework, which is why the wiki is three years stale and the SOP folder has four documents and a readme. Meanwhile every answer you give in chat solves exactly one instance and creates a precedent for asking again. The cause isn't a team that won't learn or a founder who won't write. The structure for moving knowledge out of your head as a byproduct of normal work simply doesn't exist yet.

CATCHING IT EARLY

The earliest signal is repetition you've stopped noticing. Scroll your own sent messages from the last month and look for answers you've typed more than twice — most founders find five to ten on the first pass and feel a little sick about it. Other tells before this becomes a fire: your out-of-office days generate a backlog of questions rather than a backlog of work; the phrase "quick question" has become the most common message you receive; and when someone new joins, an existing team member says "you'll want to ask her directly" about anything with history in it. The sharpest early indicator is what happens to work in your absence — not whether it stops (that's Execution) but whether it proceeds on guesses that you later have to unwind. Wrong-guess rework is Knowledge dependency's calling card. The early move costs almost nothing: start answering recurring questions in a shared place instead of a private one. Same answer, different address. You're not building a knowledge base yet — you're just refusing to keep paying full price for answers you've already written.

> ★ FIRE MARSHAL'S NOTE
>
> Title every reference page with the question your team actually asks, word for word — “where does the onboarding deck live” beats “Asset Management Overview.” People search in their own words, and a page nobody finds is a page you’ll re-answer in chat.

THE PREVENTION STRUCTURE

Two pieces. First, extraction over documentation: the founder talks — guided interviews, recorded walkthroughs, quick voice captures in the moment — and someone or something else turns the talking into documents. You edit pages; you never face a blank one. Second, a standing capture system: every time a question reaches you that the written record should have answered, the answer gets captured into that record on the spot. Answer once in chat, it's gone; answer once into the system, it's an asset. Depth beats coverage here — the ten explanations you repeat most, done properly, beat a hundred shallow stubs.

THE SOLUTION MOVES

1 LIST THE TOP TEN RECURRING EXPLANATIONS

Not everything you know — the ten things you re-explain most. Your sent messages and your team's questions from the last month will hand you the list.

2 EXTRACT BY TALKING

For each: record yourself walking through it — a screen-share, a voice memo, an interview where a team member asks and you answer. Thirty minutes of talking per topic, max.

3 TURN TALK INTO REFERENCE

Have AI or a team member draft the reference doc from each recording. You edit for accuracy — editing a wrong draft takes minutes; writing a right one takes evenings.

4 INSTALL THE REDIRECT AND THE CAPTURE LOOP

New rule, said out loud: check the reference first; if it's not there, ask — and the answer goes into the reference the same day. From now on, every question you answer is the last time you answer it.

> **Every answer you give in chat dies in the scroll. Every answer you capture becomes payroll you never pay again.**
>
> FOUNDER FIRE CODES™

04 EXECUTION & DELIVERY DEPENDENCY — YOU ARE THE ENGINE, NOT JUST THE DRIVER

HOW IT SOUNDS

The signature sentence: **"It doesn't move unless I push it."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

SLOW BURN | STEADY | **FAST** | FIVE ALARM

- You open the project board to find everything exactly where it was when you last pushed.
- "Just checking in on this" is a genre of message you send daily.
- Deadlines are real to the team in direct proportion to how recently you mentioned them.
- Things you personally kicked off ship; things you didn't quietly drift.
- Your week off reads, in the metrics, like a company-wide week off.

THE FIRES IT STARTS

This is the type underneath **Fire 03, The Delegation Boomerang** — handed-off work that drifts until you re-grab the wheel — and it's a common secondary in **Fire 05, The Broken Handoff**, where work dies in the seam between two people because nobody but you supplies the momentum across it. Custom fires of this type star you as the human cattle-prod: the client project that idles until you chase it, the podcast that publishes only in weeks you personally herd it out the door, the proposal that sat five days until you asked. If fixing the fire meant you nudging somebody, look here.

WHY IT EXISTS

Ask the diagnostic question: when you push, what are you actually supplying? It's one of three things — urgency (only your attention makes a deadline real), sequence (only you know what comes next), or permission (work waits for your go-ahead between stages). All three are missing structure. Nobody owns the workflow's outcome, so nobody feels the drift. The sequence lives in your head, so every next step needs your input. The accountability loop routes through you, so momentum is something you personally deliver, door to door, like milk.

CATCHING IT EARLY

Count your nudges for one ordinary week — every "just checking in," every "where are we on," every deadline you re-mentioned. Founders who do this expecting a handful routinely find twenty or thirty, and the count is the diagnosis: each nudge is a place where the workflow has no engine of its own. Softer early signals: your project management tool is accurate only in the day after you've asked about everything (the board updates because you looked, not so that you don't have to); "waiting on" has become a permanent status; and team members finish their assigned piece and stop, like relay runners with nobody to hand to, until you point at the next leg. Watch also for the vacation test in miniature — not a week away, just a heads-down day. If a single day of your silence visibly slows throughput, the dependency is already load-bearing. The early move: pick the one workflow you nudge most and give its next occurrence a named owner for the outcome, end to end, before building anything fancier. One workflow that moves without you teaches you — and your team — that movement was never actually your job.

THE PREVENTION STRUCTURE

Workflow ownership: one name per stage of every recurring workflow, accountable for the outcome of that stage, not just the activity. A defined sequence: what follows what, written down, so the next step is knowable without asking. And accountability loops that don't route through you — status made visible by the system (a board, a standing update, an automated report), deadlines with owners who feel them directly, and check-ins that happen on a cadence whether or not you remember to ask. Momentum should be a property of the workflow, not a service you provide to it.

THE SOLUTION MOVES

1 DIAGNOSE YOUR PUSHES

For one week, every time you nudge something, note which of the three you supplied: urgency, sequence, or permission. The tally tells you which structure to build first.

2 PUT ONE NAME ON EACH STAGE

Take the workflow that drifts worst and assign stage owners — a person, not "the team," accountable for the outcome of their stage including its deadline.

3 WRITE THE SEQUENCE DOWN

Map what follows what, with the trigger for each step stated as a fact — "when the contract is signed" — not a vibe — "when we're ready." If you were supplying permission, this is where standing permission gets written in.

4 REROUTE THE ACCOUNTABILITY LOOP

Replace your check-ins with a rhythm that pushes status to you: a weekly written update with a fixed format, a board that's actually current, a rule that blockers get raised by their owner within a day. Then — the hard part — stop chasing. Let the loop catch what it catches, and fix the loop where it doesn't.

> abel">The Chasing Trap
>
> Every chase you send teaches the team the real system: things matter when you chase them. Which means your chasing is holding the old structure up. You can't install accountability loops and keep being one. Pick the loop.

05 QUALITY & RISK DEPENDENCY — YOU ARE THE LAST LINE OF DEFENSE, EVERY TIME

HOW IT SOUNDS

The signature sentence: **"They do the work, but I have to review it, fix it, or catch what they missed."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

SLOW BURN | **STEADY** | FAST | FIVE ALARM

- Nothing ships until it's crossed your desk — and everyone, including you, is fine with that.
- You can't articulate what you check for, but you know it when you see it. So only you can see it.
- The team's finished work is your rough draft.
- Skipping your review feels reckless, because the one time you did, something got through.
- Feedback happens as fixes, not standards — you correct the deliverable and the lesson stays with the deliverable.

THE FIRES IT STARTS

This is the type underneath **Fire 04, The Rework Loop** — the same corrections, applied by you, forever. It runs secondary in **Fire 01, The Approval Pileup** (half of what queues for approval is really queuing for inspection) and shows up inside client-delivery fires of every kind. Custom fires of this type have your red pen in the story: the deck you rebuilt at midnight before the client call, the email you caught with the wrong price, the deliverable that's technically done but you can't bring yourself to send. Anywhere "done" and "done right" are separated by your personal attention, this is the wire.

WHY IT EXISTS

The standards were never written, so your internal checklist is the company's only quality system. The team genuinely cannot see what you see — not because they're careless, but because "what good looks like" has never existed outside your head. Every review you perform confirms the arrangement: you catch things, which proves you need to review, which means standards never get written, which means you keep catching things. The loop is self-sealing, and the longer it runs, the

more your eye feels irreplaceable — when what's actually irreplaceable is the unwritten checklist behind it.

CATCHING IT EARLY

The first warning is the phrase "I'll just take a quick look before it goes out" becoming standard for a category of work — note the moment a spot-check quietly becomes a checkpoint. From there the progression is predictable: your review time grows with headcount instead of shrinking (each new hire adds work to your inspection queue, which is backwards); praise in your company starts sounding like "she barely needs review," which reveals that review is the default state; and your team begins pre-flinching — submitting work with "not sure if this is what you wanted" attached, because without written standards, every submission is a guess about your taste. The most useful early test: pick a recent piece you corrected and ask the person who made it to explain why your version is better. If they can — the standard exists and just needs writing down; do it this week. If they can't — the standard exists only as your reflexes, and every day you wait, the extraction gets bigger. Either way, the answer is the same afternoon of work now versus a permanent inspection job later. Founders always think they'll write the standards "when things calm down." Things don't. Write them while they're loud.

> ★ FIRE MARSHAL'S NOTE
>
> Sort your last ten catches into two piles: taste and risk. Risk catches get written into the standard first. Taste catches get an experiment — let one ship uncorrected and see if anyone else notices. If nobody does, it was never a standard; it was a preference riding along for free.

THE PREVENTION STRUCTURE

Written standards with real examples of both "great" and "unacceptable" — and the examples are not optional, because standards without examples are just vibes. Review checkpoints that don't route through you: self-checks against the standard first, then peer review, with your eye reserved for defined high-stakes categories you chose on purpose. And risk flags that surface problems automatically — the conditions that used to trip your instincts, written down as signals anyone can watch for, so catching problems stops being a founder sense and starts being a system feature.

THE SOLUTION MOVES

1 EXTRACT THE CHECKLIST YOU ALREADY RUN

Next three reviews, narrate yourself: what do you look at first, what makes you wince, what would you never let ship? Record it. That narration is the standard, in raw form.

2 WRITE THE STANDARD WITH EXAMPLES

Turn the narration into a written standard per work type — and attach real examples: two pieces that were great and why, two that were unacceptable and why. Pull them from actual past work.

3 RESTRUCTURE THE CHECKPOINTS

New flow: creator self-checks against the standard, a designated reviewer (not you) checks next, and your review shrinks to a defined category — new client types, legal exposure, first runs of a new format. Written down, so "does this need Tiffany?" has an answer that isn't "when in doubt, yes."

4 FEED EVERY CATCH BACK INTO THE STANDARD

When something does slip through, the fix goes two places: the deliverable, and the standard. A catch that only fixes the deliverable will need catching again. A catch that upgrades the standard is the last of its kind.

> **You don't have higher standards than your team. You have unwritten ones.**
>
> FOUNDER FIRE CODES™

06 COORDINATION DEPENDENCY — YOU ARE THE HUMAN ROUTER

HOW IT SOUNDS

The signature sentence: **"I'm the one connecting people and keeping everyone on the same page."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

- Information travels between team members mostly by way of you: "Did anyone tell Marcus?" No. You tell Marcus.
- You're cc'd on everything, because being left off once caused a problem nobody wants to repeat.
- Two people who work three feet apart in the org chart learn about each other's work in your meetings.
- Project status lives in your head, assembled from fragments only you receive.
- When you're out, collaboration doesn't break loudly — it just quietly stops happening across team lines.

THE FIRES IT STARTS

This is the type underneath **Fire 05, The Broken Handoff** — work dropped in the seam between two people, because the seam was you and you were busy. It feeds **Fire 03, The Delegation Boomerang** too: delegated work that needed input from a third person stalls, because connecting those two was silently your job. Custom fires of this type are relay fires: the client update that never reached delivery, the design that got built off the old brief, the two team members solving the same problem separately for a week. The tell is the phrase "I thought someone told them."

WHY IT EXISTS

The company has no operating rhythm, so you became one. There's no meeting cadence owned by anyone else, no reporting that pushes visibility outward, no defined handoff protocol between functions, no communication norms about what gets shared where. In that vacuum, information routes through the one node connected to everybody: you. And every time you relay something

successfully, the structure gets less likely to be built — why install plumbing when the founder carries water this reliably?

CATCHING IT EARLY

This type hides inside virtues — "I like to stay close to the work," "I just want everyone aligned" — so catch it with mechanics, not motives. Early tells: your meetings are the only place certain people exchange information, and skipping one creates a visible information gap that week. Your inbox contains messages that are really for someone else — sent to you because you're the reliable switchboard. You've said "let me connect you two" about people who've worked together for a year. And status questions come to you by default: a client asks where their project stands, and you have to go collect the answer from three people, because the answer doesn't live anywhere — it lives distributed, and you're the aggregator. The earliest structural move is embarrassingly simple: for one recurring relay you perform, create the direct channel and step out of it. A shared thread between the two people, a standing fifteen-minute handoff between the two functions, a status doc the client-facing person can read without asking you. Every relay you retire is a piece of organizational plumbing installed. Keep the water-carrying and you'll eventually be doing it at a scale that has you in meetings all day relaying what the meetings said.

THE PREVENTION STRUCTURE

An operating rhythm that runs without you: a meeting cadence with non-founder owners — someone else runs the weekly, someone else owns the agenda. Reporting that pushes visibility to you instead of being chased — a fixed-format status update, on a fixed day, owned by its writers. Handoff protocols between functions: when work crosses from sales to delivery, from design to build, the package and the trigger are defined, so the seam holds on its own. And communication norms — what goes in the channel, what goes in the doc, what merits a meeting — so information has addresses that aren't your inbox.

THE SOLUTION MOVES

1 MAP A WEEK OF RELAYS

Every time you connect two people or forward information, log it: from whom, to whom, about what. The map shows exactly which structures are missing — recurring relays are unbuilt pipes.

2 INSTALL THE RHYTHM, AND HAND IT OVER

Set the cadence — a weekly team sync, a monthly review — and give each meeting a non-founder owner from day one. If you run it, it's another you-shaped structure with a calendar invite.

3 DEFINE THE SEAMS

For each recurring handoff between functions, write the protocol: what's in the package, what triggers the pass, who confirms receipt. The handoffs your relay map flagged most go first.

4 FLIP THE REPORTING DIRECTION

Replace your fragment-gathering with a standing update that comes to you — same format, same day, owned by the team. Then stop relaying: when "did anyone tell Marcus" surfaces, the answer becomes "post it in the channel," not "I'll tell him." The pipes only fill if the water stops walking itself over.

> **THE CC DETOX**
>
> Being copied on everything feels like oversight, but it makes you the de facto router — whoever sees everything gets asked to relay everything. When the reporting rhythm is live, come off the cc lines on purpose. Visibility should arrive assembled, not raw.

♡ 07 RELATIONSHIP & BRAND DEPENDENCY — THEY TRUST YOU, NOT THE COMPANY

HOW IT SOUNDS

The signature sentence: **"The client only trusts me with it."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

SLOW BURN | STEADY | FAST | FIVE ALARM

- Clients email you directly, even for things your team handles better and faster.
- "Will you personally be involved?" comes up on sales calls — and a yes is quietly load-bearing in the close.
- Introducing a team member to a client feels like a downgrade you have to sell.
- The audience follows you, not the brand — the company account is a ghost town with a logo.
- Renewals, referrals, and rescues all run through your personal relationship with the client.

THE FIRES IT STARTS

This is the type underneath **Fire 06, The Founder Trust Trap** — every meaningful client relationship wired to you personally. It's the usual primary behind renewal fires and escalation fires ("the client asked for you"), and a frequent partner to Revenue Dependency, since trust that can't transfer can't sell without you either. Custom fires of this type share a scene: a capable team member reaches out, and the client goes around them, back to you — not rudely, just certainly. The fire isn't the client's preference. It's that the business has no mechanism for changing it.

WHY IT EXISTS

Trust formed where trust was built, and it was only ever built with you. Clients met you first, were sold by you, were rescued by you — and the team entered the story late, framed as helpers rather than experts. Trust transfers through exposure and demonstrated competence, and the structure never gave clients either: team members were introduced as assistants after the relationship had set, or not introduced at all. Meanwhile the brand invested everything in founder celebrity — your face, your name, your voice — so the company itself accumulated no credibility of its own. Nobody designed a trust transfer, so no trust transferred.

CATCHING IT EARLY

The early signals arrive as compliments, which is why founders miss them. "You're the reason we signed" feels wonderful in year one and reads differently in year four, when it's still true for every account. Watch for: clients replying to your team's emails by emailing you; meeting attendance that collapses when you can't make it ("let's just reschedule for when Tiffany's back"); prospects who name you, personally, in the contract conversation; and the subtle one — your team pre-clearing things with clients by invoking you ("Tiffany reviewed this herself"), which borrows your trust instead of building their own. The early structural move is about sequencing, not stepping back: change how team members enter the story. From the next new client onward, the expert who'll own the relationship appears in the sales process, is introduced with their credentials and their wins, and takes a visible piece of the first delivery. Trust built from day one never has to be transferred — which is a far easier project than moving trust that's already set. For existing accounts, start with the next renewal cycle rather than mid-relationship; trust moves best at natural chapter breaks. And keep being trusted — the goal was never less trust in you. It's trust with more than one place to live.

THE PREVENTION STRUCTURE

Structured trust transfer, built into the client journey from the first touch: team members introduced as experts early — named, credentialed, given real moments to demonstrate competence in front of the client — rather than as assistants late. Defined relationship ownership per account, so "who does this client call first?" has an answer that was chosen, not defaulted. And brand assets that build company credibility rather than only founder celebrity: case studies that star the team and the method, a company voice clients recognize, proof that the results come from the machine and not just the founder's touch.

THE SOLUTION MOVES

1 MAP THE TRUST

List your top relationships — clients, partners, key audience segments — and mark whose trust each one actually holds: you, a team member, or the company. The all-you column is the exposure list.

2 INTRODUCE EXPERTS, NOT ASSISTANTS

For each transferring relationship, pick the receiver and stage the introduction properly: you frame them by expertise and wins, in a real working moment — not "my assistant will follow up," but "Maria leads this for us; she's who I'd want on it."

3 ENGINEER DEMONSTRATION MOMENTS

Trust moves on evidence. Put the receiver where the client can watch them be excellent — leading part of the call, delivering the win, catching the issue — while you visibly defer to them on their ground. Then step back gradually, not abruptly; every clean interaction without you transfers a little more weight.

4 BUILD COMPANY-LEVEL PROOF

Shift the asset mix: case studies and content that credit the team and the process, client-facing materials in the company's voice, wins told without you in the frame. The goal isn't hiding the founder — it's a brand that would still be credible if you took a quarter off.

Trust transfers through exposure and demonstrated competence. Never through announcements.

FOUNDER FIRE CODES™

08 CULTURE DEPENDENCY — THE STANDARD IS YOUR PRESENCE

HOW IT SOUNDS

The signature sentence: **"It gets done the way I'd do it only when I'm watching."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

SLOW BURN | **STEADY** | FAST | FIVE ALARM

- Client care, follow-through, and polish all run noticeably warmer when you're in the room.
- The team can't state the company's values without guessing — but they can all predict what would bother you.
- Corners get cut in your absence that nobody would dream of cutting in your presence.
- New hires learn "how we do things" by watching you react, one incident at a time.
- Accountability is a mood: things tighten after you've noticed something, then drift until you notice again.

THE FIRES IT STARTS

This is the type underneath **Fire 08, The Accountability Vacuum** — standards that hold by supervision instead of by agreement. It runs secondary in **Fire 04, The Rework Loop** (some rework isn't missing skill, it's missing care) and turns up inside client-experience fires where the service was technically delivered and noticeably colder than it would've been from you. Custom fires of this type share the watching problem: the tone in the client thread that made you cringe, the "good enough" that went out the door on your day off, the way the whole operation subtly relaxes when your calendar says travel.

WHY IT EXISTS

The culture is real — it's just unwritten. "How we do things here" exists only as how you handle things, enforced by presence and mood rather than codified values. Nobody ever translated your instincts into named behaviors, so the team's only way to learn the culture is to watch you react — which works exactly as far as your line of sight. This is not a commitment problem. People hold standards they've agreed to; they can't hold standards they have to infer, and inference stops when

you leave the room. Here's the extraction insight: the pattern in your last five interventions *is* the unwritten culture. It's been writable this whole time.

CATCHING IT EARLY

The early tells are small asymmetries between your presence and your absence. Client messages sent on your office days are warmer than ones sent when you're traveling. The Friday ship rate dips when your calendar says off-site. A new hire asks a veteran "is this okay to send?" and gets "she'd probably want it friendlier" — your standards being transmitted as folklore, person to person, with the accuracy folklore has. Sharpest of all: you return from time away and find nothing broken but several things slightly off — not failures, just a few degrees of drift in tone, care, and finish. That drift is culture running on inference, and it always drifts toward whatever's easiest. Catch it now, because this dependency compounds with headcount: at five people, everyone learned the culture directly from you; at fifteen, most of the team learns it secondhand from people who learned it by watching. The early move is the extraction this chapter's remedy runs on, done small: next time you correct *how* something was done, write one sentence about the value underneath the correction and put it somewhere shared. Ten corrections in, you'll have the first draft of a culture code — assembled from real moments, in your actual voice, at a cost of one sentence per incident.

THE PREVENTION STRUCTURE

A culture code: values expressed as behaviors, not posters — "we reply to clients within one business day even if the answer is 'working on it'" rather than "excellence." Leadership principles that tell the team how decisions get weighed here, so judgment has a company shape and not just a founder shape. And accountability norms that hold whether or not you're in the room: peers expected to flag misses to peers, standards reviewed on a cadence, praise and correction tied to the written code instead of to your mood. Culture by agreement, not by surveillance.

THE SOLUTION MOVES

1 MINE YOUR LAST FIVE INTERVENTIONS

Write down the last five times you stepped in on how something was done — the tone you corrected, the corner you uncut, the standard you re-raised. The pattern across those five is your actual culture. Name it.

2 WRITE VALUES AS BEHAVIORS

Turn the pattern into a short culture code — each value stated as an observable behavior with a real example. If a line couldn't settle a disagreement about how to handle a specific situation, it's a poster line; cut it.

3 MOVE ENFORCEMENT OFF YOUR SHOULDERS

Put the code into hiring, onboarding, and reviews. Set the norm that anyone can flag a miss against the code — peer to peer, no founder required — and back the first person who does it, visibly.

4 AUDIT BY ABSENCE

The test of culture is what happens when you're not watching — so check exactly that. After your next real time away, review what held and what drifted, and treat every drift as a line the code is missing or a norm that isn't landing. Fix the code, not the people.

> **THE REFRAME THAT MATTERS**
>
> "They only do it right when I'm watching" sounds like a loyalty problem. It's a documentation problem. Your team is holding the culture exactly as well as it's been written down — which, until now, was not at all.

09 SYSTEMS & INFRASTRUCTURE DEPENDENCY — THE COMPANY PHYSICALLY RUNS THROUGH YOUR ACCOUNTS

HOW IT SOUNDS

The signature sentence: **"If I were unreachable for a week, things physically couldn't happen."**

COMPOUND SPEED — HOW FAST THIS TYPE SNOWBALLS

SLOW BURN | STEADY | FAST | **FIVE ALARM**

- Key logins live in your head, your password manager, or your phone's authenticator — and nowhere else.
- The billing, the domain, the email platform, the bank: all wired to your personal accounts.
- Team members hit permission walls mid-task and wait for you to click "approve."
- Automations, integrations, and renewals are things only you understand — or only you can even see.
- Your vacation auto-reply is a lie; you'll be approving access requests from the beach.

THE FIRES IT STARTS

This is the type underneath **Fire 09, The No-Backup-Plan Breakdown** — the day something needed doing and the only person who could physically do it was on a plane. It compounds every other fire on this list: a Knowledge fix can't ship if the wiki is on your login; a delegated process stalls at a permission wall you forgot existed. Custom fires of this type are the most literal in the taxonomy: the campaign that couldn't send, the invoice that couldn't go out, the site change that waited four days, the two-factor code sitting on a phone in another time zone.

WHY IT EXISTS

Nobody wired the company through you on purpose — it accumulated. Every tool got set up in the fastest way possible, which meant your email, your card, your admin account, at a size where that was fine. The business grew; the wiring didn't. There was never a role-based access design because there were never really roles — there was you, and then help. This is the most structural type of all nine and the least personal: it's not about trust or judgment or standards. It's plumbing, installed for a one-person company that no longer exists.

CATCHING IT EARLY

This one has the clearest early warnings in the taxonomy, because they're physical. Someone asks for access and waits on you to grant it — that's the system telling you, politely, exactly where it's wired. Other tells: a two-factor prompt reaches your phone for a tool you personally haven't opened in months; a payment fails because the card on file is yours and it expired; a team member says "can you just do it, it's under your login anyway"; and the renewal email for something business-critical lands in your personal inbox, where it is one filter rule away from being missed entirely. Run the smallest version of the vanish test right now, from your chair: could your team send an invoice, publish to the website, and access the client files if your phone were at the bottom of a lake? Any "no" is a single point of failure with your name on it, and each one costs about an hour to fix — a second admin, a transferred ownership, a credential moved to a shared vault. Do the "no"s this month. Unlike every other dependency in this guide, this one doesn't need extraction, trust-building, or behavior change. It needs an afternoon and a checklist, and the only reason it goes unfixed is that it never burns until the exact day it does.

> ★ FIRE MARSHAL'S NOTE
>
> While you're in the vault: print the backup two-factor codes for your three most critical systems, seal them in an envelope, and tell two people where it lives. Ten minutes of paper beats a locked-out launch week — and yes, paper: the one backup that doesn't live behind the login it's rescuing.

THE PREVENTION STRUCTURE

Role and permission architecture: access granted to roles, not moments — each role holding what its work requires, admin rights held by at least two people, nothing critical gated on one human's phone. Tooling ownership: every system in the stack has a named owner who understands it, maintains it, and is not you. And an operating cadence for the infrastructure itself — access reviewed on a schedule, renewals and automations documented and visible, so the plumbing stays mapped as the company changes. Fully building this out is longer-horizon work; what belongs in every playbook now is naming it, closing the dangerous gaps, and putting the rest on a roadmap on purpose.

THE SOLUTION MOVES

1 RUN THE VANISH TEST

Walk your stack asking one question per system: if I vanished for a week, what breaks? Every "that stops" answer goes on a list, ranked by how fast it would hurt.

2 KILL THE SINGLE POINTS OF FAILURE FIRST

For the top of the list: second admins on critical systems, ownership transferred off personal accounts, credentials moved to a shared vault with role-based access, backup on every two-factor. Days of work, most of it, and it removes the scariest fires outright.

3 NAME A TOOLING OWNER PER SYSTEM

One person per platform who owns it — setup, access requests, renewals, "why is this broken" — with the how-it's-wired basics documented as they take it over. Distributing the stack across the team is fine; the point is that none of it defaults to you.

4 ROADMAP THE REST, HONESTLY

Full infrastructure — automation doing the mechanical remembering, metrics wired in, access reviews on cadence — is a build, not a weekend. Name what's left, put it on a dated roadmap, and don't let "we closed the worst gaps" quietly become "we finished." Named and scheduled beats ignored.

> **THE CHEAPEST FIRE INSURANCE YOU'LL EVER BUY**
>
> Most solution moves in this guide take weeks of structure-building. This one starts with an afternoon: a shared vault, a second admin, a backup two-factor. Founders put it off for years because it never burns until the exact day it does.

⊞ 10 THE MASTER TABLE

All nine on one page. Trace with the middle column; build from the right one.

DEPENDENCY	HOW IT SOUNDS	WHAT REPLACES YOU
Direction & Decision	"The team checks with me before deciding or moving forward."	Decision rights by role, spending and impact thresholds, escalation criteria, written operating principles for the gray areas.
Revenue	"It only works when I'm the one selling, launching, or being the face."	A documented sales process, messaging that converts without you live, objection and case-study libraries, at least one non-founder sales pathway.
Knowledge	"They ask me how to do it, or I'm the only one who knows the context."	The top recurring explanations extracted deeply, plus a standing capture system so knowledge keeps leaving your head as a byproduct of work.
Execution & Delivery	"It doesn't move unless I push it."	Workflow ownership — one name per stage — a defined sequence, and accountability loops that don't route through you.
Quality & Risk	"They do the work, but I have to review it, fix it, or catch what they missed."	Written standards with real examples of great and unacceptable, review checkpoints that bypass you, automatic risk flags.
Coordination	"I'm the one connecting people and keeping everyone on the same page."	An operating rhythm: meeting cadence with non-founder owners, push-based reporting, handoff protocols between functions, communication norms.
Relationship & Brand	"The client only trusts me with it."	Structured trust transfer — experts introduced early, demonstration moments engineered — plus brand assets that build company credibility.
Culture	"It gets done the way I'd do it only when I'm watching."	A culture code: values as behaviors, leadership principles, accountability norms that hold without your presence.

DEPENDENCY	HOW IT SOUNDS	WHAT REPLACES YOU
Systems & Infrastructure	"If I were unreachable for a week, things physically couldn't happen."	Role and permission architecture, named tooling owners, an operating cadence — single points of failure closed first, the rest roadmapped.

Three ways to put this table to work. Tape it up — literally or digitally — wherever you do your capturing, because the trace step goes faster when the nine sentences are in view. Hand it to your leadership team, because a process's owner can often hear which sentence a fire is saying before you can; you're too close to your own supplying. And use it as your fact-check when drafting with the Fire Codes GPT: whatever prevention structures the AI proposes for a given type should rhyme with that type's right-hand column. Where a draft proposes something the column doesn't — or skips something it does — make the AI explain the departure. Sometimes it has a good reason grounded in your specifics. More often, the column knows something the draft glossed over.

And one closing thought, since you've now seen all nine in a row. Every column on the right describes something buildable — rights that can be written, knowledge that can be extracted, trust that can be transferred, wiring that can be re-routed. Nothing on that side of the table asks you to become a different person, hire unicorns, or care less about your business. That's the whole argument of this guide in one page: the dependency was never you. It was always the missing structure, and structure is a thing you can start building on a Tuesday.

TIFFANY LOPEZ

Business Systems Architect · Luxe Business Backend™