Controlling Agent Behavior

The guardrails that make running an agent loop safe: step limits, human-in-the-loop approval, least privilege, and observability.

Intermediate 18 min

Last updated: August 22, 2026
Read online: https://www.learnforgex.com/en/topics/controlling-agent-behavior

---

Controlling Agent Behavior

"What Is an AI Agent?" placed every agent somewhere on "The Autonomy Spectrum" -- and noted that more autonomy means more decisions made without a human checking each one. "Agent Planning and Reasoning Patterns" showed that even a well-designed decision step can, in principle, loop forever without a termination condition. This lesson covers the practical mechanisms that make running an agent with real autonomy survivable: step limits, human approval points, restricting what an agent is even able to touch, and being able to see what it actually did.

Why Agents Need Guardrails

A single tool call is easy to reason about: one request, one response, one human decided to send it. An agent loop is different -- it can take many actions in sequence, chosen by the model rather than a human, and each of those actions might be a real tool call with a real side effect (sending an email, modifying a record, spending money). The same thing that makes agents useful -- deciding and acting without a human in the loop for every step -- is also what makes an unconstrained agent risky: a decision step that's wrong, or a tool call aimed at the wrong argument, doesn't get caught by a human before it runs. Guardrails exist to keep the useful part of that autonomy while bounding the damage a mistake can do.

Step and Iteration Limits

"Termination: Knowing When to Stop" in "Agent Planning and Reasoning Patterns" already introduced the idea: an agent should stop when the model concludes the goal is met, but it also needs a hard backstop for when that never happens. A **step limit** (or iteration limit) is exactly that backstop -- a maximum number of decide-act cycles the loop is allowed to run, enforced in code rather than left up to the model's own judgment. Once the limit is hit, the loop stops unconditionally, whatever it was doing -- this is a deliberately blunt mechanism,

and that's the point: it doesn't try to be clever about whether the agent is "almost done," it just guarantees the loop cannot run away indefinitely. "Building an AI Agent in TypeScript" implements exactly this kind of limit around its own agent loop.

Human-in-the-Loop: Approval Before Risky Actions

Not every action an agent might take carries the same risk. Looking something up is easy to undo (or rather, has nothing to undo); sending a message, deleting a record, or spending money is not. **Human-in-the-loop** means routing specific actions -- typically ones that are costly, irreversible, or hard to verify automatically -- through an explicit approval step before they execute, rather than letting the loop run them automatically like any other tool call. This doesn't mean every action needs approval, which would defeat the point of having an agent at all -- it means deliberately marking the subset of actions where a wrong decision is expensive enough that a brief pause for a human to confirm is worth the lost speed, referring back to where a given agent sits on "The Autonomy Spectrum".

Scoping Tool Access: The Principle of Least Privilege

An agent can only take actions its available tools allow -- so one of the most direct ways to bound what can go wrong is to bound the tool list itself. Giving an agent a tool that can delete any record, when its actual job only ever requires reading records, creates a risk that has nothing to do with how good the agent's decision-making is -- it exists purely because the capability was available to misuse. The **principle of least privilege** -- granting exactly the access a task needs, and no more -- applies to an agent's tools the same way it applies to a user account's permissions: narrower tool access doesn't make the agent's reasoning any better, but it does shrink the space of damage a bad decision (whether from a flawed decision step, or an unexpected input) can cause.

Observability: Logging and Tracing an Agent's Decisions

Because an agent's exact sequence of actions isn't scripted in advance -- that's the entire point of "The Agent Loop: Observe, Decide, Act" from "What Is an AI Agent?" -- knowing what it actually did after the fact depends entirely on having recorded it. **Observability**, here, means logging each step of the loop: what the decision step chose, what arguments it used, what the real tool result was, and how many steps ran before it stopped. Without this, a wrong final answer is nearly impossible to debug -- there's no way to tell whether the model reasoned incorrectly, called the right tool with the wrong arguments, or got a correct result and misused it. "Building an AI Agent in TypeScript" prints exactly this kind of step-by-step trace for its own agent loop, for this reason.

Best Practices

- Set a step limit on every agent loop, even ones that seem unlikely to misbehave -- see "Step and Iteration Limits" -- a guardrail that's rarely triggered still isn't wasted.
- Decide which actions need human approval based on cost and reversibility, not on how complex the action looks in code -- see "Human-in-the-Loop: Approval Before Risky Actions".
- Grant an agent's tools by what its task genuinely requires, and re-check that list as the task changes -- see "Scoping Tool Access: The Principle of Least Privilege".
- Log every step of the loop, not just the final answer -- see "Observability: Logging and Tracing an Agent's Decisions" -- a final answer without a trace behind it can't be debugged when it's wrong.

Common Mistakes

- **Relying only on the model's own judgment to stop the loop.** As "Step and Iteration Limits" explained, a decision step is still the same kind of model covered by "Reasoning Limits" in "LLM Capabilities and Limitations" -- a hard step limit is a backstop for exactly the cases where its own judgment doesn't conclude.
- **Requiring human approval for every single action.** As "Human-in-the-Loop: Approval Before Risky Actions" noted, this removes the autonomy that makes an agent useful in the first place -- approval should target the specific actions where a mistake is costly, not all of them uniformly.
- **Giving an agent broad tool access "just in case it's needed."** "Scoping Tool Access: The Principle of Least Privilege" covered why this expands the damage a bad decision can do without making the agent's decision-making any more reliable.
- **Treating a missing log as harmless because the final answer looked right.** "Observability: Logging and Tracing an Agent's Decisions" explained that a wrong answer without a trace is nearly impossible to diagnose after the fact -- the trace matters most exactly when it's needed, which can't be predicted in advance.

Summary, Cheat Sheet, and Glossary

Summary

- Guardrails exist because an agent's decisions and actions happen without a human checking each one -- the same autonomy that makes agents useful is what makes an unconstrained agent risky.
- A step limit is a hard, code-enforced backstop that stops the loop no matter what, independent of whether the model's own decision step ever concludes the goal is met.

- Human-in-the-loop routes specific costly or irreversible actions through an approval step, rather than requiring (or refusing) approval uniformly.
- The principle of least privilege bounds an agent's tool access to what its task genuinely needs, shrinking the space of possible damage from a bad decision.
- Observability -- logging every step, not just the final answer -- is what makes a wrong outcome debuggable after the fact.

Cheat Sheet

- Step limit = hard maximum on decide-act cycles, enforced in code, always stops the loop when hit.
- Human-in-the-loop = explicit approval step before specific costly/ irreversible actions, not every action.
- Least privilege = agent's tools scoped to exactly what its task requires, nothing broader "just in case."
- Observability = a per-step log/trace (decision, arguments, real result) covering the whole run, not only the final answer.

Glossary

- **Guardrail:** a mechanism that bounds what an autonomous agent loop is able to do, independent of the correctness of its own decisions.
- **Step limit (iteration limit):** a hard maximum on how many decide-act cycles an agent loop may run before it's forced to stop.
- **Human-in-the-loop:** routing specific agent actions through an explicit human approval step before they execute.
- **Principle of least privilege:** granting exactly the access (here, the tool list) a task needs, and no more.
- **Observability:** the ability to see, after the fact, exactly what steps an agent took and why -- typically through logging or tracing each step of the loop.