The Wordsmith Agent prompting guide.

How to get the best work out of your Wordsmith Agents — and the four building blocks that make them powerful.

Elly Meenan
OWNER

THE WHEN **Agents** The worker that decides *when* to act, in what order, and when it's done.	THE HOW **Skills** The packaged method that tells it *how* to do a task (calling a **Playbook** or **Blueprint**).
WHAT IT KNOWS **Repositories** Your documents, synced and stored in Wordsmith as a library the Agent can reference.	THE WHERE **Connectors** Live tools to read and act in outside systems in the moment.

Start here

It's worth keeping these building blocks in your head. If something goes wrong, you can usually trace it back to one of them: the agent acted at the wrong time (a **when** problem), it did the task the wrong way (a **how** problem), it didn't know something it should have (a

knowledge problem), or it couldn't reach the right system (a **where** problem). Different problems require different fixes.

The rest of this guide walks through each block, then shows how to prompt across all of them.

Before you start

This guide assumes you have:

1. **A clear outcome in mind.** You know what "done and correct" looks like — a redlined contract back in Slack, a filled blueprint stored in Google Drive, a ticket on a Jira board.
2. **A way to tell whether the output is right.** A standard, a checklist, or a reviewer who can sign off.

If you don't have those yet, spend the time there first. The single biggest lever on output quality is a clear definition of success — just like a human, an Agent can't hit a target you haven't described.

TWO QUICK RESOURCES

No idea how to phrase the ask? State the goal in plain English and let Wordsmith propose the steps back to you — or allow it to help you build it. Refine from there.

Want house-standard output every time? That's a job for a Skill, not a longer prompt. See *Skills — the how* below.

When prompting is the right fix (and when it isn't)

Not every disappointing result is a prompting problem. Before you rewrite the prompt, locate the failure in the blocks:

- **Wrong thing at the wrong moment** — jumped to drafting before gathering facts, or stopped early. → A **when** problem. Fix the instructions to the agent: sequence, goal, and stopping condition.
- **Right task, not to your standard** — off-brand tone, missing a required section. → A **how** problem. Encode the standard in a Skill rather than re-explaining it each time.

- The Agent **didn't know a document it should have** — missed a precedent, a policy, a signed template that lives in your library. → A **knowledge** problem. Check what's in the **Repository**.
- **Couldn't see the non synced document, ticket, or record it needed.** → A **where** problem. Check the connector, not the prompt.

BLOCK 1 · THE WHEN

Agents — the when

A Wordsmith Agent is the worker. Give it a goal, and it decides what to do, in what order, which skill to reach for, which system to pull from, and when the job is finished. Each one carries a role, a remit, and judgment about how to pursue it.

The defining trait of an agent is **autonomy over sequence**. You are not clicking through steps; you are handing over an objective and trusting the agent to orchestrate.

What you control when you prompt the agent:

- **The objective** — what success looks like, stated as an outcome.
- **The order and priorities** — what to do first, what matters most, what to skip.
- **The guardrails** — what it must not do, when to stop and ask you, what needs human sign-off.
- **The finish line** — how it knows it's done and what "done" should look like.

Prompt the agent like you'd brief a capable new colleague:

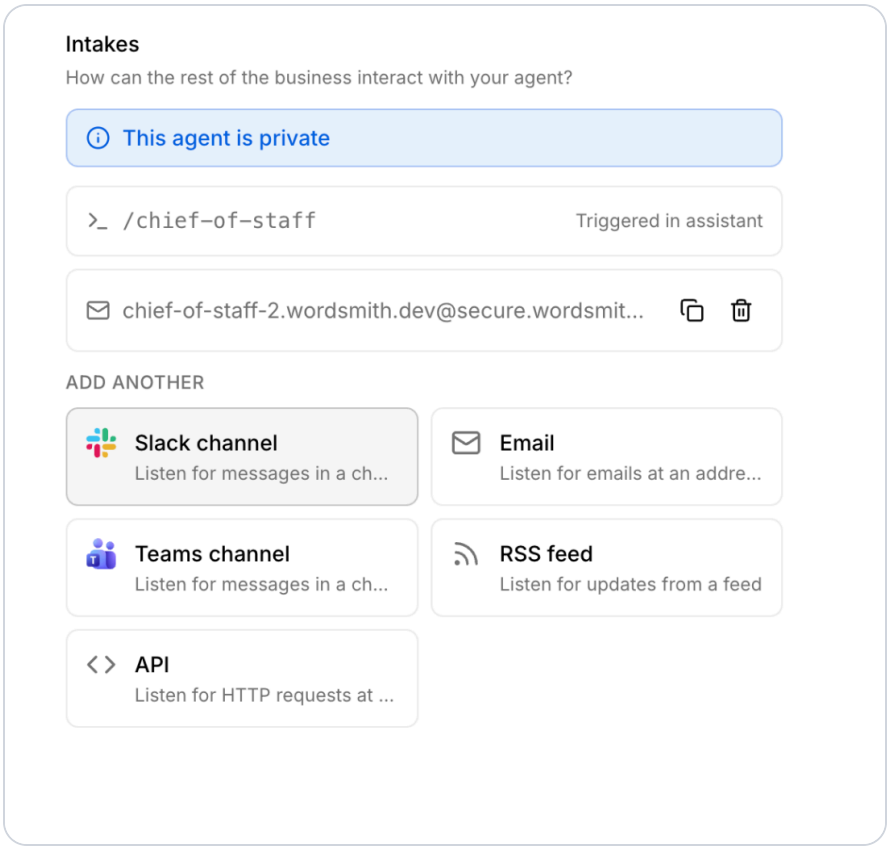
EXAMPLE AGENT PROMPT

```
You are a "Contract Triage & Routing" agent that triages every inbound
contract for the Wordsmith legal team and routes it by value. For each
contract submitted, follow this router:

1. DETERMINE VALUE. Read the contract and establish its total value.
2. UNDER GBP 50,000 — apply the "Redline sub-50k contract" skill to redline
the contract.
3. GBP 50,000 OR MORE (including exactly GBP 50,000) — apply the "Route
over-50k contract for legal review" skill.

Always state which branch you took and why: the value you found, and —
for over-50k contracts — the assignee chosen and whether a DPA sub-task
was raised.

Never assign tasks to anyone other than Elly, Gigz or Emily Spooner.
If a submitted document is clearly not a contract, say so and take no
triage action.
```

A complete agent prompt: role, decision procedure, reporting, guardrails — and nothing else.

Notice how cleanly this prompt maps to the building blocks:

- **It steers the when.** It sets the role, the decision procedure (determine value first, then branch), the reporting requirement, and hard guardrails (only these three assignees; do nothing if it isn't a contract). Sequence, priorities, and stopping rules — exactly what you control at the agent level.
- **It delegates the how to skills.** It doesn't spell out how to redline a sub-50k deal — it names the skills and lets each one carry the house standard. That's why the prompt can stay short: the expertise lives in the skills.
- **The where is implicit.** This example uses no connector — it rides on Wordsmith's deep Slack integration. For Slack, Email, Teams and Zendesk you configure intake in the "Intakes" section of Agents.



Intakes — how the rest of the business reaches your agent: a Slack or Teams channel, an email address, an RSS feed or the API.

BLOCK 2 · THE HOW

Skills — the how

A Skill is a packaged capability: instructions, templates, examples, and sometimes small scripts that teach the Agent how to do a specific kind of task the way you want it done.

The mental model that unlocks skills: a skill is an onboarding guide you write once for a new team member.

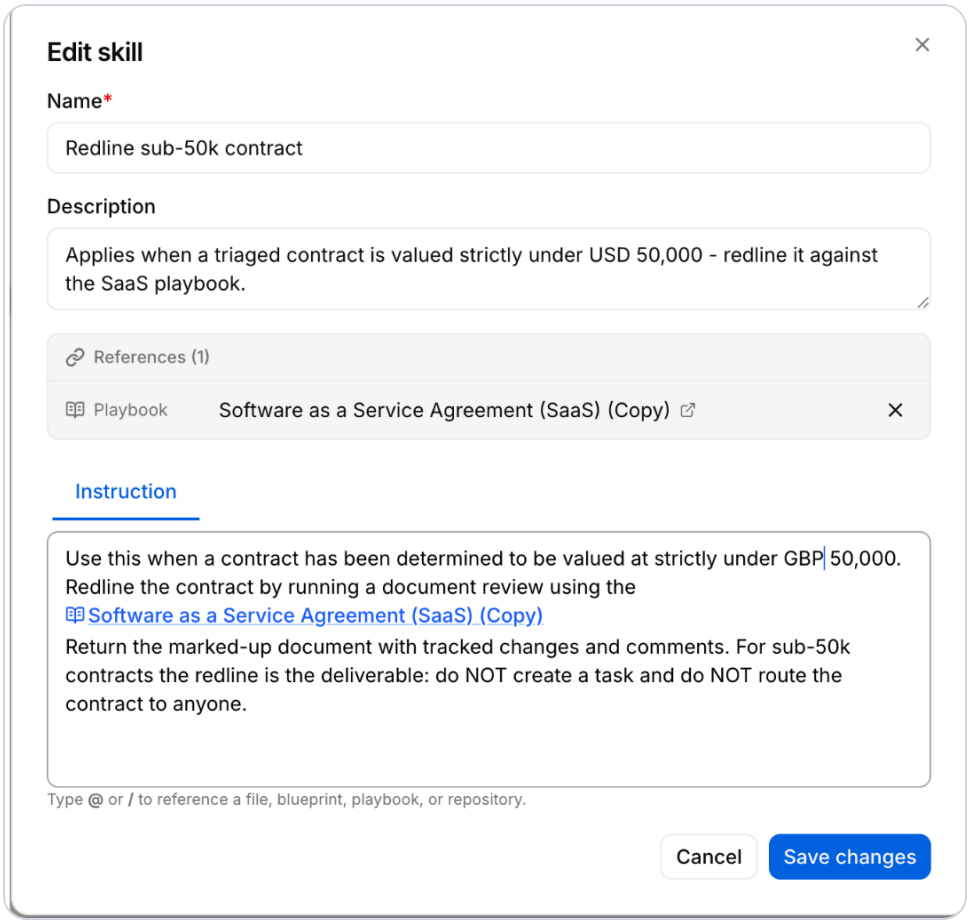
Why skills beat longer prompts. A prompt is a one-off instruction for a single conversation. A skill is reusable expertise that applies automatically whenever it's relevant — no copy-pasting the same three paragraphs of guidance into every chat. Create it once; the Agent reaches for it on its own.

How skills load — and why that keeps the Agent fast. Skills use progressive disclosure: the Agent doesn't swallow the whole thing upfront. It loads in stages:

STAGE	WHAT LOADS	WHEN
1 · Name + description	A one-line summary of what the skill does and when to use it	Always — so the Agent knows the skill exists
2 · Instructions	The body: workflow, standards, guidance	Only when your request actually triggers it
3 · Resources	Blueprints, Playbooks, Repositories (Files) — add these to a task by typing "@" or "/"	Only the specific things a given task needs

The payoff: you can equip an Agent with dozens of skills without slowing it down or crowding its attention. It only pulls the relevant one into focus — the way a colleague pulls the right binder off the shelf instead of reading the whole library.

Not sure where to start with a skill? Ask Wordsmith to build it for you — or start from Wordsmith Shared Skills.



The "Redline sub-50k contract" skill — a precise name and "when it applies" description, the SaaS playbook it references, and the instruction.

What a skill encodes (the method):

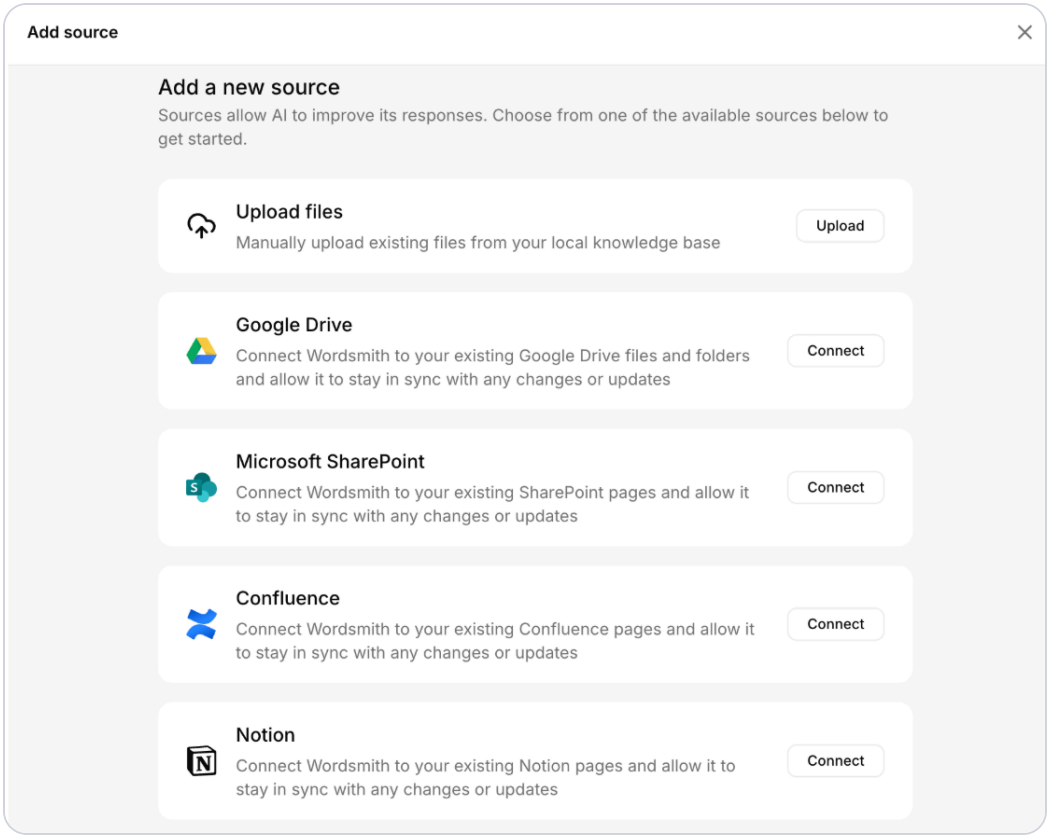
- **The workflow** — the steps a task should follow, in order.
- **The house style** — tone, structure, formatting, required sections.
- **Which playbook, blueprint or repository to consult** — and when to apply it.
- **The escalation rules** — what gets flagged and to whom.

SIGNS OF A HOW PROBLEM

The output is structurally right but wrong on the details. Diagnose one level deeper — is the **method** wrong (skipped a step, wrong order) or is the **standard** wrong (accepted a position you'd reject, wrong fallback)? A broken method is a skill fix; a wrong position is a playbook fix. Either way, don't patch it with a longer prompt each time — encode it so it applies automatically and identically for everyone on your team.

Repositories — what it knows

A Repository is your library inside Wordsmith. Documents are synced from Google Drive, SharePoint, Notion or Confluence and indexed as Knowledge the Agent can reference whenever it's relevant — precedents, templates, executed agreements, policies or docs. Think of it as the shelf of binders the Agent reads from.



Adding a source — upload files directly, or connect Google Drive, SharePoint, Confluence or Notion and Wordsmith keeps them in sync.

It's searchable, it's there every time without re-fetching, and it stays current as the source updates. This is what the Agent *knows* — its standing memory of your documents.

What a Repository gives the Agent:

- **A durable reference library.**
- **Grounding** — answers and drafts anchored to your documents rather than the Agent's general training.

- **Reuse across every conversation** — sync once; agents, reports and chats can all draw on it.

Signs of a knowledge problem: the Agent's answer is off, or misses something it should have known. That's usually not a prompting fix — the right document may not be in the Repository, or the sync is stale. Check what's synced.

REPOSITORY VS. CONNECTOR

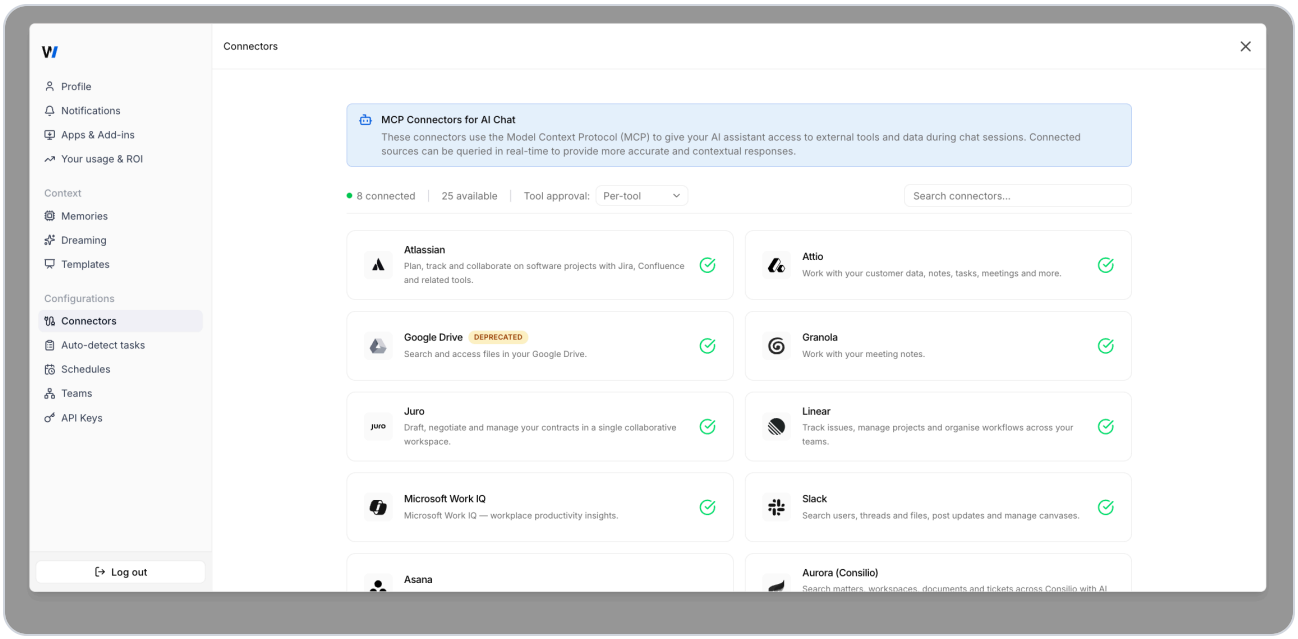
A Repository stores a synced copy of documents in Wordsmith (**persistent knowledge**). A Connector reaches a live system in the moment and stores nothing — data is fetched live from its original source through an MCP server. Reach for a Repository when the Agent needs to know your standing documents; reach for a Connector when it needs live, current data (like a ticket) or needs to act.

BLOCK 4 · THE WHERE

Connectors — the where

A **Connector** is how an Agent reaches out of the chat and into a live system in the moment — email, Slack, calendar, ticketing, a CRM. **MCP (the Model Context Protocol)** is the open standard that makes those connections work: a common language that lets the Agent talk to an outside system without a custom integration for each one. A Connector gives the Agent *tools* to use — search this, read that, create the other.

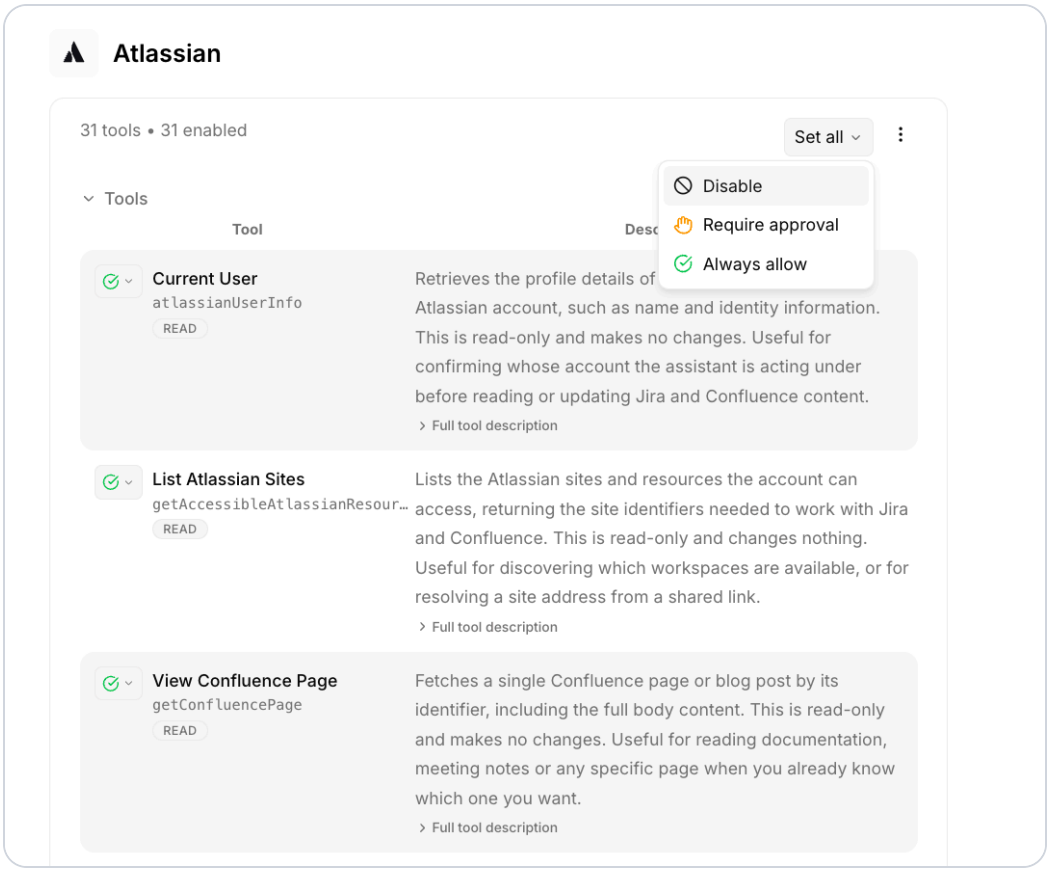
The defining trait, and the thing that separates it from a Repository: **a Connector doesn't sync or store anything.** It reads or acts live and moves on. When the task ends, there's no retained copy in Wordsmith — just the result of what it did.



The Connectors panel — MCP connectors give the Agent live access to outside systems, with per-tool approval you control.

What a Connector lets the Agent do:

- **Read** live data — the current inbox, today's calendar, the real-time ticket status — instead of relying on what you paste in or a stored snapshot.
- **Act** in a system — draft a reply, raise a ticket, update a record, find a message — when you've granted it permission to.

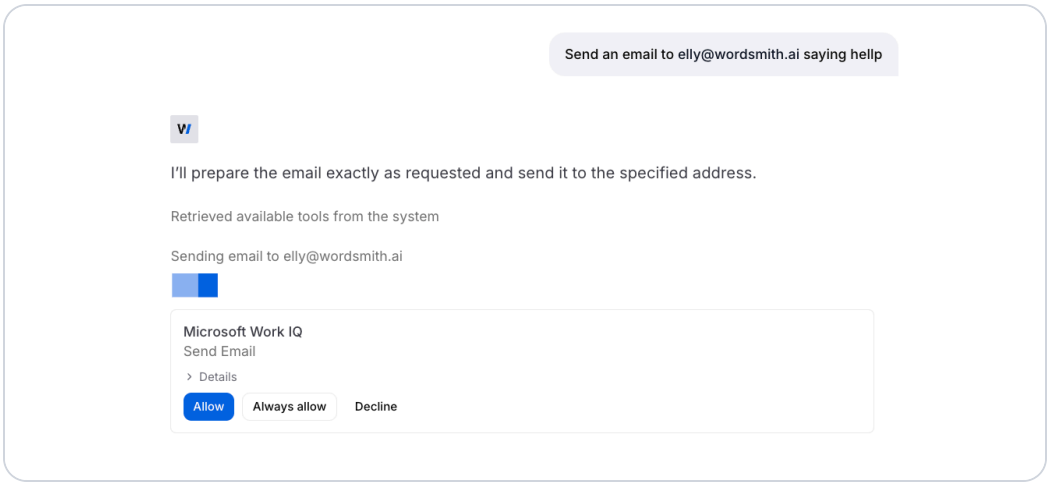


A connector exposes named tools — here Atlassian's read tools — each one something the Agent can call.

You control access deliberately. Connectors aren't all-or-nothing. Access is configured per tool:

- **Allowlist** — grant only specific, named capabilities (e.g. search and read only). Everything else stays off.
- **Denylist** — grant broad access but explicitly switch off the risky ones (e.g. allow reading and drafting but block delete and send).

For anything that changes state — sending, filing, deleting — Wordsmith defaults to a human confirmation step unless you explicitly change this.



Before a state-changing action runs, the Agent pauses for your sign-off — Allow once, Always allow, or Decline.

Signs of a reach problem: the Agent says it "can't access" a live system, or can read but can't write back. This is almost never fixed by rewording the prompt — check whether the Connector is live and authorized, and whether the specific capability you need is switched on.

NOTE ON SETUP

Connecting a system requires authorizing it once (an OAuth sign-in) in your connector settings. If an Agent reports it can't reach a live system, the first thing to check is whether that Connector has been authorized — not the prompt.

How the four work together

A single request can (but not always) touch all four blocks.

YOU ASK

"Check whether we've signed anything with Contoso, and if their MSA is up for renewal in the next 60 days, draft the non-renewal notice in our standard form. Show me before anything goes out. Also, can you find our Contoso AM's email address?"

Here's what each block contributes:

1. **Agent — the when.** Decides the sequence on its own: first find what's signed, then check renewal timing, then draft only if the 60-day condition is met — and stops before sending, because you said so.
2. **Skill — the how.** Runs the workflow — the right structure, in the right order. Surfaces the executed Contoso MSA from your synced library (**repository - knowledge**), then calls your non-renewal Blueprint for the exact house language, so the draft is right the first time, not a generic letter.
3. **Connector — where it reaches.** Checks your email for the AM's address, can't find it there — so checks your CRM. Bingo.

Your prompt didn't have to spell out any of the specifics. You stated the outcome and the guardrail, and the agent drew on the right documents, live data, method and standard automatically. That is the whole point: **you steer the when, and let well-configured skills, playbooks, repositories and connectors handle the rest.**

Building an Agent: the prompt vs. the skill

Everything above is about *using* Agents. This section is about *building* one — an agent prompt and a skill are written differently because they do different jobs.

- **An agent prompt is a brief.** It reads like a job description crossed with a routing policy: who you are, when you act, how you decide, who you hand off to, what you're not allowed to do. It stays high-level and stable.
- **A skill is a procedure.** It reads like a standard operating procedure: given you're doing this task, here are the exact steps, the format, and the playbook to apply. It carries the depth.

Put crudely: the agent decides and delegates; the skill does the work. If you catch yourself writing detailed how-to inside an agent prompt, that's the signal to stop and pull it into a skill.

Anatomy of a good agent prompt

A strong agent prompt has these parts.

PART	WHAT IT DOES	IN THE TRIAGE EXAMPLE
1 · Role	One line: who the agent is and what it's for	"You are a Contract Triage & Routing agent…"
2 · Scope / trigger	What it handles — and what to ignore	"…for every inbound contract"; "if not a contract, take no action"
3 · Decision procedure	The routing logic: steps and branches, in order	"Determine value → under 50k → 50k or over"
4 · Delegation	Which skill to call at each branch	"apply the 'Redline sub-50k contract' skill"
5 · Reporting	What the agent must tell you about its choices	"Always state which branch you took and why"
6 · Guardrails	Hard limits, approvals, and stop points	"Never assign to anyone but Elly, Gigz or Emily"
7 · Finish line	What "done" looks like	Implicit: routed, reported, nothing sent without review

Notice what is **not** in the agent prompt: how to actually redline a sub-50k contract. That lives in the skill (inside the playbook). The agent prompt names the skill and moves on. That separation keeps the agent prompt short, readable and stable while the real expertise sits in skills you can update independently.

A basic FILL-IN-THE-BLANK AGENT PROMPT

```
You are a [role] that [does what, for whom]. For each [input], follow
this procedure:
1. [first step / how to decide].
2. If [condition A] → apply the "[skill name]" skill.
3. If [condition B] → apply the "[skill name]" skill.
Always [what to report and why].
Never [hard guardrail].
If [out-of-scope case], [what to do instead].
```

Anatomy of a good skill

A skill is built around a different spine — it assumes the agent has already decided to run it, and focuses entirely on doing the task well:

- **Name + description** — a tight summary of what the skill does and when it applies. This is what the agent reads to know the skill exists, so write the "when" precisely.
- **The method** — the steps, in order, the way you want the task done every time.
- **Format and examples** — the structure of the output, with a model example if you have one.
- **What to call** — the playbook, the blueprint or the repository

Agent prompt vs. skill — side by side

	AGENT PROMPT	SKILL
Job	Decide when to act and what to run	Do the task the right way
Reads like	A job description / routing policy	A standard operating procedure
Contains	Role, scope, decision logic, delegation, guardrails	Steps, format, examples, which playbook to apply
Does not contain	The detailed how-to (that's the skill's job)	Broad mission or when-to-run logic (that's the agent's job)
Changes	Rarely — it's the stable policy	Whenever the method improves
Length	Short	As long as the task needs

THE EXTRACTION TEST

If it describes **when to act, what to prioritise, or what not to do**, it's agent. If it describes **how to do a task step by step**, it's a skill. If it describes **your position, standards or knowledge**, it's a playbook/repo/blueprint.

Prompting best practices

These apply no matter which Agent you're working with.

1. **Be clear, direct, and specific.** State the outcome, the audience, and the format. Vague in, vague out. Prefer "Summarize the indemnity clause for a non-lawyer in three bullet points" over "tell me about the indemnity."

2. **Give the Agent the context it needs.** Explain the why and the situation, not just the task. "This is going to the client's CFO, who cares about cost exposure" produces a sharper result than the same request with no framing.

3 **Show an example of what "good" looks like.** One example of the format, tone, or structure you want is worth a paragraph of description. If you have a model output, share it.

4 **Let the Agent think before it concludes.** For anything involving judgment — risk assessment, comparing positions, spotting issues — ask it to reason through the analysis before giving its answer. Rushed conclusions are worse conclusions.

5 **Structure long or multi-part prompts.** When a prompt has several distinct inputs (a contract, your standards, and specific questions), label the sections clearly so the Agent doesn't blur them together. Headings or simple tags like "CONTRACT:" / "QUESTIONS:" work well.

6 **Break big jobs into steps.** For complex work, have the Agent do it in stages and check in between — extract the facts first, confirm they're right, then draft. This catches errors early, before they compound.

7 **Set explicit guardrails.** Tell it what not to do and where to stop. "Don't send anything," "flag it instead of deciding," "ask me if you're unsure" — these are as important as the task itself, especially where a connector can act on your systems.

8 **Push the repeated how into a skill or playbook.** The most common prompting mistake is re-explaining the same thing on every request. If you're repeating the steps, that's a skill waiting to be built. If you're repeating the positions, that's a playbook. Encode it once and let the skill call the playbook.

Quick diagnostic — which block do I fix?

WHAT YOU'RE SEEING	LIKELY BLOCK	WHAT TO CHANGE
Acted out of order, stopped early, or overshot	Agent (when)	Tighten the objective, sequence, and stop points in your prompt
Right task, wrong steps / order / format	Skill (method)	Fix the workflow encoded in the skill
Wrong answer, or missed a known precedent	Repository	Add the document to the repository, or check the sync
Working from an out-of-date version of a document	Repository	Re-sync the source; the stored copy is stale
"I can't access that system," or needs current live data	Connector	Check the connector is authorized and enabled
Can read but can't file / send / update	Connector	Enable the specific action capability (deliberately)
Answers a general question from knowledge instead of looking it up (or vice versa)	Connector	Fine — it's distinguishing knowledge from live lookup by design

THE ONE-LINE SUMMARY

Agents decide when. Skills define how. Repositories are what it knows. Connectors are where it reaches.

Prompt the agent for the outcome; encode the how in skills and playbooks so it's consistent; sync your documents into repositories so it works from the truth; connect live systems so it can see current data and act. Get those right and prompting stops being a battle and starts being a briefing.