

BUILDER · 6 MINUTE READ

Hooks: automatic rules that run at the right moment

A hook is a small rule that runs by itself when something happens in an AI setup, so important checks never depend on memory.

What it is

A **hook** is a rule that runs on its own when a certain moment arrives in an AI tool or agent setup. The moment might be just before the AI uses a tool, just after it changes a file, before a message goes out, or at the end of a working session. When that moment comes, the hook runs a check, blocks an action, writes a note or reminds the AI of a rule.

Think of the beep a car makes when you forget your seatbelt: you do not have to remember, the car checks at the right moment.

Hooks are different from instructions. An instruction asks the AI to behave in a certain way, and the AI may forget or misread it. A hook is run by the software around the AI (the harness, see guide 23), so it happens every time, whatever the AI decides.

Some agent and coding tools support hooks, usually through a settings file; others offer similar controls under different names.

Why it matters

- Rules you do not have to remember: the check happens even on a busy day.
- A blocking hook stops a risky action before it happens, not after.
- A log of every action makes it easy to see what an agent did and when.
- Hooks support people; they do not replace the person who reviews and approves.

How to do it

1. **List** the moments in your setup where something could go wrong: sending, publishing, deleting, changing shared files.
2. **Pick** the riskiest action and write one blocking rule for it, such as "never publish files online without a person approving".
3. **Write** the hook in plain words first: when [moment], check [condition], then [block, warn or log].
4. **Test** it on purpose with made-up data: try the blocked action and confirm the hook stops it.
5. **Add** gentler hooks later, such as a spelling and style check before anything is sent, or a log of every action.
6. **Keep** each hook short, with one job.
7. **Name** an owner who maintains the hooks, reviews them on a fixed schedule, for example every three months, and removes ones that no longer help.

Example

A team uses an agent to draft customer replies and save reports. They write their hooks in plain words first:

```
Hook 1 (before any tool is used):
  If the action is to publish, upload or push files online,
  BLOCK it and show: "A person must approve publishing."

Hook 2 (before a message is sent):
  Run the spelling and house-style check on [draft].
  If it fails, stop and show the problems.

Hook 3 (after any file is changed):
  Write the file name, time and action to [activity log].

Hook 4 (end of session):
  Ask the agent to list what it did, what failed
  and what a person should review.
```

Before: the team relied on memory, and one draft went out with the wrong product name.

After: the style check runs every time, publishing is blocked until a person approves, and the end-of-session summary tells the reviewer where to look. A named person still reads each reply before it goes out.

Common mistakes

- **Starting with ten hooks.** Fix: begin with one blocking rule for the riskiest action, then add more.
- **Never testing the block.** Fix: try the forbidden action with made-up data and confirm it is stopped.
- **Complicated hooks nobody understands.** Fix: one hook, one job, written in plain words first.
- **Treating hooks as the only check.** Fix: keep a person reviewing anything that is sent, published or paid.
- **No owner.** Fix: name who maintains the hooks and reviews them on that schedule.

Try it this week

1. Write down three moments in your AI work where a mistake would do real harm (3 minutes).
2. For the worst one, write a hook in the pattern: when [moment], check [condition], then [block, warn or log] (5 minutes).
3. Show it to a colleague and ask: "What would this miss?" (4 minutes).
4. If your approved tool supports hooks, ask your technical contact to set it up and test it with made-up data (3 minutes to ask).

Going further

- Combine hooks with permissions (see guide 26): permissions say what is allowed, hooks add checks at the moment of action.
- Use an end-of-session hook to list possible lessons for a person to review and add to the lessons log (see guide 24).
- Make hooks fail safe: if a check cannot run, block the action rather than let it through.
- Rerun your tests (see guide 27) after changing a hook, since a broken hook can block good work or let bad work pass.

Key words

- **Hook:** a rule that runs on its own at a set moment in an AI setup.
- **Trigger:** the moment that starts a hook, such as "before a message is sent".
- **Blocking hook:** a hook that can stop an action from happening.
- **Activity log:** a record of what the AI did and when.
- **Fail safe:** designed so that if something breaks, the safe option happens.