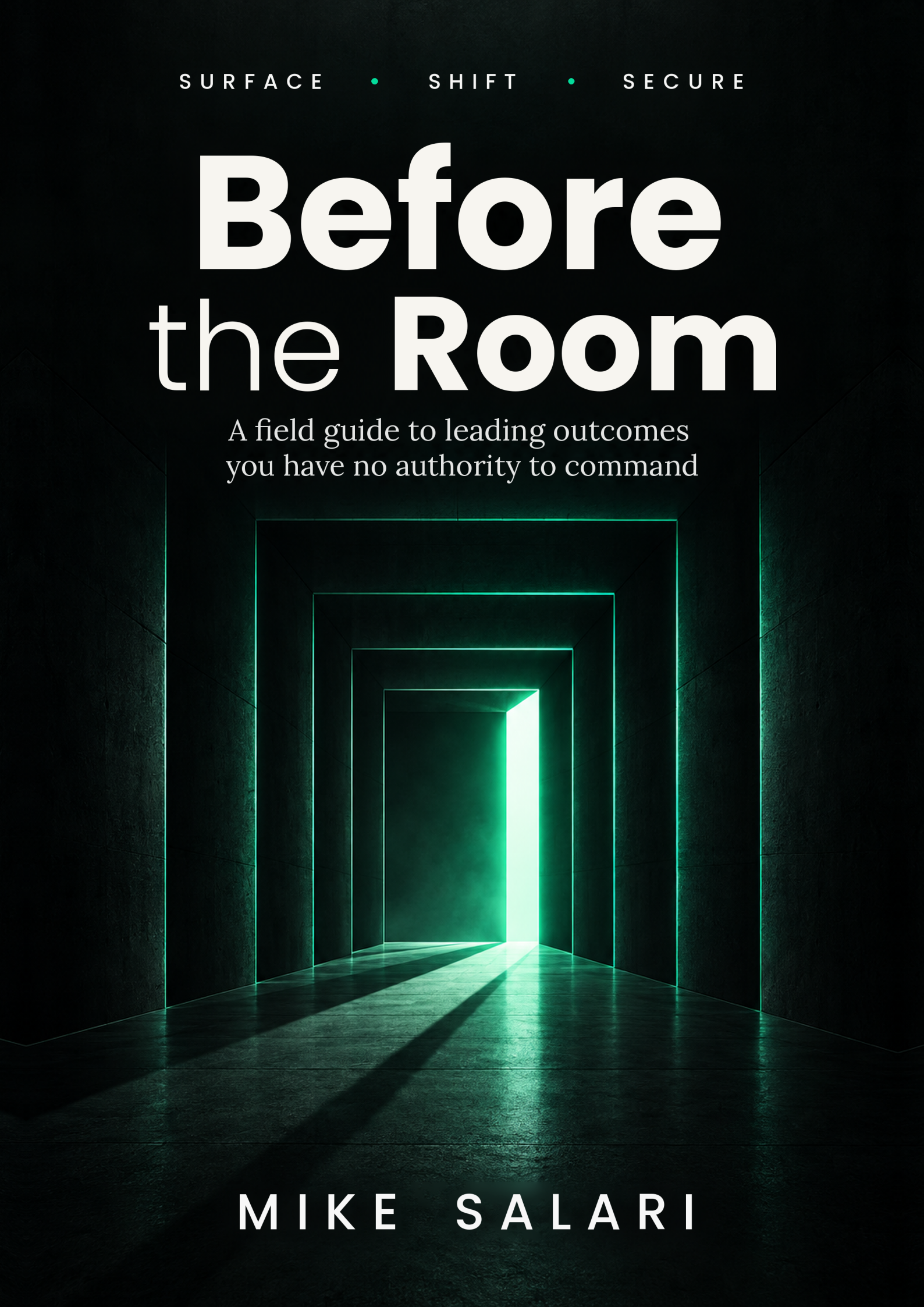


SURFACE • SHIFT • SECURE

Before the Room

A field guide to leading outcomes
you have no authority to command



MIKE SALARI

SURFACE · SHIFT · SECURE

Before the Room

*A field guide to leading outcomes
you have no authority to command*

MIKE SALARI

Salari Press

Before the Room

A field guide to leading outcomes you have no authority to command

Copyright © 2026 Mike Salari. All rights reserved.

No part of this book may be reproduced or distributed in any form without written permission from the author, except for brief quotations in reviews and articles.

Published by Salari Press.

First edition, 2026.

The scenes in this book are composites. Names, dialogue, dates and quantities inside them are illustrative, not records of real people, teams or companies. The book describes practices, not legal, financial or employment advice.

salari.dev/books/before-the-room

BEFORE THE ROOM

Preface: the view from the other chair

I work as a staff mobile engineer, and in about fifteen years of building mobile software I have sat on the interviewer's side of more than five hundred interviews. Debriefs, hiring committees, promotion calibrations, the quiet conversation after the candidate leaves the room. That seat shows you something the candidate almost never gets to see: the moment a decision about a person is made, and how often the reasons that settle it are not the ones anyone says out loud.

I have watched the same thing happen to technical decisions. The architecture call, the migration, the deadline, the launch that everyone agreed was green. For consequential decisions, the visible meeting is often not where the decision happens. It was shaped earlier, in a hallway, in a review comment, in an incentive nobody named, and the meeting largely ratifies it. I have seen engineers lose outcomes they should have won because they prepared to win the meeting, bringing a better diagram to a room that had already made up its mind.

This book is about the gap that opens when you become the person a result depends on before anyone gives you the authority to command it. Senior engineer, tech lead, staff engineer: the title changes, the gap does not. You are accountable for something you cannot order into existence. You can be completely right and still watch the outcome drift, and if you have felt that, this book is written for you.

I am not going to hand you leadership principles. There are good books full of those, and I will point you to some. What I could not find was the field manual for the specific moments where technical risk, incentives, power and decision rights collide, and where being right is the entry fee rather than the win. So I wrote it, from the vantage point I know best: the debrief room.

The scenes in this book are composites. The note on method and evidence that follows says exactly what in them is invented and what is not.

The book has one engine, three moves, and you will meet it in chapter three. The chapters before it exist to convince you that your problem is not a knowledge problem. Everything

after it is the three moves applied to the hardest situations a technical leader meets: the migration no team wants to pay for, the deadline you cannot hit, the status that is green on the dashboard and red in reality, the incident, the conflict with someone smarter than you in a narrow domain, the review where your work was real but the evidence was missing.

Read it with a current problem in mind. The book is built to be used on Monday, not admired on a shelf.

BEFORE THE ROOM

A note on method and evidence

This book is written from practice, not from a study. The vantage described in the preface is its advantage and also its limit, and it is worth being clear about both.

The scenes are composites. Each one is assembled from patterns I have seen many times rather than from any single team, person or company. Inside a scene, the names, the dialogue, the dates, the durations and the numbers are illustrative details, composed so the pattern can be followed; read them as a worked example, not as case-study data. What is not invented is the pattern underneath. The rule I have kept is about evidence: the book presents no fabricated statistics, studies, research findings or named real-world cases as support for its claims. When a scene shows a hiring manager's sentence or a debrief's verdict, it is a pattern I have heard in many forms, written as one line, not a transcript.

I have tried to keep the language of certainty honest. When I write that I have repeatedly seen something, that is a practitioner observation. When I write that this suggests a rule, that is my inference from those observations, and you may reasonably draw a different one. The notes and further reading at the end list related published work on organizational decision-making, incident analysis and engineering practice. That is where the broader literature lives; it is not the source of the ideas here.

The individual tools are not mine. Decision records, stakeholder maps, escalation briefs, postmortems, explicit decision rights and pre-alignment are established practice, and many teams already use them well. What this book adds is the synthesis: naming the authority gap, running Surface, Shift and Secure in that order, carrying each decision in a durable artifact, and turning the whole thing into a weekly operating system.

The book also has boundaries, and it names them where they matter. Not every organizational failure can be solved by a sufficiently skilled engineer. Some technical objections are simply correct. Some costs cannot be made fair, and some decisions belong above you, including risks that no one at your level can rightly accept. Where a technique could be used to silence dissent, shift risk onto someone without the authority to hold it, or paper over structural harm, I have tried to say so plainly. Surface, Shift and Secure are tools to test, not laws proven in a lab: treat them as hypotheses to run on your own situation. The evidence that should convince you is the outcome you manage to move.

BEFORE THE ROOM

How to use this book

Read part one in order. It names the problem and, in chapter three, gives you the engine: three moves called Surface, Shift and Secure. Nothing else in the book makes full sense without them.

After that, the chapters are close to modular. Parts two, three and four are named for Surface, Shift and Secure in turn and run through the hardest situations a technical leader meets. Every chapter uses all three moves; the part names the move that is hardest there. Part five turns the moves into a system: a weekly rhythm, a thirty-day repair, and the evidence that decides your level. If you are staring at a stalled migration this week, read chapter thirteen tonight. If your project is green on the dashboard and you do not believe it, read chapter fourteen.

If you do not yet know which chapter you need, start by naming the kind of failure you are in.

IF THE WORK IS STUCK BECAUSE	IT IS A	START WITH
Nobody can say what is actually being decided	Decision failure	Chapter four, hidden-decision worksheet (template 1)
Nobody owns it, or the wrong person does	Ownership failure	Chapter seven, decision-rights map (template 3)
The people who pay for it gain nothing from it	Incentive failure	Chapters five and six, stakeholder and incentive map (template 2); chapter thirteen, migration wave plan (template 10)
Everyone says it works, and nobody can prove it	Evidence failure	Chapter fourteen, outcome evidence table (template 12)
Teams agreed, but each means something different by done	Coordination failure	Chapter eleven, outcome contract (template 9)

Every chapter has a short block called "In the field." It names the move the chapter drills, the artifact that does the work when you are not in the room, the signal that tells you whether the outcome held, the level calibration, the boundary where the method stops applying, and the one sentence to say. The artifacts live in the appendix as one system,

each with a completed example. If you want one place to start after the last page, template 16, the thirty-day repair plan, runs all three moves once on a single real situation.

A word on levels. The book is written for senior engineers, tech leads and staff engineers, and the level line in each block uses one lens. A senior engineer solves the problem. A staff engineer changes the decision around the problem. A principal engineer changes the system that keeps producing that decision. Each block says what that looks like in its own situation. The principal-level observations come from working alongside principals, seen from the staff seat, not from having held that level myself. The lens describes the scope of a move, not a universal career ladder; companies level differently, and it still works where the titles differ.

This is a field guide. Mark it up, argue with it, take one sentence into your next difficult meeting and see what happens. The test of the book is whether an outcome moves.

BEFORE THE ROOM

What the full book covers

Preface: the view from the other chair

A note on method and evidence

How to use this book

PART ONE. THE GAP

1. The engineer who was right and lost
2. Responsibility without control
3. The decision is made before the meeting

PART TWO. SURFACE

4. Find the hidden decision
5. Map power before you argue
6. Read the incentive conflict
7. Credibility and decision rights

PART THREE. SHIFT

8. Pre-alignment is not politics
9. Strategy that can be adopted
10. Architecture alignment and decision systems
11. Turn ambiguity into executable work
12. Scope, risk and deadline negotiation
13. Lead migrations when the cost is local
14. When green status hides red reality

PART FOUR. SECURE

15. Escalation without status warfare
16. Incident leadership when the system failed
17. Conflict with high performers, product and executive pressure

PART FIVE. MAKE IT A SYSTEM

-
18. The weekly operating system
 19. The thirty-day repair
 20. The evidence built before the review

REFERENCE

Appendix: the artifact system

Notes and further reading

PART ONE

The gap

why being right stopped working

CHAPTER 1

The engineer who was right and lost

The debrief runs for twenty minutes and the candidate is not in the room.

Five of us are. The candidate interviewed for a staff role, and on the whiteboard, an hour earlier, he had been the strongest engineer any of us saw that week. He caught a consistency bug in the design before the interviewer finished drawing it. He knew the failure modes of the datastore cold. When someone pushed on his caching call, he defended it with numbers and he was right to.

And then the hiring manager says the sentence that ends it.

"He was clearly the best engineer in the loop. I just do not see him moving anything at this level."

Nobody argues. That is the part that stays with you. Five people who agreed he was the sharpest technical mind of the week also agreed, without much discussion, that he would stall as a staff engineer. The loop downlevels him. The verdict had formed in a single moment of the system design round, when the interviewer asked what he would do if the team owning the upstream service refused to make the change his design depended on.

He said, "I would explain why they are wrong."

He meant it as strength. He heard a test of whether his design was correct. The question was whether he could move an outcome that lived inside a team he could not command, and his answer told us that when he hit that wall, he would push harder on the wall.

The scene is a composite, like every scene in this book, but the verdict is one I have heard in many forms.

What the room was actually testing

In the debriefs I have sat in, this profile, the engineer who is right and cannot move anything, is one of the reasons I have most often seen a strong technical candidate downlevelled. He rarely sees it coming, because from the inside it does not feel like a weakness. It feels like integrity.

Many engineers assume the thing between them and the next level is knowledge: deeper distributed systems, cleaner concurrency models, more patterns. So they study, get sharper, and the wall stays where it was.

Finding the right answer was the entry fee. What that debrief was weighing, as I read it and many like it, was whether he could move an outcome that depends on people, teams and leaders he did not control. He could find the right answer faster than anyone in the building. He had no move for the moment after it, when a person with their own roadmap, their own on-call load and their own reasons says no. In technical leadership that moment is much of the job.

Call what he was standing in the authority gap: the distance between what you are accountable for and what you are allowed to command. It tends to widen with each level, and chapter two maps it. For now it explains the question: the interviewer asked a staff candidate about the team that refuses because, at that level, the team that refuses is where much of the week goes.

The fixes that do not work

When they first feel the gap, many strong engineers reach for one of three tools, and on their own all three keep them stuck.

The first is to be more right: more data, a cleaner diagram, a tighter proof. Sometimes that works. When the resistance comes from an incentive, a better argument can make the proposal look more threatening instead of more persuasive, and chapter six is about why.

The second is to work harder. If the outcome will not move through the system, they carry it themselves, absorbing the coordination and the toil. It scales until it does not, and it teaches the organization that nothing needs to change.

The third is private frustration. The org is broken, the politics are stupid, the other teams are irrational, and they retreat into being correct and unheard.

None of the three touches the decision, and the decision is where outcomes are won. The candidate did not need to be smarter. He needed to see the upstream team's refusal as a decision, owned by someone, driven by an incentive he had not looked at.

Hold onto that shape, because naming the problem correctly comes first. You are not underqualified. You are standing in the authority gap, reaching for tools built for a smaller job, and the question is whether you learn the moves this job needs before or after the debrief you never get to hear.

IN THE FIELD

The move. Before the three moves comes the one that makes them possible: see the gap. Notice when you are reaching for individual-contributor tools, being more right and working harder, in a job that has become about moving decisions you do not own.

The artifact. No template yet. The first tool is a question you ask yourself the moment an outcome stalls: which decision is stuck, who owns it, and what would move them.

The signal. You catch yourself about to bring more data to a wall, or to carry the work alone across the line, and you stop and look for the decision underneath instead.

The level. Faced with the refusing upstream team, a senior engineer checks that the design is sound and explains it well; a staff engineer treats the refusal as a decision with an owner and an incentive. A principal asks why the organization keeps producing refusals like it, such as cross-team dependencies that nobody funds.

The boundary. Sometimes the stalled outcome is a knowledge gap or a wrong answer. When the work itself is flawed, fix the work first, because no amount of decision-moving rescues a bad result.

The line. "The room is not testing whether I am right. It is testing whether I can move something I do not control."

CHAPTER 2

Responsibility without control

Priya's name is on the launch. It went out in an email from her director three weeks ago: "Priya is driving the checkout migration, please give her whatever she needs." She did not ask for the sentence and cannot give it back. What she has, from that moment, is a result she will be measured on and a set of teams who heard the same email and filed it under things that are not their problem.

The week that follows has a shape I have seen many times. The migration depends on the payments team shipping a new idempotency key by the end of the sprint. Priya does not manage the payments team. Their lead is friendly, agrees the work matters, and has a director of his own who has told him this quarter is about latency. So the key slips a sprint, politely, with reasons, and the lever that would move it belongs to people who do not report to her and are not wrong to protect their own commitments.

Meanwhile the staff engineer on platform has good opinions about her rollback strategy, posted in a review comment on a Friday afternoon, that reopen a question she thought was closed and that she is not senior enough in that domain to overrule. The product manager wants the date pulled in, because sales promised a customer. And the engineer on her own team who knows the legacy payment flow best thinks the migration is riskier than she is letting on, is right often enough that she cannot dismiss him, and says so in a way that makes the junior engineers nervous.

She will answer for all of it in six weeks and can directly command almost none of it. That distance, between what lands on her name and what obeys her hands, is the authority gap. It is the job she was given, arriving in its normal clothes.

The gap is the promotion

The instinct, the first few times, is to treat this as a staffing error. Someone forgot to give you the authority that should come with the accountability, and once the reorg settles or the title arrives, the rights will catch up.

In the organizations I have seen, they mostly do not, and waiting for them is expensive. The director who wrote the email is managing the same gap one level up, moving outcomes through peers she cannot direct. What I have observed is that the gap rarely closes as

people climb; it becomes much of the content of the role, which is why I read promotion at these levels as largely a bet on skill at operating without authority. The debrief in chapter one was trying to predict how a candidate would behave in exactly this week.

As an individual contributor, Priya was responsible mostly for what she could change. Now her accountability has crossed the boundary of her control, and being correct and capable, still necessary, has stopped being sufficient. From the inside that feels like needing more of what always worked, so she does exactly that. She brings the payments lead incident data and a diagram of what breaks if the key ships late. He agrees with every slide and his sprint does not change. She writes the key herself over three late nights. The migration stays on schedule, Priya is exhausted, and the organization has learned that it does not need to fund the key, because Priya will.

Be precise about what you actually hold

What you reliably hold is technical credibility, the right to be heard; what you usually do not hold is decision rights over the systems your result depends on, such as the roadmap that would fund Priya's key or the rollback design one Friday comment can reopen. Chapter seven maps the distance between the two.

State the gap plainly and it stops being a mood and becomes a map. You are responsible for an outcome produced by decisions you do not own, made by people you do not manage, driven by incentives you did not set. Every noun in that sentence is a place the outcome can move. The payments team's sprint priority is a decision with an owner. What green is allowed to mean at the launch review is a decision nobody has made yet.

For Priya, chapter one's question points at the payments lead's director, the latency goal underneath the polite slip, and the one artifact that might change his sprint math. She will know it is working when the key gets prioritized because its owner chose to move it. Nobody is going to hand you the authority. You can learn to produce the launch without it.

IN THE FIELD

The move. Name precisely what you hold and what you do not. You hold credibility, the right to be heard, not decision rights over the systems your result depends on. Spend your effort on the decision and its owner.

The artifact. No template yet. Before any document, you need the one-sentence map of your own situation: you are responsible for an outcome produced by decisions you do not own, made by people you do not manage, driven by incentives you did not set.

The signal. You stop waiting for authority to catch up with accountability, and start asking, for every stalled outcome, which decision is stuck and who owns it.

The level. A senior engineer gets the idempotency key prioritized by working the problem with the payments lead instead of writing it at midnight; a staff engineer takes the latency-versus-migration conflict to whoever owns both priorities. A principal fixes how launches are assigned: the email that names a driver also names the decisions and owners the launch depends on.

The boundary. Naming what you hold does not manufacture influence you do not have. If you have neither credibility with the owner nor a path to them, the honest step may be to refuse the accountability rather than absorb a risk you cannot move.

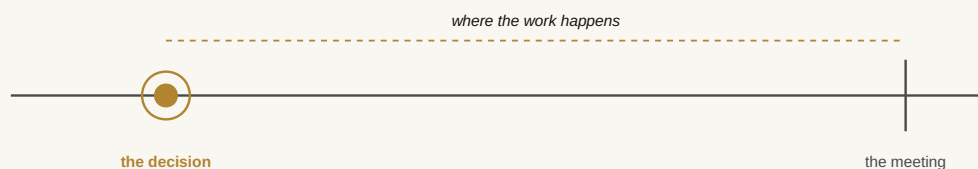
The line. "I do not own this decision, so who does, and what would move them?"

THE METHOD, IN ONE PAGE

The engine, and where to get the rest

You have met the problem. You are accountable for outcomes you cannot command, and being right is the entry fee, not the win. The whole book turns on one idea and three moves.

WHEN THE DECISION IS ACTUALLY MADE



The decision is often shaped before the meeting. The meeting frequently ratifies it. The work happens in the span between.

Surface. Find the decision that is actually being made, who owns it, and the incentive that will decide it.

Shift. Change that decision with one artifact and the exact words to say, in front of the right owner, before the room.

Secure. Set the threshold that triggers escalation and the follow-up signal that proves the change held.

This sample gives you the hook and the shape. The full method, run through seventeen situation chapters, with twelve figures and sixteen templates, each with a completed example, is in the full book.

GET THE FULL BOOK

Download the complete edition, PDF and EPUB, at salari.dev/books. Also available on Amazon in Kindle and paperback.