

A FIELD GUIDE

# Who Is the Agent?

*Identity, authority, and audit for non-human actors.*

Identity was built for people. Agents break that. This is a practical guide to giving software actors their own names, the smallest possible authority, the shortest possible time, and a record that survives them, so the systems you build with agents are ones you can actually leave running.

**Juan Cardena**

Enterprise AI & Agentic Systems Architect

CONTENTS


01 The Question We Skipped

02 Why Agents Break Identity

03 Give Every Agent Its Own Name

04 Short-Lived Beats Long-Lived

05 As Itself, or On Behalf Of

06 What They May Know: The Index That Knows Who's Asking

07 What They May Do: One Front Door

08 Audit and Provenance

09 A Field Checklist

CHAPTER ONE

# The Question We Skipped

For most of my career, the question at the edge of every system had a human answer. Who is this login, this request, this change? A person, or a service standing in for a team of people. Then agents arrived, software that decides and acts on its own, and the old answer quietly stopped working.

When we wire up our first agent, almost nobody asks who it is. It runs as us, because we started it, or under a shared service account with a password in a config file, because that was fastest. It acts, things happen, and for a while nothing goes wrong. That is the trap. The identity question never announces itself until an agent does something you did not expect, and then the first thing you want to know is which one, acting for whom, with whose permission. If the answer is a shared account a dozen things use, you have no answer at all.

> **The thesis of this guide:** *the intelligence of an agent gets the attention, but its identity is what makes it safe to run. Who is the agent is the central security question of agentic systems, and everything else is built on the answer.*

This is a short, practical guide to answering it. It is anti-hype and production-first. It will not tell you which vendor to buy. It will tell you the properties an agent's identity must have, whatever you build on, so that a year from now you can still open a log and read a plain sentence about who did what, and why they were allowed to.

CHAPTER TWO

# Why Agents Break Identity

Our identity systems were built around two kinds of actor, and an agent is neither. A human user is one person: present, accountable, and slow. A classic service is a fixed piece of software doing a fixed job. An agent has the speed of a machine and the discretion of a person, a genuinely new combination.

It is software, so it is fast and can be copied a hundred times over. But unlike a plain service it makes decisions, calls tools, and often acts for a specific human in a specific moment. Trying to model it as just another user means handing it a person's login and hoping. Trying to model it as just another service means giving it one broad, permanent identity and trusting it to only ever touch the right things. Both are where the trouble starts, because both pretend an agent is something we already knew how to secure. It is not. It needs its own category.

The good news is that the properties that category needs are not exotic. They are the oldest ideas in security, applied honestly to a new kind of actor: give it its own name, give it the least authority that works, give it that authority for the shortest time that works, and write down everything it does. The rest of this guide is those four ideas, made concrete.

CHAPTER THREE

# Give Every Agent Its Own Name

The first rule is that every agent gets its own identity, distinct from the human who started it and distinct from every other agent. Not a shared account. Not your personal credentials passed through. Its own.

This sounds like bookkeeping and it is the foundation of everything else. Once an agent has its own name, you can give it exactly the permissions it needs, you can see what it did, and you can turn it off without turning off everything else. Without its own name, none of that is possible, because every action dissolves into an anonymous pool that could have been anyone. You cannot grant, revoke, observe, or hold accountable an actor you never named.

> *A shared identity is convenient right up to the moment you need to answer a question about it, and then it is worthless. Name every agent before you need to.*

The test is simple. Picture the worst afternoon: an agent has done something strange, and you are trying to find out which one and on whose behalf. If your naming scheme lets you answer that in seconds, you built it right. If it forces you to guess, you have a shared-identity problem you have not felt yet.

CHAPTER FOUR

# Short-Lived Beats Long-Lived

A human password might live for months. That is already risky, and for an agent it is far worse, because an agent's credential sits in automated systems where a leak can be used instantly and at scale.

So the second rule is that an agent's authority should be short-lived by default. It gets a credential good for the length of its task, or minutes, not months, and then that credential expires on its own. A leaked token that is already dead is not much of a prize. Short-lived credentials turn a permanent risk into a brief one, and they force the system to keep proving, over and over, that this agent is still supposed to be doing what it is doing.

There is a cultural benefit too. When credentials are permanent, expiry feels like an emergency and teams route around it. When credentials are short by default, renewal is just how the system breathes, and nobody is tempted to paste a long-lived secret somewhere convenient. The safest secret is the one that would be useless by the time anyone found it.

CHAPTER FIVE

# As Itself, or On Behalf Of

There are two ways an agent acts, and keeping them straight is where most real incidents are prevented. Sometimes it acts as itself, doing background work that belongs to no particular person. Sometimes it acts on behalf of a specific human, and there it should borrow only that person's authority, only for that task.

The dangerous shortcut is to give an agent broad standing power so it can handle anything a user might ask. That convenience is the risk. An agent that always holds a lot of authority is a loaded tool sitting in the open, and anyone who can get it to act, including through a cleverly worded request, can borrow that authority. There is an old name for the failure: the confused deputy, a helpful actor with more power than the person asking, tricked into using it on their behalf.

> **Delegation, not accumulation.** *When an agent acts for a person it should reach exactly what that person could reach, scoped down to the task, for a short and revocable window, and then hand the authority back.*

Scope down, never up. If the job is to read one thing, the borrowed power should permit reading that one thing, not everything the user could read, and certainly not writing. Return it promptly, and make it revocable everywhere at once. Do this and the worst a manipulated agent can be talked into is whatever the person it serves was already allowed to do, which is a bounded, survivable failure instead of a catastrophic one. You will not prevent every trick. You make sure no trick grants more than the person deserved.

CHAPTER SIX

# What They May Know: The Index That Knows Who's Asking

Identity does not stop at what an agent may do. It also governs what it may know. Put a model on top of all your company's knowledge and you have built a useful assistant and a dangerous one at once, and the difference is a single property of the index underneath: does it know who is asking.

An enterprise retrieval index must return only what the asker is allowed to see. That is not a filter you add at the end. Permissions have to travel with the content into the index, every query has to carry the asker's identity, and access has to be enforced at query time, because who may see what changes constantly as people join teams and leave them. Freeze the permissions at index time and you have a snapshot that goes stale the first time someone changes roles.

Notice where the leak actually happens. People worry about the model, but the model can only speak about what retrieval handed it. If the index returns a restricted document, the model will faithfully summarize it. The boundary is at retrieval, and it has to be built together with retrieval quality, not against it. An index that leaks and an index that cannot find things are both failures. The one you can safely put in front of the whole company is the one that behaves, for each person, as if it only ever contained what they were allowed to have.

CHAPTER SEVEN

# What They May Do: One Front Door

The first agent that calls a tool on its own is a genuine milestone. The hundredth is a management problem. Somewhere between, a system turns into a mesh: every agent holding its own credentials, reaching its own tools its own way, each connection a small door nobody is watching.

The fix is boring and powerful. Put one gateway between the agents and the tools, a single front door every tool call passes through. Now there is one place that sees every tool use in the system, and that place is worth an enormous amount. It is where you enforce one consistent set of rules, where identity and delegation are checked, where cost is metered, and where you can shut off a capability everywhere with one change. The connector behind it can be a command line, an API, a skill, or a protocol; that is an implementation detail. The gateway is where governance lives.

> *Policy scattered across a hundred agents is a hundred slightly different rule sets, and the gaps between them are where incidents live. One gate, one rule set, applied the same way to everyone.*

This is the same instinct that keeps agents trustworthy elsewhere: let the model propose, let deterministic code enforce. The agent can ask the gateway for anything. The gateway is the code that decides, using the agent's own name, its delegated authority, and the rules you wrote down once.

CHAPTER EIGHT

# Audit and Provenance

Everything in this guide converges on one payoff. When something goes wrong, and eventually something will, you want to open the log and read a plain sentence: this agent, acting for this person, called this tool, on this target, at this time, and here is the identity that authorized it.

That sentence is only possible if every agent had its own name from the start, if its authority was scoped and short-lived, and if every action passed through a place that recorded it. Audit is not paperwork you add at the end. It is the reward for getting identity right, and it cannot be faked in after the fact. Provenance, the ability to trace an action back through the agent, the delegation, and the human intent behind it, is what lets you hold a non-human actor accountable at all.

I learned this the way most lessons arrive, by doing it wrong. An early system of mine ran several agents under one shared identity, because it was simpler, and for weeks it was. Then one behaved oddly, and I sat down to find out which and could not. The log showed the shared account acting, but not which agent, not for whom, not under what authority. I had built something I could not reason about the moment it mattered. The fix was not clever: give each agent its own short-lived identity and log every action under it. Nothing about the agents changed. Everything about my ability to trust them did.

CHAPTER NINE

# A Field Checklist

If you take one thing from this guide, take the questions to ask of any agent you are about to run. They are short, and they are the difference between a system you can leave running and one you cannot.

- **Its own name.** Does this agent have an identity distinct from the human who started it and from every other agent? Never a shared account.
- **Least authority.** Does it hold the smallest set of permissions that lets it do its job, and nothing it is not using right now?
- **Short-lived.** Does its authority expire on its own, in minutes or task-length, not months?
- **As itself or on behalf of.** Is it clear which, and when it acts for a person does it borrow only that person's authority, scoped down and revocable?
- **No standing power.** Does it avoid holding broad permanent authority between tasks that a manipulated agent could be made to use?
- **Knows only what it may.** When it retrieves, does the index return only what the asker is allowed to see, enforced at query time?
- **One front door.** Do its tool calls pass through a single gateway where identity, policy, cost, and audit are enforced?
- **Attributable.** Can you open a log and read who did what, for whom, and on whose authority, in one plain sentence?

> *The systems I am willing to leave running are not the ones with the smartest agents. They are the ones where I can always say exactly who did what, and why they were allowed to.*

Give every agent its own name, the least authority for the shortest time, borrowed from the person it serves and returned, over knowledge it is allowed to hold, through one door you control, with a record that outlives it. Do that, and who is the agent always has an answer. That answer is what turns a clever demo into a system you can trust.

**Juan Cardena**

Enterprise AI & Agentic Systems Architect. Designs and operates production agentic pipelines, model routing, and guardrails, and ships to a Kubernetes cluster.

Read the companion posts at blog.jcardena.com. This guide is anti-hype and production-first, and it names no vendors on purpose: the properties outlined here hold whatever you build on.