
The ontology is the software.

How a large model reads an enterprise

When execution is nearly free, an enterprise is its definition.

Enterprise software changed its interface three times. Its user never changed.

1980s

Terminals and forms

People adapted to the software: commands, fields, codes.

2000s

Web and mobile

The software adapted to people: a tap, a swipe, anywhere.

2020s

A conversation box

The software disappears into language: say it, and it is done.

THE ONE THING THAT NEVER CHANGED

The user was a person. The API was written for a programmer, the documentation for a reader, the audit log for an auditor.

For the first time in forty years, software has a new user. It is not a person. It is an agent.

A company, a few years from now, nine in the morning

- 09:00** Three hundred orders arrive. Stock checked, quoted, scheduled: agents finish in ten minutes, and nobody was called.
- 09:12** One discount exceeds its authority. The agent stops and waits for a person's yes. One line lands in the audit log.
- 09:30** A decision to open a branch in Chengdu. Copy the enterprise's definition, change three things, take orders by the afternoon.
- All day** Agents running, a few dozen people setting rules and approving exceptions. Nobody is "operating software".

THE QUESTION FOR TODAY

On that day, take the people out and switch the systems off. What is left of the enterprise?

To answer it, we have to go back to an older question: what is an enterprise?

What is an enterprise?

COASE, 1937

A firm exists because coordinating inside it is cheaper than transacting in the market.

DRUCKER, 1954

There is only one purpose of a business: to create a customer.

BOTH ARE RIGHT. NEITHER SAYS WHAT A FIRM IS MADE OF.

An enterprise = a set of definitions + the people and systems that execute them

what we have

how it relates

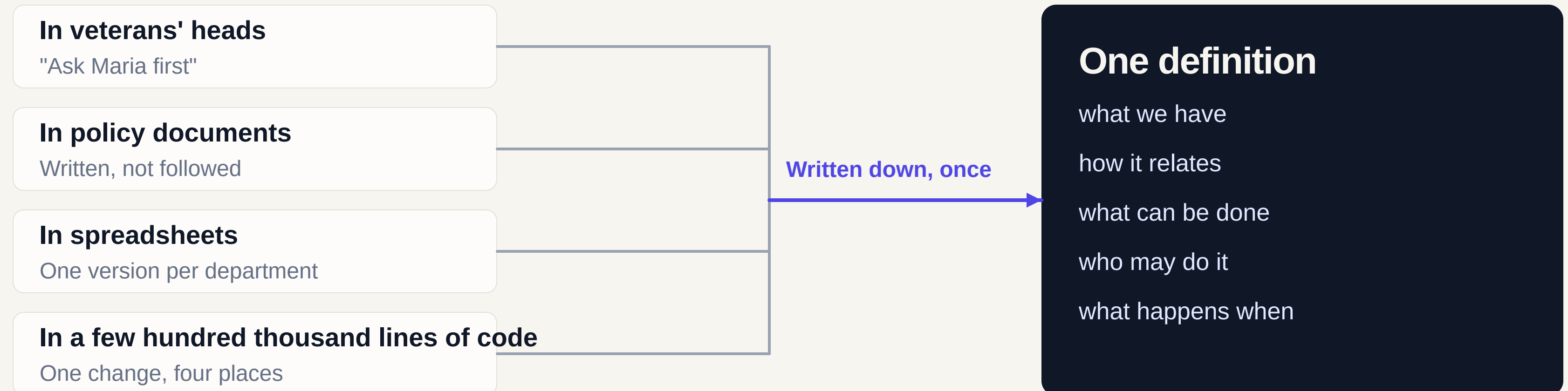
what can be done

who may do it

what happens when

The definition is the asset. Execution is the cost.

Every enterprise has this definition. It was never written down.



For forty years the definition was remembered and the execution was done by people.

The model knows everything. Except your company.

IT HAS READ THE WHOLE INTERNET

It writes poems and code, passes bar exams, reads financial statements. It scores higher than most of us.

IT HAS NEVER READ YOUR ENTERPRISE

Ask which customer complained most last month: no answer. Tell it to assign a ticket to Wang: it cannot.

95%

of enterprise generative-AI pilots produced no measurable P&L impact.

The bottleneck is a learning gap, not model quality. MIT NANDA, The GenAI Divide, 2025

People call this "the last mile". The root cause: the model does not know this enterprise.

It is a brilliant new hire on day one

01

No world

It does not know this company: where the customers are, how a process runs, who may see what. The more specific the business, the less it has seen.

02

No hands

It can talk, not act. It cannot create a table, open an endpoint, or change a record.

03

No fence

It does not know what is forbidden. And when it acts, nobody writes it down.

What the model lacks is not intelligence. It lacks a world, hands, and a fence.

The industry has named the problem: in June 2026 Forrester published a report titled [Combine Semantics, Ontology, And Knowledge Graphs For AI-Ready Data](#).

TWO · THE LARGE MODEL ARRIVES

When execution is nearly free, an enterprise is its definition.

AI is driving the cost of execution toward zero. Where the definition lives, in what form, and whether it can run: that is now the only question in enterprise software.

Ontology: the study of what there is

IN PHILOSOPHY

What exists

Ontology begins with Aristotle: what exists, how it is classified, how it relates.



IN COMPUTER SCIENCE

Writing "what there is" down

In 1993 Gruber defined an ontology as "an explicit specification of a conceptualization": what a domain contains and how it relates, in a form a machine can read.



IN AN ENTERPRISE

The sentence a business says about itself

The definition from part one, written in a form a machine can read and validate.

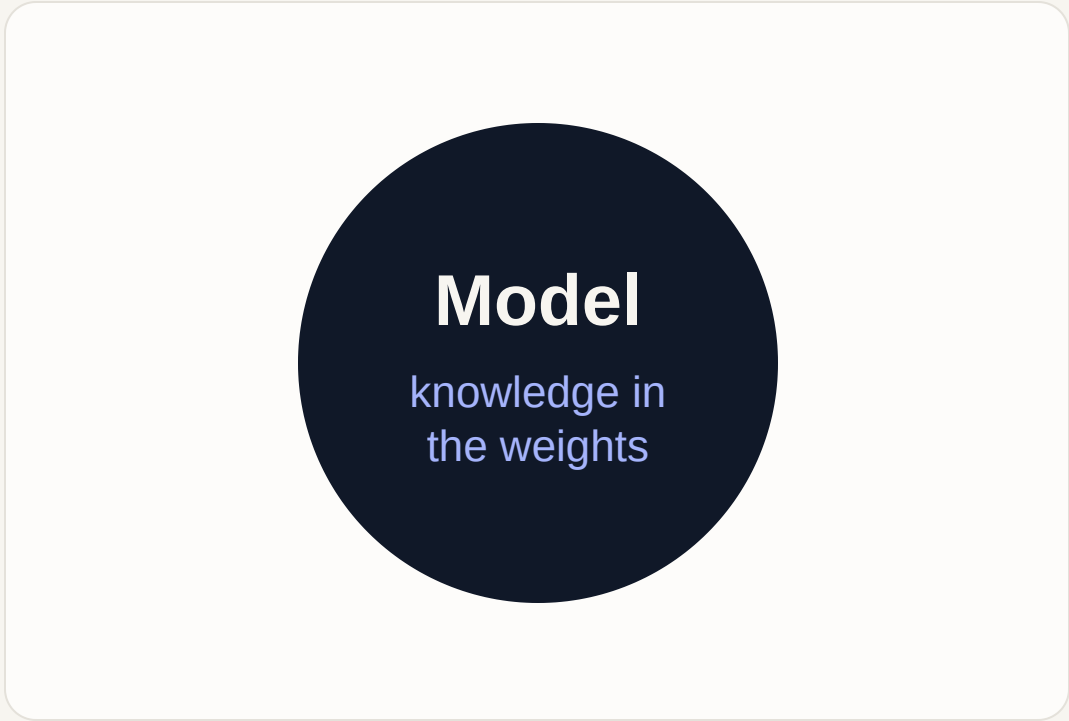
An enterprise's ontology is the definition from part one.

A Chinese industry report (AIIA and CCSA TC601, July 2026) names the three pain points of LLM deployment: missing business knowledge, uncontrolled reasoning, decisions that never land.

THREE · WHAT AN ONTOLOGY IS

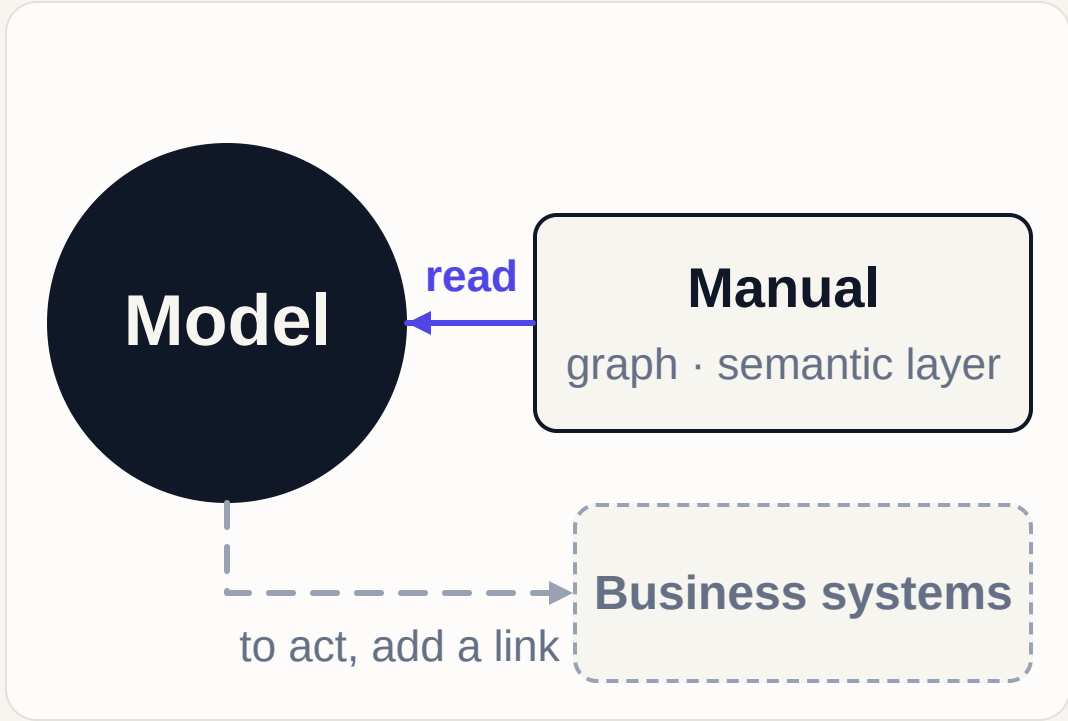
Once it is written down, where does it live?

ROUTE ONE · TRAIN IT IN



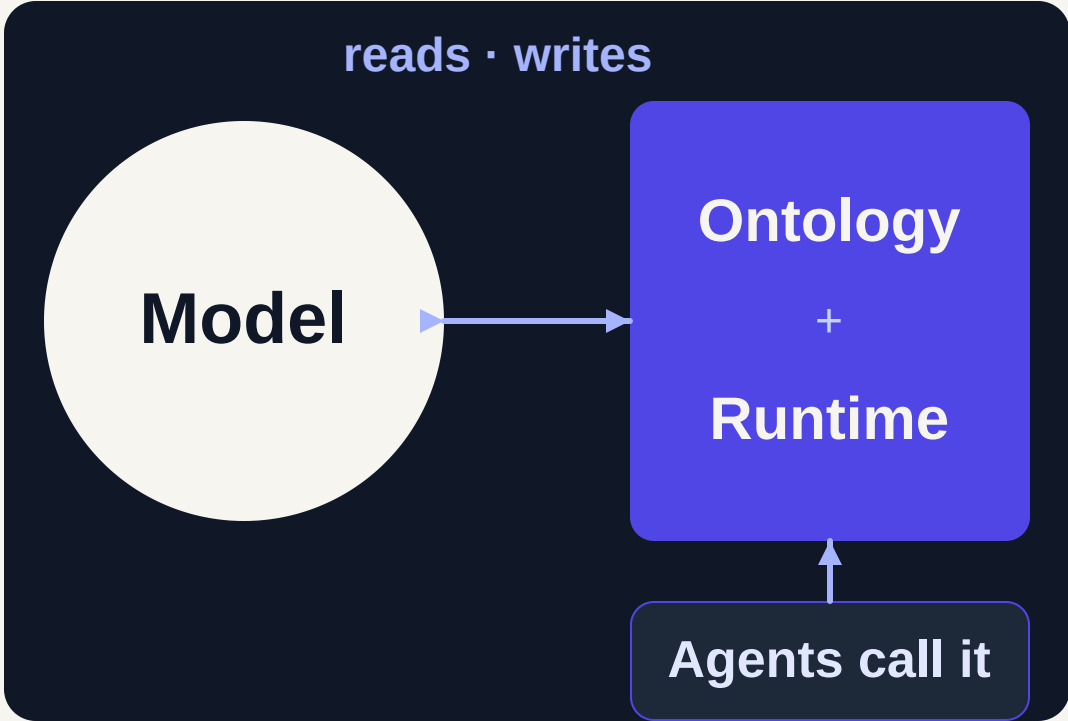
Change one rule, retrain. Nothing to inspect, nothing to audit. Change the model, lose it all.

ROUTE TWO · OUTSIDE, AS A MANUAL



The model can read it, but reads a description. Acting needs another system; permissions are yet another.

THIRD ROUTE · OUTSIDE, AND IT RUNS



Written precisely enough, the ontology is not a manual. The model reads and writes it, the system runs it, agents act through it.

Knowledge outside the model: models change, the ontology stays.

Who is building ontologies in 2026

READ-ONLY

IT RUNS

Knowledge ontologies

Describe a domain and reason over it. Thirty years of knowledge engineering.

Protégé, OWL, the biomedical and financial standard ontologies

Semantic layers

Make existing data legible to models. Read-only; the system lives elsewhere.

Microsoft Fabric IQ, Databricks, Snowflake, AWS; Yonyou LOM in China; open source: semantica, utopia

Operational ontologies

Add actions and permissions over existing systems. Can act; still a layer on someone else's system.

Palantir; Kingdee Lingee in China; open source: OpenBKN

Executable ontology

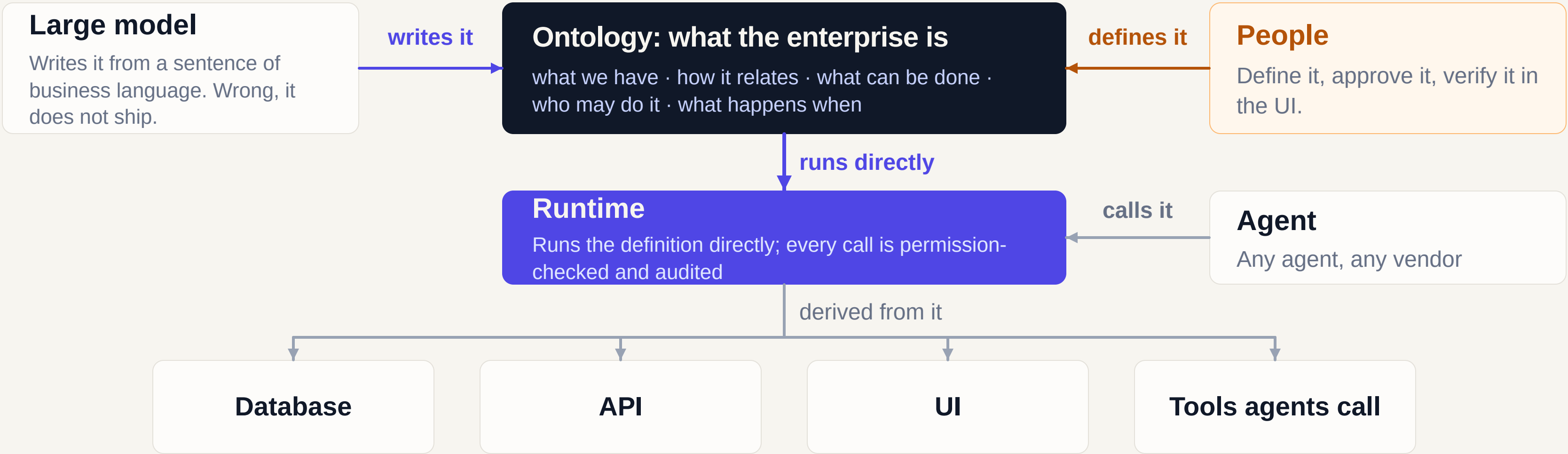
The ontology is the system: tables, APIs, UI and agent tools all derive from it.

ObjectStack, open source, Apache-2.0

In 2026 almost everyone is in the second cell. The fourth, for now, is us.

The fourth cell is small because it covers new systems only. Existing systems live in cells two and three, and that is their home ground.

An ontology written clearly enough runs directly



The ontology is not the design document for the software. The ontology is the software.

Open-source implementation: ObjectStack, Apache-2.0. A ticket object is a dozen lines; the whole system exists the moment it does.

What the runtime does with the definition

You define	The runtime derives	The runtime enforces on every call
Objects and fields	Tables and columns, REST and realtime APIs, read and write tools agents can call	Types and required fields, field-level security
Relations	Foreign keys and joins, endpoints for related objects	Referential integrity, cross-object permissions
Actions	Callable action endpoints; only those declared AI-exposed become agent tools	Preconditions, who may call, audit on every call
Permissions	Roles, permission sets, sharing rules	Role, row-level and field-level checks; one set for people and agents
Flows	Record triggers, scheduled jobs, approval chains, state machines	Executed as defined, every step recorded
Projections	Lists, boards, forms, dashboards, app navigation, translations	Change the core and projections follow; delete a projection and the business is unchanged

Left: the ontology. Middle: the software. Right: the fence. Nobody writes the middle column anymore.

All three, now

01 No world



WORLD = ONTOLOGY

The model reads it and knows the enterprise

An enterprise's whole definition is small enough to fit in one context window. It reads the enterprise itself, not documents about it.

02 No hands



HANDS = RUNTIME

Objects are tools, actions are tools

Every object and action in the ontology is callable by an agent.

03 No fence



FENCE = PERMISSIONS AND AUDIT

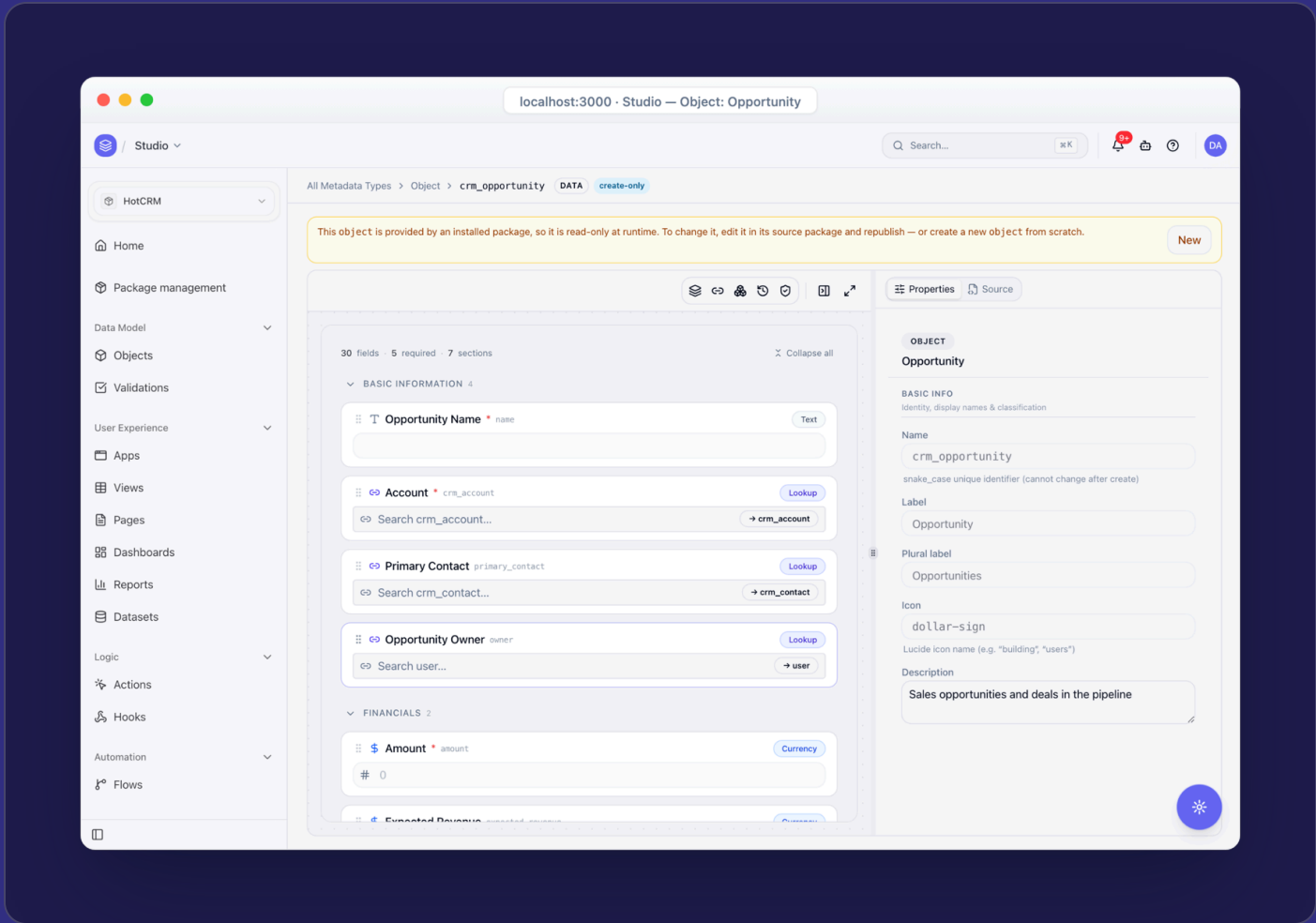
An agent can do exactly what the ontology says

Who may do what is written in the ontology; every call is checked and recorded.

Software because it runs. Safe for agents because someone guards the door while it runs.

CAICT (China Academy of ICT), September 2026: agents move AI from "can talk" to "can act"; risk moves from biased output to uncontrolled action.

An agent reads an enterprise in three steps



Studio's object designer: what an ontology looks like in the product

01 Connect
Point the agent at the enterprise. It lists the objects and actions it sees: the ontology, not one word more.

02 Act
"Create an urgent ticket, customer cannot log in, assign it to Wang."
The ticket appears; the audit log says an agent created it.

03 Refused
Repeat the call as a read-only identity. Refused. An agent's permissions are a person's, not one bit more.

People verify in the UI. They do not operate it.

What an enterprise truly owns in the AI era

Not the model: it is rented, and changes every year. **Not the chat box:** they will all look alike. **Not the agents:** there will be more of them.

The more specific your business, the less any model has seen of it, the more this definition is worth, and the less it belongs in anyone else's hands.

Owned by the enterprise, not by a vendor.

- The ontology lives in your own repository, not with a vendor
- Connect one model today and another tomorrow; the ontology does not change
- Open protocol: change the runtime or the vendor and take it with you
- An ontology owned by a vendor is a deeper lock-in than SaaS

The one thing an enterprise truly owns, and only needs to own, is its definition of itself.

"You are paying for the right for them to migrate your IP, your know-how, your expertise to their model." Alex Karp, Palantir Q2 2026 earnings call, August 2026

The future enterprise: one person, ten thousand agents

	Today	A few years from now
Organization	A group of people, divided by role	A few definers and decision-makers, plus any number of agents
Org chart	An HR document	A permission graph: who may do what, written in the ontology the system runs
Core asset	People and a few hundred thousand lines of code	One ontology: a merger merges ontologies, a franchise copies one
Boundary	Set by internal coordination cost	Agents transact across firms; boundaries soften, firms shrink, interfaces multiply
People's work	Operating software	Defining the enterprise; reversible work to agents, irreversible decisions to people

One person defines the enterprise. Ten thousand agents run it.

One and ten thousand are limits, not today's reality. Today's real shape is a few definers plus a handful of agents. The limit shows the direction.

What you get when the enterprise is written as an ontology



WITHIN MONTHS

Felt immediately

- The scattered definition written once; one change, one place
- New apps in days: describe the business, AI writes the ontology, the gate catches the mistakes
- Agents connected by default, under the same permissions and audit as people



IN A YEAR OR TWO

Structural

- The ontology becomes your asset, independent of any model or vendor
- The enterprise becomes legible to AI once; the return compounds
- What is written wrong does not ship; what is done wrong stays inside the fence



LONG TERM

Strategic

- Operations scale without headcount: more business, not more people
- Experiments get cheap: define it and it runs, idea to running in days
- Replication gets cheap: entities, branches, franchises and acquisitions are copies or merges of one ontology

The first investment is not a platform. It is sitting down to answer five questions.

Say the boundary plainly: this covers business that can be defined and built or rebuilt; core legacy systems are not in it. The ontology is only as good as the people who define it.

THREE SENTENCES TO TAKE HOME

An enterprise is definition plus execution. When execution is nearly free, it is its definition.

What the model lacks is not intelligence: a world, hands, and a fence.

An ontology written clearly enough is software. Models change; the ontology stays.

Write the enterprise as an ontology that runs. Give execution to agents. Keep definition for people.

ObjectStack Team

`npm create objectstack@latest`

`github.com/objectstack-ai/objectstack` · `objectstack.ai/blog/the-ontology-is-the-software`