

The AI Bootcamp

16

5 Steps for Jobs Too Big for One Prompt

In this chapter you'll learn

Why one enormous request buys a thin answer, and the 5-step sequence that fixes it

The AI Bootcamp

THE PROBLEM

Most AI education is a pile of disconnected tips. A LinkedIn carousel here, a YouTube tutorial there, a \$2,000 cohort course that goes out of date before the refund window closes. None of it is built to be read in order, so nothing compounds. You finish twelve videos and can recite twelve facts, not perform one new skill.

THE FIX

The AI Bootcamp is built the other way. 90 skills, packed 2 to 3 per chapter, 36 chapters, 3 tracks. Every chapter states what it needs from you and what it hands to the chapter after it, so skill 47 only works because skill 12 already landed. Read it in order and by the end you are not collecting facts about AI, you are running work through it.

LETTER FROM THE EDITOR

We built this because we kept meeting smart people who felt behind on AI for no reason other than nobody had organized the material for them.

This book does not assume you are technical, and it does not water anything down either. Every chapter earns its place: a real skill, a real exercise, a one-page companion you can flip back to in month three when you have forgotten the details but still need the answer.

If a chapter loses you, or a chapter changes how you work, I want to hear about it either way.

Kris

Editor, AI Central

Write to me at editor@thecentral.ai

THE 3 TRACKS

AI 101

Get running

Pick a tool, set it up, write a clear instruction, use it safely at work.

AI 202

Prompt properly

The framework behind every good instruction, and catching AI when it is wrong.

AI 303

Build without watching it

Connect AI to your tools, and automations that run while you are not looking.

IN THIS LESSON YOU'LL LEARN

By the end of this chapter you can tell which jobs are too big to ask for in a single message, run one of them as a sequence of 5 steps where each step takes the last one's output as its input, and place the cuts so the pieces assemble into one document instead of a pile.

You have the framework for a good prompt and you know how to make it think before it answers, and you still get fluent, weightless work back whenever the job is large. The problem is not your wording. Some jobs cannot be asked for in one message at all, and nobody tells you which ones, or where to break them.

CONTENTS	16.1	One enormous request buys you one thin answer
	16.2	The thinness has 3 causes, and better models fixed only part of one
	16.3	Five messages, one board update
	16.4	Cut where you would hand the work to a person

TOPICS REQUIRED

Chapter 15. You need the prompt framework from Chapter 13, the habit of asking it to reason before it answers, and one real piece of work from your desk that is too big to ask for in a single message.

AFTER THIS CHAPTER

Chapter 17 assumes you have run a sequence end to end and can name its steps in your own words, because those step names are what become saved, reusable prompts.

16.1 One enormous request buys you one thin answer

Nothing in it is wrong, which is what makes it so hard to fix. Every heading is present, every paragraph is competent, and not one sentence says anything you could argue with.

The single ask

You describe the whole deliverable in one message and ask for the finished thing back.

TRAITS

- One instruction, one output, one attempt
- You see nothing until all of it arrives
- Every judgement inside it was made without you

PROS

- + Fast, and on small work genuinely enough
- + Nothing to set up first

CONS

- Covers every heading, commits to none
- The only place to correct it is at the end

The sequence

You run the same job as 5 asks, each one taking the previous answer as its input.

TRAITS

- Each step has one job and one output
- You read and correct in the gaps between steps
- Later steps inherit only what you approved

PROS

- + Depth where the work is actually hard
- + A wrong turn costs one step, not the document

CONS

- Slower, and it needs you in the room
- Cut in the wrong place and the pieces will not assemble

Fig 16.1 The same deliverable, two routes. The difference is not how well you worded it. It is how many times you got to see and correct a decision before it was baked into prose.

16.1

The tell. Everything is covered and nothing is argued. If you cannot find a sentence to disagree with, the ask was too big.

What you really asked. One message, 6 different jobs, and no checkpoint between any of them.

The price. Roughly 5 messages and 20 minutes instead of 1 and 3. Spend it where the document gets read closely.

Here is the situation this chapter exists to fix. You have a quarterly board update to write, you have finally cleared 40 minutes, and you put the whole thing into one message: the quarter's numbers, the miss on renewals, where you stand against two competitors, and the headcount you intend to ask for. Back comes 900 words that are fluent, correctly ordered and completely weightless.

Nothing in it is wrong. That is the frustrating part. You cannot fix it with an edit, because there is nothing specific enough to edit, and you cannot say what you wanted instead, because what you wanted was for it to have decided something.

Look at what you actually asked for. Six separate jobs, wearing one sentence: recall the facts, work out what the story is, decide the order, write the difficult section, anticipate the pushback, and produce polished prose. All 6 at once, with no point at which anything could be checked with you.

The fix is not a better paragraph of instructions. It is the same work run as a short sequence, where each message does one of those jobs and hands its result to the next. You read what comes back at every stage, correct it in a line or two, and that correction travels forward instead of being discovered in the meeting.

It costs you 5 messages instead of 1 and about 20 minutes instead of 3. That is worth paying on work with your name on it that a room of people will read closely. It is not worth paying on a 200-word email, and section 16.4 draws that line precisely.

16.2

The thinness has 3 causes, and better models fixed only part of one

This is not laziness and it is not a badly worded prompt. Three separate things happen when a request gets large, and knowing which one is biting tells you where to cut.

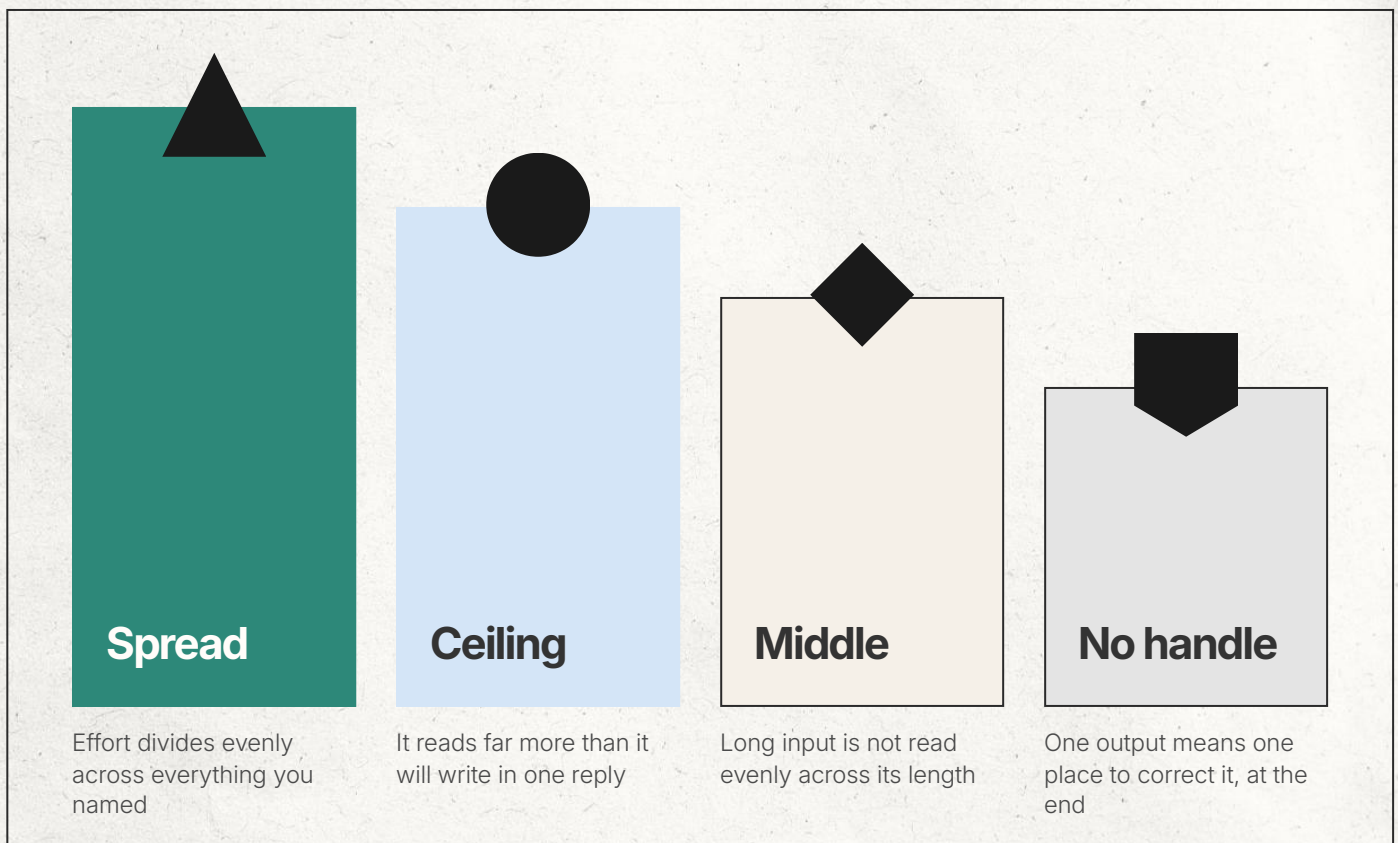


Fig 16.2 Three mechanisms, plus what they cost you. The bar heights are visual rhythm only, not a measurement: none of these is bigger than the others, and on any given job usually one of them is doing the damage.

16.2

Spread. Name 6 things and it gives each of them roughly a sixth of the effort. The trivial point and the load-bearing point get the same 3 sentences.

Ceiling. Every assistant will take in far more than it will put out in one reply. Ask for a whole document and you are asking it to compress.

Attention. More context is not more attention. Accuracy falls as the input grows, on every model tested, not just the older ones.

The first cause is spread. Asked to cover 6 things, an assistant divides its effort roughly evenly between them, because nothing in your request said which one carries the argument. Researchers describe this as breadth without prioritisation: the throwaway point and the load-bearing point are treated as equals. A human writer solves it by deciding what the piece is about before writing a word. Unless you make that decision a step of its own, it never gets made.

The second is the ceiling on a single reply. Every one of these assistants will read far more than it will write. The same tool that takes a 200-page contract in its stride will produce a few thousand words in one go, and long-form benchmarks through 2026 keep finding the same pattern: models fall short of explicit length targets, and the shortfall grows the longer the target gets. Ask for the whole document and you are asking it to compress. Ask for one section and you are asking it to write.

The third is that a long input is not read evenly. Chroma's context rot study tested 18 frontier models and found every one of them getting less accurate as the input grew, with the steepest fall in the range people reach when they paste in everything they have. In a separate evaluation of 4 leading models, accuracy on a simple two-step question dropped from between 82% and 99% to between 22% and 70% once that same question sat inside 32,000 tokens of surrounding material.

Now the honest part, because this advice has genuinely weakened in one place. Models reason internally now in a way they did not three years ago, and much of the step-by-step scaffolding people used to type by hand is done for you: that is Chapter 15's territory, and it is real progress. What has not changed is the rest. Context windows in the millions did not stop the degradation, and a larger window has never once cured the thinness.

Anthropic's own current guidance says it plainly: the value of splitting a task is no longer mainly about fitting inside a context window, it is about letting the model do its best work inside one narrow set of requirements. That is the reframing worth carrying. You are not breaking work up because the machine cannot hold it. You are breaking it up so you can see and correct each decision before it is set in 900 words of prose.

16.3

Five messages, one board update

Here is the same deliverable run properly, start to finish. The sequence itself is the thing worth keeping: it transfers to a competitive review, a customer research synthesis, or any strategy memo you have to defend.

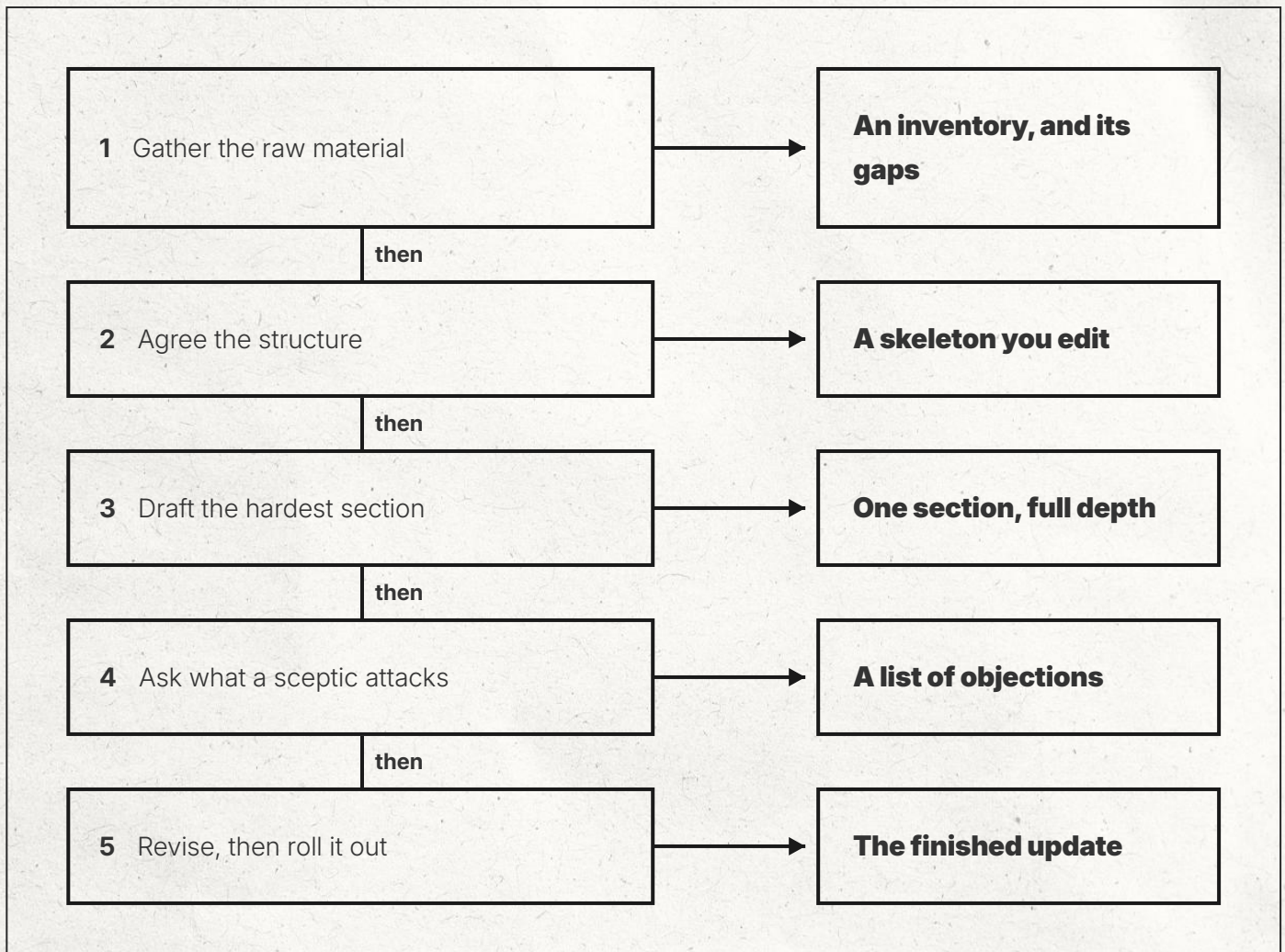


Fig 16.3 The reusable sequence. Every right-hand box is the input to the next step, and none of them except the last is prose. If a step hands you a draft when you asked for a list, you skipped a decision.

16.3

The gaps do the work. What makes this a chain is that you read and correct between messages. Fire all 5 without looking and you have only typed more.

One thread. Run the whole sequence in a single conversation so each step can see the last, which is why Chapter 4 keeps one thread per topic.

Hardest section first. It is the one most likely to change the structure, and once written it becomes the sample the easy sections imitate.

Message 1 gathers. Give it what you have, the numbers, last quarter's deck, the renewals report, and ask for an inventory rather than a draft: which facts are in this material, which of them a board would consider load-bearing, and what is missing that a board would expect to see. What comes back is a list. Reading it takes 2 minutes, and it usually surfaces at least one gap you would otherwise have met for the first time in the meeting.

Message 2 agrees the structure. Working only from that inventory, ask for the section list and, for each section, the single sentence it has to land. This is the step people skip and it does more work than any other, because it is where the document acquires an argument rather than an outline. Cut sections, reorder them, rewrite two of the sentences in your own words. What you approve here is what every later step inherits.

Message 3 drafts one section, and it should be the hardest one: the renewals miss. In full, at the depth you would defend in the room. You come out with two things, a section you can actually use, and a specimen of what good looks like in this document, which the remaining sections can be told to match. Chapter 14 is where showing it a model answer came from. This is that technique pointed at the document's own material.

Message 4 attacks it. Give the drafted section back and ask what the most sceptical person at the table goes after first, and what they would ask that this section does not answer. You want objections, not a rewrite, so say so. Chapter 21 is entirely about getting genuine criticism out of something whose instinct is to agree with you. Here you need one honest pass, not the full treatment.

Message 5 revises against those objections, and only then do you roll out the remaining sections against the approved skeleton and the approved specimen. Notice what the order protected. By the time any easy section gets written, the structure is settled, the difficult section exists, and the objections are already known.

That is a different document from the one that arrives in a single reply, and the sequence behind it is 5 words long: gather, structure, draft, critique, revise. Write those 5 down. They are what Chapter 17 turns into something you can run again in a fraction of the time.

16.4

Cut where you would hand the work to a person

Splitting a job is easy. Splitting it in the right places is the actual skill, and a bad cut is worse than no cut, because you end up assembling pieces that were each written as though the others did not exist.

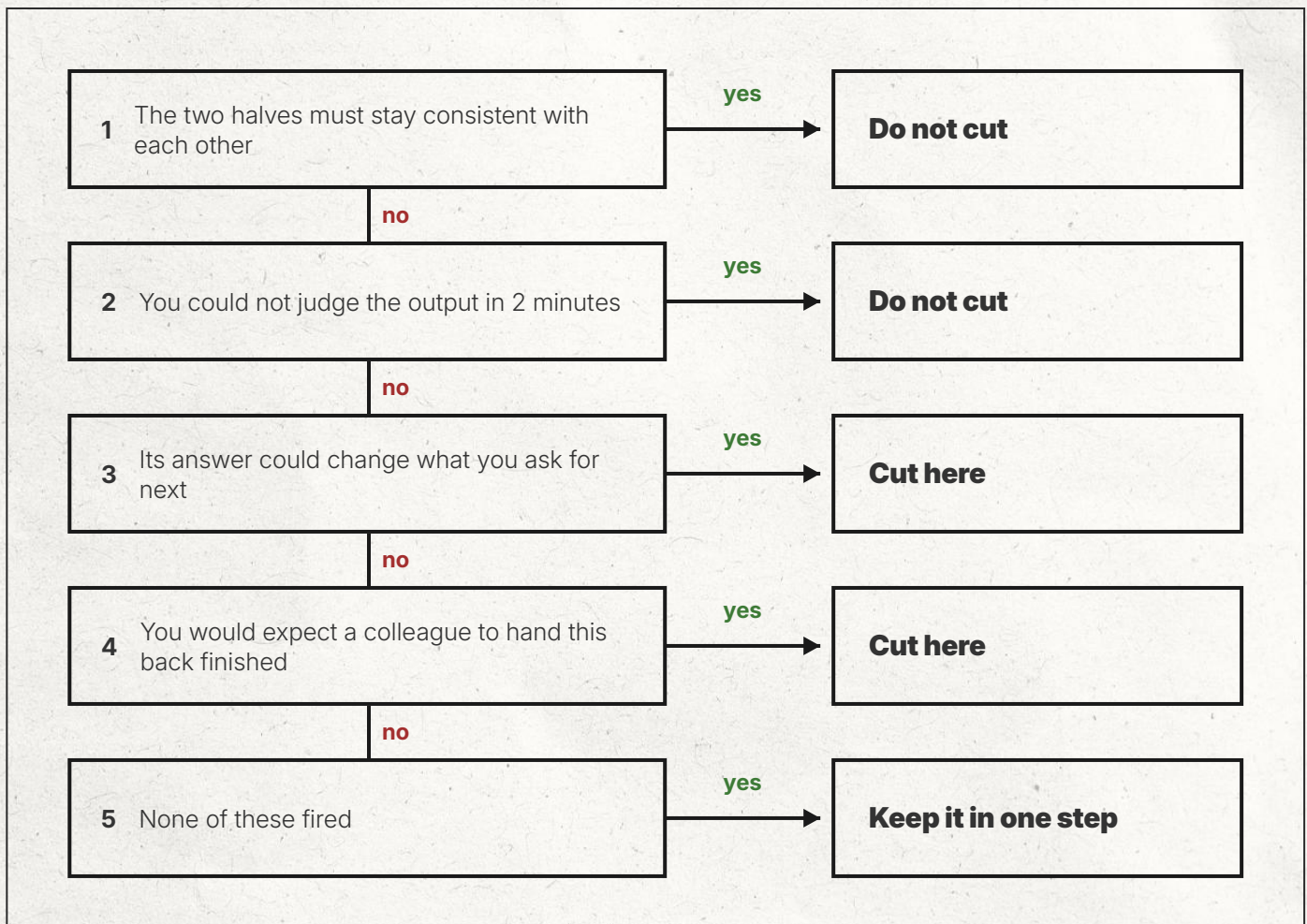


Fig 16.4 Read down until a line is true. The two vetoes come first on purpose: a cut in the wrong place costs you more than a cut you failed to make, because you only find out at assembly time.

16.4

Cut at decisions. If the answer to a step could change what you ask for next, it has to be its own step. That is the moment you need to see it.

Never split a comparison. Two competitor profiles written in separate steps will sit next to each other. They will not compare.

Five is a working size. Fewer than 3 and you have not really split it. Past 8 or so a chain starts losing the thread it began with.

The most common bad cut is by section. You ask for section 1 in one message and section 2 in the next, and you get two documents: two openings, the same background explained twice, two different registers, and no argument running through either. Nothing in the second message knew what the first one decided.

The cut that works falls where a decision gets made. Gather and structure are separate steps for exactly that reason: what the inventory turns up changes the skeleton, and the skeleton changes everything after it. The practical version of the test is to ask whether you would give this piece to a capable colleague and expect something finished back. If the answer is yes, it is a step.

Two things veto a cut, and the veto wins. If the halves have to stay consistent with each other, in register, in a running argument, or in a comparison across several items, keep them in one step so both are in front of it at the same time. And if you could not judge a step's output in about 2 minutes, it is not a checkpoint, it is a delay: you have bought waiting time and no control.

You can also cut too far. Long chains drift. Each step carries a little less of the original intent, a mistake in step 3 quietly travels into everything downstream, and by step 10 you are administering a process instead of writing a document. Five steps is a comfortable size for a real deliverable. If you find yourself at 12, you have cut along seams that were not there. Automating a chain across several different applications is a genuinely different problem, and it is Chapter 33's.

None of this applies to a 200-word email. One well-formed prompt from Chapter 13 handles that, and reaching for a 5-step sequence there is a way of looking busy. Keep it for work that has a shape: sections, an argument, and someone who will read it closely.

Take the piece of work you brought into this chapter and mark where the first cut goes. Not all 5 steps yet. Just the first one, and what you expect it to hand you.

READING ENDS HERE

Everything before was reading. Everything after is doing.

3 exercises, About 25 minutes and one real decision that opens Chapter 17.

Do these before Chapter 17

3 exercises. About 25 minutes

Name the big job. Write down one deliverable from your real work that you have been tempted to ask for in a single message, and the one sentence the whole thing has to land.

Make the first cut for that deliverable. Fill in all three, and check the middle box is a list or a skeleton, not prose.

STEP 1 ASKS FOR	WHAT IT HANDS OVER	WHY STEP 2 NEEDS IT

Now run steps 1 and 2 only: gather, then structure. Write one thing the inventory told you that you did not already know, and one thing you changed in the skeleton by hand.

Chapter 16 on one page

THE 5-STEP SEQUENCE

Gather · structure · draft the hardest section · critique · revise and roll out. Each message takes the previous answer as its input

WHY ONE BIG ASK THINS

Effort spreads evenly across everything you named, a single reply has a length ceiling far below what it can read, and long input is not read evenly

WHERE TO CUT

At a decision whose answer changes what you ask for next, and where you would expect a colleague to hand back something finished

WHERE NOT TO CUT

Across halves that must stay consistent with each other, and anywhere you could not judge the output in 2 minutes

NEEDED BEFORE CHAPTER 17

One sequence you have actually run, with its steps written down in your own words

BRING TO CHAPTER 17

The sequence you ran and the step names you wrote down. Chapter 17 stops you rebuilding that sequence by hand every time you need it.

BOTTOM LINE

You can spot a job that is too big for one message, run it as a sequence where each step feeds the next, and place the cuts so the pieces assemble.

NEXT CHAPTER

Turn Today's Prompt Into a Template

Chapter 17 takes the sequence you just ran and makes it permanent. Prompts with slots you fill in, kept somewhere you can find them, so a chain you worked out once takes 5 minutes to run next quarter instead of an afternoon.

BRING WITH YOU

- ☐ The 5 step names from the sequence you ran
- ☐ The one message in it you had to rewrite most often

GO FURTHER

The Ultimate AI Library

Every guide, prompt pack and playbook AI Central has published, in one place. The reference shelf this course keeps pointing you to.

explore the library



WHO MAKES THIS



AI Central Media is a London-based editorial team. Our flagship publication covers practical AI for senior professionals, and 100+ AI companies, SaaS platforms and growth teams have advertised with us since 2023.

100+

AI companies, SaaS platforms, education brands and growth teams have advertised with us since 2023

613K

Accounts reached every month, across the 7 channels we operate

151

Countries our readers are in, plus all 50 US states

PROUD PARTNER OF

GAMMA

II ElevenLabs

guide.

 Outskill

 **Luma AI**

HubSpot

 **attio**

S Synthflow

 Notion

 **Replit**

We turn attention into pipeline.

visit thecentral.ai

