FOR PHYSICIANS WHO SPEAK

The Command Center

How to build a system that holds the answer, so nothing falls through while you're on a plane

CAMILLE ROSE

CORNERSTONE EA

CONTENTS

What's Inside

> A NOTE ON THE ARCH CHECKS
>
> Throughout this book you'll find boxes like this one. They are not filler. Each one asks you to look at how you are actually running things right now, not how you intend to.
>
> Answer them honestly and in writing. By the last page you will have an accurate picture of where your operation leaks, which is worth more than anything I can tell you in the abstract.

The Email She Never Saw

A program coordinator emailed my client about a venue change. Three weeks later, nobody had answered.

Not because she is disorganized. She is one of the most capable people I have worked with. The message landed in a folder she never opens, on an account she checks between clinic and a connecting flight. By the time it surfaced, the coordinator had followed up twice and assumed she had lost interest.

I found it because checking that folder is part of my Monday pass. Not because anything alerted me. Not because a system flagged it. Because a human being goes looking in the places where things quietly go to die.

That is the whole problem in one story. You did not become a physician to manage a filing system. You became good enough at your specialty that people started inviting you places, and the invitations did not stop arriving just because your inbox filled up.

Why I wrote this

Four reasons, plainly.

Because the failure is structural, not personal. Every physician-speaker I have met believes they are uniquely bad at admin. They are not. They are running two full-time jobs on a system designed for one.

Because most advice on this is useless. "Use a CRM." "Try time-blocking." That advice was written for people whose work is predictable. Yours is not.

Because I built something that works and there is no reason to keep it to myself. What follows is the actual architecture I run for a physician doing roughly ninety sponsored programs a year, not a theory about what might work.

Because you should be able to build the foundation yourself. I mean that. If you follow Part Three, you will have a working Command Center by the end of a weekend. Whether you want to run it yourself afterward is a separate question, and an honest one. I will come back to it.

The three things to take away

> IF YOU READ NOTHING ELSE
>
> 1. **A settled fact should be answered once.** Most of the chaos in a speaking calendar comes from re-deriving the same answers weekly, with a fresh chance to get each one wrong.
> 2. **Silence is the failure mode, not error.** Things do not usually get dropped. They stall, quietly, in the gap between "I'll get to that" and "did anyone get to that?"
> 3. **A system nobody opens is a filing cabinet.** The build is the easy half. What makes it work is a recurring pass through it by someone whose job that is.

Summed up in one sentence: **the point of the system is that the answer already**

exists somewhere you will actually look.

What I want for you

My goal is not that you finish this book. It is that you have a working Facts of Record table before you finish it. That is chapter nine, and it takes about twenty minutes.

My highest hope is more specific than "less stress." It is that you stop opening your laptop at eleven at night to check whether something got sent.

BEFORE YOU READ ON

If your speaking calendar has already outgrown what you can hold in your head, a conversation costs you nothing and will probably save you a weekend of trial and error.

Book a call: calendar.notion.so/meet/cornerstoneea/ea

I help physicians who speak build the operational spine their calendar already needs, so the back office stops costing them evenings they will not get back.

Let's build it.

Camille Rose
Cornerstone EA

PART ONE

What Changes

Proof that this works, who it works for, and the five problems it solves.

CHAPTER ONE

The New World

What it looks like when the system holds the answer.

The first thing I ever built was not impressive. It was a table with four columns: the fact, the source it came from, the date it was confirmed, and who confirmed it.

I built it because of a venue. My client was flying to a program in three days and our board still said the location was unknown. It was not unknown. It had been sitting in her own sent mail for six days. Three separate places in our system were confidently telling us we did not have it.

That was the moment I understood the actual problem. It was not a memory problem. It was that the same question was being answered from scratch every week, and each time there was a fresh chance to answer it wrong.

What changed, measurably

CASE STUDY: A HIGH-VOLUME ALLERGY AND IMMUNOLOGY SPEAKER

She came to me already successful and already drowning. Roughly ninety sponsored programs a year across multiple manufacturers, plus a clinical practice.

- **Decisions that used to sit for two weeks now land the same day.** Not because she got faster. Because they stopped arriving as open-ended questions.
- **Twenty-three stale drafts cleared in one audit.** Seven of them should never have been sent. Two were duplicate expense submissions that would have double-filed with the same sponsor.
- **A fourteen-day silence caught and closed.** A university contact had been waiting on us. Nobody ignored him. He fell into a gap nobody was watching.
- **Reimbursements stopped aging past thirty days** because unpaid finally became an event rather than a non-event.

What she does now that she could not do before: she gets on a plane without a mental inventory of what might be rotting in an inbox. That is the actual deliverable. The tables are just how we get there.

The strategy underneath all of it

There is one thread running through everything in this book, and it is worth naming early: **write down the answer where you will look for it next, not where you found it.**

Almost every operational failure I have cleaned up traces back to a violation of that sentence. The venue was in the sent folder, which is where it was found, not where anyone would look for it. The reply was checked off on a to-do list, which is where the intention lived, not where the proof lived.

The information almost always exists. It just has no home.

Borrowed proof

This is not a novel idea, which is a point in its favor. Aviation solved it with checklists after concluding that expert memory was the weak link, not expert skill. Surgical safety checklists cut complications substantially for the same reason. Neither profession decided its people were careless. Both decided that unaided recall under load is a bad place to store anything that matters.

Your speaking calendar is a high-load, high-interruption environment with multiple external parties and money attached. It is exactly the environment where written records outperform capable people.

What I no longer do

Before: reread the same email three times because I never processed it fully the first time. Search for the same flight confirmation weekly. Reconstruct expense reports from memory.

After: open one page, see what is actually outstanding, and act. The rest of the day is real work.

> ARCH CHECK ONE
>
> Answer these in writing before moving on.
>
> 1. Name one question about your own schedule you have answered more than twice this month. Where does the answer currently live?
> 2. When you last needed a venue, a contract value, or a contact name in a hurry, where did you look first? Was it there?
> 3. How many separate places would someone have to check to reconstruct your next program in full?
>
> If the answer to the third question is more than two, the rest of this book is written for you.

CHAPTER TWO

Who This Book Is For

And, just as usefully, who it is not for.

I did not set out to solve this. I backed into it by watching the same thing break repeatedly.

My early mistake was assuming the problem was capture. If we could just get everything written down somewhere, the theory went, nothing would fall through. So I built capture. Beautiful capture. Nested pages, linked databases, tidy properties.

My client never opened it.

Not out of resistance. She told me plainly: she lives in her inbox and on her phone, between clinic and a plane, and she has no reliable window to go hunting through pages. The system asked her to come to it. She could not.

The turning point

The fix was not a better system. It was inverting the direction of travel. The system now goes to her. Every decision she owns arrives in one weekly message, same shape every time, and each item is a yes, a no, or a date. Never an open question. Never "what are your thoughts on the Vienna situation."

> *A system that does not fit the person is just a second job you handed them.*

Four things that changed

1. **Facts became settled rather than perpetually re-derived.** One table, one source, one date.
2. **Asks acquired a clock.** The moment we ask anyone for anything, it gets a row with a date. Age is computed, never remembered.
3. **Completion became provable.** A checkbox is a claim. A sent message is proof. If there is no sent message, the item reopens.
4. **Programs acquired stages,** so a stall became visible instead of silent.

Who should keep reading

This book is for you if you are a physician who speaks, and most of the following are true.

- You carry a clinical practice alongside a sponsored speaking calendar.
- You have double-booked clinic and a program at least once.
- Your expense reports get reconstructed from memory more often than from receipts.
- "I'll deal with it on the plane" is a genuine part of your workflow.
- At least one reimbursement is currently older than you would like to admit.

Who this will not help

I would rather lose you here than waste your evening.

If you are employed by a large group or an academic department that already handles your travel and filings, your bottleneck is different from the one this book solves. You may still find Part Two useful for the parts your institution does not touch, but the full build is aimed at owner-operators and independent physicians who carry their own back office.

If you do fewer than about a dozen programs a year, a Command Center is more infrastructure than your volume justifies. Build the Facts of Record table from chapter nine and stop there. It will be enough.

What I am really teaching in these pages is how to move the answer out of your head and into a place that survives your calendar.

> ARCH CHECK TWO
>
> 1. Where do you actually live during a working day? Inbox, phone, EMR, calendar? Be honest, not aspirational.
> 2. Has anyone ever built you a system you stopped opening within a month? What specifically made you stop?
> 3. If a decision needed to reach you today and be answered today, through which channel would it have to arrive?
>
> Whatever you wrote for question three is where your decision queue has to live. Everything else is a preference.

The Five Problems

These are the failures I see in every practice before anything is built.

Problem one: the same question, answered weekly

You know the venue. Someone knows the venue. The information exists in an email, a text, a confirmation. But because it was never written where you look, it gets re-derived every time it is needed, and each re-derivation is a fresh opportunity for error.

How the framework solves it: the Facts of Record table. One row per settled fact, with its source and the date confirmed. Once a fact is in there, nobody goes looking again. A newer answer only wins if it comes with a newer source.

Problem two: nothing records when you asked

"Flag anything that has been waiting more than a week" sounds like a simple request. It is impossible, because nothing anywhere records the moment you asked.

How the framework solves it: the Waiting On ledger. This is the core mechanism and gets its own chapter. The short version is that the ask becomes a row with a date, and aging becomes arithmetic rather than memory.

Problem three: intention mistaken for completion

A reply gets drafted. It gets flagged to send Monday. Someone ticks the box. It sits in drafts for four days while the person on the other end concludes you have gone quiet.

How the framework solves it: the sent-box rule. Every item marked done is matched against the sent folder before it counts. No matching message means the item reopens. This one rule catches more than any other single thing I have implemented.

Problem four: stalls are invisible

Things almost never get dropped deliberately. They stall. An invitation waits on a reply. A trip sits half-booked. An expense report is built and never submitted. A fee goes unpaid, and unpaid does not feel like an event, so nobody reacts.

How the framework solves it: named stages plus one rule underneath them. Once a thing has a stage, you can see it stop moving. Before that, it just quietly ages.

Problem five: decisions arrive in the wrong shape

Someone asks you an open question at the exact moment you have no capacity to think. So you defer it. Then it becomes stale, and the deferral compounds into a small crisis.

How the framework solves it: every decision arrives as a yes, a no, or a date. Framing is the work. If a question cannot be reduced to one of those three, it is not ready to be asked.

ARCH CHECK THREE

Score yourself one to five on each, where one is "this is fine" and five is "this is my life."

Problem	Score
Re-answering settled questions	
No record of when we asked	
Marked done but never sent	
Silent stalls	
Open-ended decisions	

Your highest score is where to start. Not the first chapter, the highest score.

A SHORTCUT

If you scored four or five on three or more of these, you are past the point where a weekend build alone will fix it. That is not a sales line, it is a volume observation. Let's talk about what running it actually takes.

calendar.notion.so/meet/cornerstoneea/ea

The Secret Key

One external system of record that holds the answer. I call it the Command Center.

Everything in this book depends on a single idea, and if you take only one thing from these pages, take this: **there must be exactly one place that holds the current answer, and it must not be your memory or your inbox.**

Not three places. Not "mostly the shared drive." One.

The questions that led me here were unglamorous. Why do I keep finding the answer in the sent folder? Why does the board disagree with the inbox? Why does she ask me something on Sunday that we settled on Tuesday? Each of those is the same question wearing a different coat. There was no single authority, so every source was equally plausible and equally wrong.

What your clients will ask about this

Is this just a project management tool?

No. Project tools track tasks. A Command Center tracks *settled facts, open asks, and pending decisions*, which are three different things that most tools flatten into one list. The distinction is the whole point.

I already use Notion. Isn't this the same?

Notion is where mine lives, but the tool is not the method. Most Notion workspaces I have inherited are elaborate filing cabinets. They store; they do not surface. The difference is whether anything ever gets pushed to the person who has to act.

Won't this take more time than it saves?

For the first two weeks, yes. Honestly, yes. After that the arithmetic reverses sharply, because you stop paying the re-derivation tax on every recurring question.

Can't I just be more disciplined with email?

Email is a transport layer, not a system of record. It has no concept of "settled," no concept of "we are waiting," and no way to compute age. You can be extraordinarily disciplined and still lose things in it, which is exactly what keeps happening to you.

What if I travel constantly?

That is the case this was built for. The Command Center is not a place you visit. It is a place that generates what comes to you.

Who owns it if I hire someone later?

You do. Build it in your account, not an assistant's. This matters more than people expect.

Why single-source-of-truth is not my idea

Every mature operational discipline lands here eventually. Version control exists because engineering teams concluded that multiple competing copies of the truth is the root cause, not a symptom. Accounting has run on a general ledger for centuries for the same reason. Medicine itself moved to the unified chart because parallel records were

killing people.

You already accept this principle in your clinical life. The chart is the chart. Nobody keeps a personal side-record of a patient's medications and hopes it matches. Your speaking practice is the last place you are still running parallel records.

> ARCH CHECK FOUR
>
> 1. List every place a fact about an upcoming program might currently live. Email, texts, a notes app, your calendar, a spouse's memory. Count them.
> 2. If two of those sources disagreed tomorrow, which one wins? Is that rule written down anywhere, or does it live in your head?
> 3. Pick one program in the next sixty days. Can you name its venue, contact, fee, and travel status without opening anything? Which one did you miss?

The Catch

Why your last system failed, and why it was not your fault.

Here is the part most people selling you a template will not say.

You have probably built something like this before. A Notion workspace. A spreadsheet. A folder structure with real thought behind it. And it worked for about three weeks.

You did not fail because you lack discipline. You failed because **a system of record is passive, and passive systems decay silently.**

How everyone gets this wrong

The standard advice stops at the build. Set up your databases, define your properties, migrate your data, and you are done. That advice treats the system as the product.

The system is not the product. The *pass* is the product.

A Command Center that nobody walks through on a schedule degrades in a specific and predictable order. First the Waiting On ledger goes stale, because closing rows requires noticing that an answer arrived. Then the Facts of Record drift, because a newer source showed up and nobody updated the row. Then people stop trusting it, and once trust goes they revert to the inbox, and you are back where you started with an extra tab open.

The build is a weekend. The pass is forever.

The one thing you must add

A recurring, structured walk through the system by someone whose job it is. I call it the daily pass, and its order matters more than its existence.

1. **Full inbox**, not just unread, organized by program, newest first.
2. **Stale drafts** reviewed and thrown out.
3. **Sent folder** checked against the newest inbound item, to see whether it is already answered.
4. **Calendar** reconciled against program stages.
5. **Reference pages** checked against the standard operating procedure and known preferences.

That order is not arbitrary. Checking sent before drafting prevents duplicate replies. Clearing drafts before checking sent prevents you from resurrecting something that should die. I learned both of those the expensive way.

The honest part

I want you to build this. I have written Part Three so you genuinely can, and if you follow it you will have something real and working that belongs to you.

But I am not going to pretend the pass runs itself, because that would be the same lie

that made your last system fail. Somebody has to do it, on a schedule, when it is boring, in the weeks when nothing appears to be wrong. That is precisely when it is doing its job.

You have three options, and all three are legitimate.

- **Run it yourself.** Twenty to forty minutes, three times a week, protected. Viable if your volume is moderate and your weeks are predictable.
- **Train someone in your practice.** Your office manager often feels this pain most acutely and may want the remit. Give them Part Three and the SOP.
- **Hand it to someone whose entire job is this.** Which is what I do.

What does not work is intending to do it yourself and then not doing it. That produces a system that looks authoritative while quietly being wrong, which is worse than no system at all, because you will trust it.

> ARCH CHECK FIVE
>
> 1. Name the last organizational system you abandoned. How long did it last?
> 2. What specifically happened the week you stopped? Travel, a deadline, illness, a launch?
> 3. Look at your next eight weeks. Is there a window that looks like the week you named in question two? What happens to the pass during it?
>
> Answer three honestly. It is the whole ballgame.

PART TWO

How It Works

The mechanism, the method, and everything that has to sit around it.

The Core Mechanism

The Waiting On ledger. Without this, nothing else in the system functions.

You can have a beautiful Command Center, a perfect Facts of Record table, and staged programs, and still lose money and relationships. The thing that prevents it is the least glamorous object in the entire build.

The principle you must accept: an ask is an object with a lifespan, not an event that happened. Until you treat it that way, you are managing your calendar with half the information.

How it works

The moment you ask anyone for anything, it becomes a row. Not when they are late. Not when you remember. At the moment of asking.

Field	What goes in it
What	The specific thing owed. "Signed contract," not "contract stuff."
Who owes it	A named person, never an organization.
Direction	They owe us, or we owe them. Both are tracked.
Date asked	The date. This is the field that makes everything else possible.
Age	Computed, never typed. Today minus date asked.
Program	Linked to the program it belongs to.
Status	Open or closed. Closed only when the answer actually lands.

The direction field is the one people skip, and it is the one that saves relationships. Most of us track what others owe us. Almost nobody tracks what we owe others. The fourteen-day silence I mentioned earlier was in the second category. Nobody had ignored him. There was simply no object representing our own debt.

The rule that makes it work

Any open row aged past seven days gets a holding reply the same day it is noticed. Not the answer. A holding reply. "We have this, give us about a week."

Silence damages relationships. Waiting does not. People are remarkably patient when they know they have been heard, and remarkably unforgiving when they do not.

Nothing here goes quiet. Every ask gets a date, and the clock starts the moment we ask.

If this feels like a lot

It will, at first. You will think you are creating administrative work to manage administrative work, and for about ten days that will be true.

Then something shifts. You will stop carrying the low background hum of *what am I forgetting*, because the answer will be visible. Most people describe that as the moment the system started paying rent.

What makes this different from every task app you have tried: task apps record what *you* intend to do. The Waiting On ledger records what is *owed between people*, in both directions, with time attached. No consumer tool models that well, which is why you have never quite made one work for this.

ARCH CHECK SIX

1. Right now, without looking: name three things you are waiting on from other people. How long has each been outstanding? How confident are you in those numbers?
2. Now the harder one. Name three things *other people are waiting on from you*.
3. Which list was harder? For nearly everyone it is the second, and that is the list that costs you relationships and repeat invitations.

BUILD THIS ONE FIRST

If you implement nothing else from this book, implement the ledger. If you want it set up correctly the first time, with the aging formula and the holding-reply rule already wired in, that is a single working session.

calendar.notion.so/meet/cornerstoneea/ea

The Cornerstone Method

The whole framework, step by step.

Four layers, built in order. Each one depends on the one before it, so resist the urge to jump ahead to the interesting parts.

Record what is settled. **Track** what is owed. **Stage** what is moving. **Deliver** what needs deciding. If you do those four things in that sequence, you have a Command Center. Everything else is refinement.

Layer one: Record

Build the Facts of Record table. Four columns minimum: fact, source, date confirmed, confirmed by.

Actions: Open your next three programs. For each, write down venue, date, sponsor contact name and email, fee, and travel status. For each entry, note where the information came from and when it was confirmed. Where you cannot find a source, mark it unconfirmed rather than guessing. The unconfirmed rows are your immediate work list.

The rule: a row changes only when a newer source appears. Not when someone remembers differently. Not when it seems wrong. A newer source, or it stands.

Layer two: Track

Build the Waiting On ledger as specified in chapter six.

Actions: Seed it from your sent folder. Go back thirty days and find every message where you asked someone for something. Each becomes a row with its actual send date. This is tedious and takes about an hour, and it will surface at least two things you had forgotten entirely. It does for everyone.

Layer three: Stage

Give every program a lifecycle. Seven stages, in order.

Stage	Stall rule
1 Invited	Never sits without a reply. Same-day acknowledgment.
2 Accepted	Travel must be booked within fourteen days.
3 Travel booked	Confirmations filed to Facts of Record.
4 Confirmed	Contact and venue verified against a source.
5 Attended	Receipts captured within seventy-two hours.
6 Expensed	Never sits longer than fourteen days.
7 Paid	No fee passes thirty days without a chase.

Break any of those and the program surfaces on today's board automatically. Naming the stages was the entire trick. Once a thing has a stage, you can see it stop moving.

Layer four: Deliver

This is the layer almost everyone omits, and it is why almost everyone's system dies.

Once a week, generate one message containing: what got done, then what only you can decide, ordered by urgency. Every decision framed as a yes, a no, or a date.

Actions: Pick the channel you actually read. Not the one you should read. Draft the standing format once and reuse it. Never send an open question. If you cannot reduce something to yes, no, or a date, it is not ready and the work of framing it is yours, not the reader's.

What you get

Follow those four layers exactly and you will reach a specific state: you can answer any question about any program in under thirty seconds, and you will know what is overdue without thinking about it. That is the outcome. Not inbox zero, which is a vanity metric. Operational certainty.

> ARCH CHECK SEVEN
>
> Map your current reality against the four layers.
>
> | Layer | Do you have it? Where does it live? |
> |---|---|
> | Record | |
> | Track | |
> | Stage | |
> | Deliver | |
>
> Most people find they have some version of Record and nothing else. If that is you, the ledger is your next move.

The Critical Components

Everything that has to sit around the framework for it to hold.

The written SOP

Numbered sections, boring on purpose. Mine covers settled facts, calendar rules, reply templates, and who owns what. The test of an SOP is not whether it is comprehensive. It is whether a competent stranger could run your week from it.

Write it in numbered sections so it can be cited. "Per section four" is a functional sentence. "Per the document" is not.

Ownership check before expensing

This one has direct financial consequences and almost nobody does it.

Before any charge goes on an expense report, confirm you actually paid it. If the charge is not on your own account, or the hotel folio shows a zero balance, the sponsor already covered it. Filing it anyway is at best an error and at worst something a compliance officer will want to discuss.

I have caught this more than once. It is the single highest-consequence check in the entire system.

Drafts hygiene

Nobody talks about the drafts folder. It is where good intentions go to quietly not happen.

Audit it weekly. Anything older than a week either gets sent today or gets deleted. There is no third option, and "I'll get to it" is how you end up with twenty-three of them and two duplicate expense filings.

Meeting harvest within twenty-four hours

A recording is not action. A transcript is not action. If a call produces decisions and they are not extracted into the Command Center within a day, they will not be extracted at all.

The AI layer, honestly

I use Claude against my Command Center daily, and it is genuinely useful. It is also the part most oversold, so let me be precise about where the line falls.

It is good at: summarizing a thread into what was actually decided, drafting the weekly decision message from raw notes, reconciling an inbox against a sent folder to spot unanswered items, and turning a rambling call transcript into candidate ledger rows.

It is bad at: knowing what is true. It will confidently produce a venue. It has no idea whether that venue is correct. Everything it generates has to land against the Facts of Record and be checked, or you have automated the production of confident errors.

The useful framing: AI is a fast drafter and a poor witness. Let it write; do not let it decide.

Two myths worth killing

"Automation will handle the pass." Automation handles rules. The pass is mostly judgment about ambiguous things, which is exactly what rules cannot do. Automate the aging calculation and the reminders. Do not pretend that is the pass.

"More tools means more coverage." Every additional tool is another place the truth can live, which is the original disease. Fewer places, checked properly.

Worth reading

Atul Gawande's *The Checklist Manifesto*, for why competent experts benefit most from written process. And any decent primer on double-entry bookkeeping, which will teach you more about tracking obligations in two directions than any productivity book will.

> ARCH CHECK EIGHT
>
> 1. Open your drafts folder right now and count. How many are older than a week? What is the oldest?
> 2. Pull your last submitted expense report. Can you prove, from a source, that you personally paid every line on it?
> 3. If you were unreachable for two weeks, could a competent stranger run your speaking calendar from what is currently written down? What is the first thing they would get wrong?

PART THREE

Build It This Week

Concrete instructions. By the end of this part you will have a working system.

CHAPTER NINE

The First Step

Twenty minutes. Start here, today, before you finish the book.

The first thing to do is build the Facts of Record table for your next three programs. Not your whole year. Three programs.

Why this is hard

Not technically. Emotionally. Building it means confronting how much you do not currently have written down, and that is uncomfortable enough that most people find a reason to do it later.

Do it anyway, and do it badly. An incomplete table with honest gaps beats a perfect table you never start. The gaps are the point; they are your work list.

Why it matters

Everything else depends on it. The ledger needs programs to attach to. Stages need facts to verify. The weekly decision message needs somewhere to point. Record is the foundation, which is why it is layer one.

Exactly how

1. Open a new Notion database, or a spreadsheet. Genuinely, either works. Do not spend today choosing a tool.
2. Create columns: Fact, Value, Source, Date Confirmed, Program, Status.
3. Status is a simple select: Confirmed or Unconfirmed.
4. For each of your next three programs, create rows for: venue name, venue address, program date and time, sponsor contact name, sponsor contact email, fee, travel booked yes or no, hotel confirmation number.
5. Fill what you know. For each, name the source: "Sanofi confirmation email, 12 March."
6. Anything you cannot source, mark Unconfirmed. Do not guess. A confident wrong row is worse than a blank one.
7. Every Unconfirmed row becomes a Waiting On row tomorrow.

What you will find

Two things, reliably. You will find at least one fact you were confident about that has no source anywhere. And you will find at least one thing you believed was settled that is not.

That second discovery is the one that pays for the exercise. Better to find it now than three days before the flight, which is how I found the venue.

ARCH CHECK NINE

Do not answer these until the table exists. It takes twenty minutes.

1. How many rows ended up Unconfirmed?
2. Which single unconfirmed row would cause the most damage if it were wrong?
3. Who is the named person who can confirm it, and when will you ask?

Question three is your first ledger row. You have started.

The Eight Components

What a complete Command Center contains.

1. Facts of Record

Settled facts with sources and dates. **Why it matters:** it ends re-derivation. **Action:** built in chapter nine. Add a rule that no row changes without a newer source.

2. The Waiting On ledger

Open asks in both directions, with dates and computed age. **Why it matters:** it makes silence visible. **Action:** seed from thirty days of sent mail.

3. Program pipeline

Every engagement with its stage. **Why it matters:** stalls become detectable. **Action:** one row per program, stage as a select field, sorted by date.

4. The decision queue

Pending items only you can resolve, framed as yes, no, or a date. **Why it matters:** it is the only part the physician actually touches. **Action:** build it as a filtered view, not a separate list, so it can never drift from the source.

5. The written SOP

Numbered sections covering rules, templates, and ownership. **Why it matters:** it makes you replaceable in the good sense. **Action:** start with five sections. Add as you hit edge cases.

6. Reply templates

Standing language for the six or seven messages you send constantly: accepting an invitation, holding replies, chasing a fee, requesting a W-9, declining. **Why it matters:** most delay is drafting friction, not decision friction. **Action:** write the holding reply first. You will use it most.

7. Expense staging

Receipts captured within seventy-two hours, with the ownership check applied before anything is filed. **Why it matters:** it is where the money and the compliance risk both live. **Action:** photograph the folio at checkout, every time, even when the balance is zero. Especially then.

8. The daily pass checklist

The five-step order from chapter five, written down. **Why it matters:** it is the component that keeps the other seven alive. **Action:** write it as a physical checklist, not a habit. Habits fail in exactly the weeks you need them.

What you get when all eight work together

You get a practice where the answer already exists. Where a stalled thing raises its hand. Where a decision reaches you in a form you can resolve between patients. And where, if you were unreachable for a fortnight, someone competent could pick it up and keep it running.

ARCH CHECK TEN

Tick what genuinely exists today, not what you plan to build.

Component	Exists	Next action
Facts of Record		
Waiting On ledger		
Program pipeline		
Decision queue		
Written SOP		
Reply templates		
Expense staging		
Daily pass checklist		

Build It in a Weekend

A realistic schedule. Roughly six hours across two days.

Saturday morning, ninety minutes

Facts of Record for your next three programs, per chapter nine. Then extend to every program in the next ninety days. Work from confirmation emails only. If you cannot find a source, the row is Unconfirmed and that is the correct answer.

Saturday afternoon, ninety minutes

The Waiting On ledger. Open your sent folder, filter to the last thirty days, and read every message you sent that contained a request. Each becomes a row with its true send date. Then add every Unconfirmed fact from this morning as a row directed at whoever can confirm it.

Sort by age descending. Look at the top of that list. That is the most useful five minutes of the entire weekend.

Sunday morning, two hours

The program pipeline with the seven stages, and the decision queue as a filtered view. Then write your holding reply template, because you will need it before the ledger is an hour old.

Sunday afternoon, one hour

The SOP, first five sections: how facts get confirmed, how asks get logged, the stall rules, the holding-reply rule, and who owns which step. Then write the daily pass checklist.

Accelerators

Steal the structure, do not design it. Every hour spent on database design is an hour not spent on data. The schema in chapter six is sufficient. Use it.

Use AI for the tedious extraction. Paste a confirmation email and ask for the venue, address, date, contact, and fee as structured fields. It is fast and it is accurate at reading. Then verify against the original before it becomes a row, because it is not accurate at knowing.

Do not migrate history. Start from today plus the next ninety days. The urge to backfill two years is what kills weekend builds at hour four.

Set the recurring pass before you finish. Put it in your calendar this weekend, with a real time attached. A system without a scheduled pass is a system with an expiry date.

ARCH CHECK ELEVEN

1. Which weekend? Write the date.
2. What will you push to make room for it?
3. When the pass appears in your calendar on a bad week, what is your honest prediction about whether you will do it?

If you hesitated on question three, that is information, not a failure. It tells you which of the three options from chapter five is actually yours.

CONCLUSION

Where This Leaves You

The whole book in ten lines

- You are not disorganized. You are running two full-time jobs on a system built for one.
- The information you need almost always already exists. It has no home.
- Settled facts should be answered once, with a source and a date.
- An ask is an object with a lifespan. Log it the moment you make it.
- Track what you owe others, not just what others owe you. That list is shorter and costs more.
- Silence damages relationships. Waiting does not. Send the holding reply.
- A checkbox is a claim. A sent message is proof.
- Things do not get dropped, they stall. Stages make stalls visible.
- Decisions must arrive as a yes, a no, or a date. Framing them is the work.
- The build is a weekend. The pass is forever, and it is the part that fails.

Your next step

If you have not built the Facts of Record table, do that. It is twenty minutes and it is the only thing in this book that must happen before anything else.

If you have built it and you can see the pass fitting into your weeks, keep going. Chapter eleven is the schedule. Everything you need is in these pages, and I have not held anything back to create an artificial reason to call me.

And if you have built it and you already know the pass will not survive your March, that is not a failure of will. It is an accurate reading of your own calendar, and it is worth more than optimism. That is the conversation I would like to have with you.

BOOK A CALL

Thirty minutes. Bring your Arch Check answers, particularly chapters three, five, and ten. We will look at where your operation actually leaks and what running the pass would take. If it turns out you do not need me, I will tell you that.

calendar.notion.so/meet/cornerstoneea/ea

Resources

- **The Calm Close-Out Checklist** and the daily pass template. Ask me and I'll send them over.
- **The Checklist Manifesto**, Atul Gawande, on why written process helps experts most
- **CMS Open Payments**, openpaymentsdata.cms.gov, to see your own reported program payments against your records. Most physicians have never checked. It is a useful reconciliation.

About the author

Camille Rose came to this work through bookkeeping, where a missed filing is not an inconvenience, it is money. That is the habit of mind she brings to sponsor reimbursements and program logistics: reconcile it, source it, and never assume a thing happened because someone meant to do it.

She started her own practice because she needed real control over her own schedule. Three children and a household that homeschools will teach you quickly that a calendar either runs on a system or it runs you. That turned out to be the work she loves.

She now builds and runs the operational spine behind physicians whose speaking calendars have outgrown what any one person can hold in their head: travel, sponsor filings, contract and fee tracking, and the weekly decision queue that keeps the whole thing moving.

Cornerstone EA
kol-ea.com · hello@kol-ea.com
Book a call: calendar.notion.so/meet/cornerstoneea/ea

CORNERSTONE EA