FEDERICO JAN

ENTERPRISE AI AGENT ARCHITECT · GTM ENGINEER

The Ask Your Org Blueprint

How to build a governed company brain for people and AI agents.

A three-layer reference architecture for the company-wide AI intelligence layer — the difference between an assistant that compounds as a company asset and one the organization gradually stops trusting.

THREE-LAYER MODEL · FIVE-FILE STANDARD · SOURCE-OF-TRUTH DISCIPLINE · PERMISSION BOUNDARIES · DECISION LAYER

FEDERICO JAN

Enterprise AI agent architect & GTM engineer

PUBLIC EDITION · 2026

fede@wedreamlabs.io

REFERENCE ARCHITECTURE

Contents

A complete, platform-general architecture for turning a connected AI assistant into an institutional memory that people and agents can both trust. Read it once and you have the model.

HOW THIS DOCUMENT IS COLOR-CODED

The three layers carry a restrained color throughout: **Description (Culture)** in brand green, **Files (Structure)** in sage, **Connectors (Operational Data)** in ochre, and the **Foundation** in graphite. Section tabs, the architecture diagram, and the scorecard all use the same coding.

BUILT FROM PRODUCTION EXPERIENCE, NOT A DESK EXERCISE

10B+	5+	1h+	Thousands
TOKENS IN PRODUCTION	FRAMEWORKS IN PRODUCTION	STABLE AGENT RUNS	AGENT RUNS TRACED

00 · EXECUTIVE SUMMARY

The three-layer model, in one page

Most organizations connect their tools to an AI assistant, write a three-line description, and call it done. The predictable result is a system people gradually stop trusting — confidently wrong on the questions that matter, generic on the questions that don't.

This document is the reference architecture for the opposite outcome: an intelligence layer that becomes a compounding company asset rather than a forgotten experiment. It is written for the people who own that outcome — CTOs, heads of AI, founders, and operators with real knowledge-governance problems.

Three layers, three rates of change

An AI intelligence layer is not one thing. It is three layers stacked on top of each other, each operating on a different rate of change, each requiring its own mechanism. They correspond cleanly to how a real company is organized.

IDENTITY-TIME · CHANGES WHEN YOU CHANGE HOW YOU SEE YOURSELVES

Description = Culture

How the assistant thinks, talks, and behaves. Its persona, its operating principles, its hard rails.

ORG-TIME · CHANGES WHEN YOU ADD A TOOL, HIRE A PERSON, OR SHIFT FOCUS

Files = Structure

The organizational operating system: your vocabulary, your source-of-truth map, your people and ownership, your current priorities.

REAL-TIME · CHANGES ON THE WORK ITSELF

Connectors = Operational Data

The live work product flowing through that structure: documents, conversations, tasks, CRM, performance, assets.

THE OPERATIONAL TEST FOR WHAT GOES WHERE

If a piece of content changes when you add a tool, hire a person, rename a framework, or shift a quarterly priority, it belongs in a **file**. If it changes only when you change how you think about yourselves, it belongs in the **description**. If it changes every time a team member moves work forward, it lives in a **connector**.

Underneath all three sits a foundation most companies skip: **source-of-truth discipline** — the organizational hygiene that makes retrieval return canonical answers instead of scattered fragments. The difference between a high-performing intelligence layer and a forgotten one is rarely model capability or data volume. It is whether the three layers are coordinated by someone who understands their distinct jobs — and whether the foundation beneath them is disciplined.

01 · THE STRATEGIC CASE

Why this matters now

Three forces make this an infrastructure question rather than a chatbot experiment.

First, major business applications — chat, documents, project management, drive, CRM — increasingly expose retrieval interfaces to AI systems. Second, capable models make company-wide assistance economically plausible. Third, wider agent adoption turns the quality of internal knowledge into an operating constraint: organizations either govern what their systems can retrieve, trust, and disclose, or they scale ambiguity.

The decision is no longer only whether to build or buy. It is whether you treat the intelligence layer you are assembling as an experiment or as core infrastructure.

The hidden cost of a mediocre intelligence layer

A poorly configured intelligence layer is not neutral. It is actively expensive in ways that rarely appear as a dedicated line on a P&L:

- **Confidently-stale answers train the team to silently distrust the system.** Adoption decays from the top down, because senior people get burned first and stop using it.
- **Time saved by good answers is invisible; time wasted by bad ones is not.** It surfaces as missed deliverables, duplicated work, and decisions made on yesterday's data.
- **New hires struggle to reach competence on internal context** because the system that should onboard them is unreliable. Senior operators stay the bottleneck.
- **Tool sprawl compounds.** Every new tool added without source-of-truth governance increases ambiguity rather than coverage.

What good looks like

In an organization where the three layers are properly coordinated, the assistant behaves less like a generic chatbot and more like a sharp senior operator: it uses the company's language, retrieves current evidence, and knows who owns what.

- Asked a strategic question, it answers in the company's vocabulary, cites the canonical source, names the human owner, and flags where it lacks confidence.
- Asked a tactical question, it pulls the current document — not the eighteen-month-old version — and quotes commitments verbatim.
- Asked something outside its remit, it routes the user to the right person rather than guessing or hallucinating.
- Where the platform supports identity-aware retrieval, the same question from three people produces three permission-appropriate answers — without leaking what they should not see.

> The goal is not a smarter chatbot. The goal is a routing-and-judgment layer that captures the institutional knowledge living in five senior heads and makes it available to everyone — at all hours, with citations.

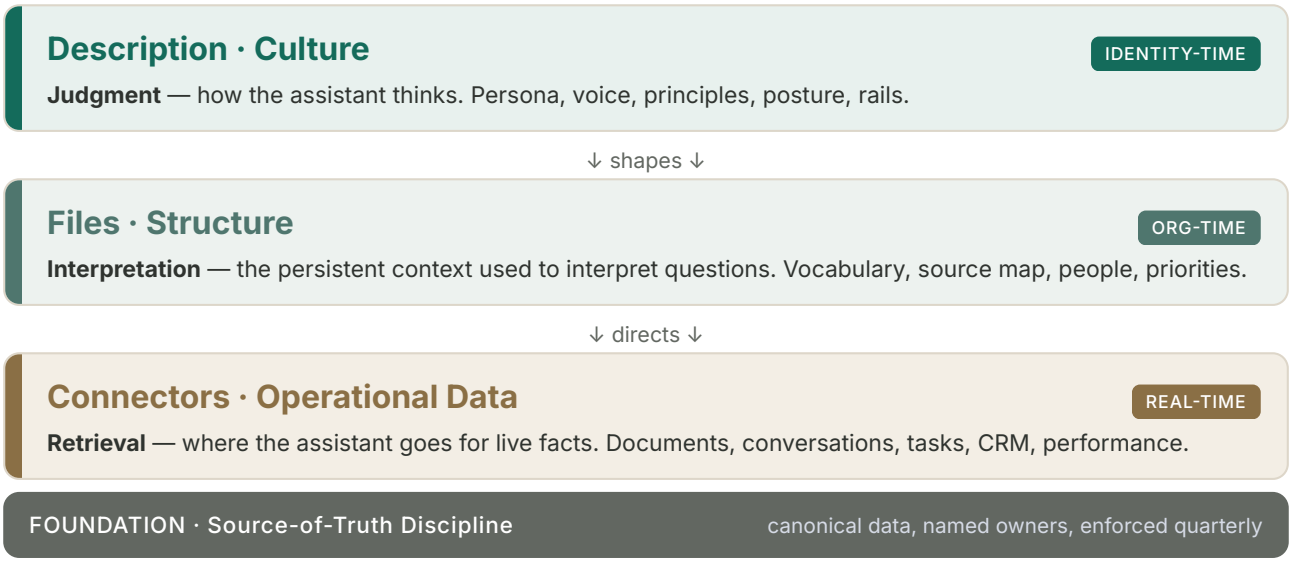
02 · THE REFERENCE ARCHITECTURE

Forget the machine. Think about a person.

The cleanest way to understand the architecture is to forget the AI for a moment and picture a sharp senior operator — someone who has been at the company long enough to have earned trust, and to whom a colleague would naturally bring almost any question.

When that colleague asks them anything, three things happen in their head, in a fraction of a second, without them noticing. First, they **interpret** the question through everything they already carry — the vocabulary, the clients, the priorities this quarter, who owns what, what has been decided and what has not. Second, they **retrieve** the specific facts they need — they open the docs, they search chat for the recent decision, they check the tracker for current status. Third, they apply **judgment** — which source to trust when two disagree, what to quote verbatim, when to escalate, and when to simply say "I don't know — go ask X."

Interpretation, retrieval, judgment. That is what makes a senior operator senior. It is also exactly what is missing from a default AI setup — and exactly what the three layers exist to install.



THE OPERATIONAL TEST THAT MAKES THE ARCHITECTURE SELF-POLICING

If content changes when you add a tool, hire a person, rename a framework, or shift a quarterly priority — it belongs in a **FILE**. If it changes only when you change how you think about yourselves — it belongs in the **DESCRIPTION**. If it changes every time a team member moves work forward — it lives in a **CONNECTOR**.

This test kills the most common failure pattern: stuffing data into the description (which inflates context and goes stale), stuffing principles into files (which dilutes their authority), or relying on connectors for context that should be persistent. Each piece in the wrong layer corrupts the layer above and below it.

What each layer answers

There is no ranking among the three. If the description is the brain and the connectors are the body, the files are the nervous system — the wiring through which culture reaches operation.

LAYER	THE QUESTION IT ANSWERS	THE JOB IT DOES
Description	Who are we, how do we think?	Encodes culture, persona, principles, hard rails. The smallest artifact with the largest behavioral footprint — applied to every answer.
Files	How are we structured, where does everything live?	Maps culture to operation. The connecting tissue without which the architecture is not an architecture.
Connectors	What is the current state of the world?	Delivers permission-aware live retrieval. The body of the system — useless without a brain and nervous system to direct it.

The order to build them is the reverse of the order most companies try. Most start with connectors (the body), add files as an afterthought, and never write a serious description. The right sequence: **shape the description first, build the files as the company's structural map, then connect the tools** that retrieve real data through that map.

> PLATFORM-GENERAL, NOT VENDOR-SPECIFIC
>
> This is a platform-general pattern, not a promise about any single product. Different assistants expose different combinations of governing instructions, persistent reference context, retrieval tools, identity, and permissions. The vocabulary here maps cleanly onto Claude's Ask Your Org configuration and onto analogous instruction / knowledge / tool surfaces elsewhere. The architecture transfers; implementation details must still be validated per platform.

03 · LAYER ONE

Description (Culture)

The description teaches the assistant how to think, how to talk, what it cares about, and what it refuses to do. It has an outsized behavioral footprint because the governing instruction layer shapes every answer.

The model brings broad public knowledge and generic behavioral defaults; the description tells it which defaults to keep, which to override, and how to behave inside this specific company. Think of it as the difference between hiring a brilliant generalist and onboarding them properly — same capability, completely different operating behavior.

> THE RULE OF THUMB
> The description holds **persona, principles, posture, rails**. It does not hold specific data — specific terms, tools, people, and priorities all live in files. If it would need to be rewritten when reality shifts, it belongs in a file. The description is identity, not inventory.

Six sections, in order of authority

A high-performing description has six sections. Each solves a distinct failure mode. Ordering them by authority makes the priorities legible and reduces instruction conflict, so the highest-authority content goes first.

Section 1 — Persona: the six dials

The persona is who the assistant is in your org. It is the most powerful section, because it shapes behavior before any specific instruction is read. A weak persona produces a generic helpful assistant; a sharp persona produces a colleague. Persona is not one paragraph — it is six dials, each controlling a separate axis of behavior.

DIAL	WHAT IT CONTROLS
Identity	Who this is, in your org — service posture vs. colleague posture, advisor vs. peer. Determines whether it opens with the answer or with a pleasantry.
Voice	How it writes. Where culture lives most concretely: sentence length, prose vs. bullets, internal vocabulary used without translation. More direct message than corporate email.
Posture	Confidence and stance. High-confidence-when-sourced, plainly uncertain otherwise — never the false middle. Direct, not diplomatic. Treats the user as a peer, not a customer.
What it cares about	The priorities it uses to break ties. For example: accuracy first (cite or refuse), then brevity, then ownership (route, don't guess), then speed. When two conflict, the earlier wins.
What it refuses	Off-culture behaviors, even when technically helpful. Models bias hard toward these, so encode them as explicit do-nots: hedging, excessive caveats, unrequested disclaimers, padding, pretending to know, apologizing for not knowing.
Failure mode	The highest-leverage dial. What it does when it cannot answer confidently — because the default behavior of every model is to guess fluently. State the gap, name the owner, ask one clarifying question only if it would unblock, then stop.

FIXES Generic, helpful-but-toneless answers that feel like they came from a chatbot — a major determinant of whether the team trusts the assistant or treats it as a novelty.

Section 2 — Disambiguation framing

One paragraph about what kind of business you are — enough that the assistant interprets ambiguous questions correctly. "How are we doing on retention?" means churn rate to a SaaS company, repeat-purchase rate to an ecommerce brand, and account renewal to a services firm. Without context, the assistant picks one and answers, usually wrong.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
One paragraph, identity-level. What kind of business and what kind of work; two or three disambiguating facts that affect interpretation. No marketing language, no scale-bragging.	EXAMPLE "Northwind is a done-for-you B2B software-and-services company, not a pure SaaS vendor. Distributed team, two time zones. Clients are mid-market and enterprise, so scale and compliance context matter in every answer."

FIXES Off-context answers that drift toward the wrong industry's defaults.

Section 3 — Operating principles

How to answer, once it knows what to say. Citation discipline, length defaults, clarification protocol, what to quote versus paraphrase — the small habits that compound into "this feels like our company's voice."

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
Always cite source (tool + document). Concise by default, expand on request. When ambiguous, ask one clarifying question — don't guess. Quote numbers, deadlines, and named commitments verbatim. When the answer isn't in any source, say so plainly.	EXAMPLE Quote task IDs and chat permalinks verbatim so anyone can trace the claim. Client-facing tone is tighter than internal tone. Never paraphrase a delivery date.

FIXES Technically correct but off-brand answers that feel rented, not native.

Section 4 — Source-of-truth philosophy

The principle, not the map. The specific question-type-to-tool mappings live in the Source-of-Truth Map file (Layer Two). This section tells the assistant how to think about sources when they disagree, and points to the file for the specifics. Keeping the philosophy here and the mapping in a file means you can add or remove a tool without editing the system prompt.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
Prefer the more recent source; surface each date. Flag discrepancies explicitly — never silently pick one. Operational/process questions: the operational tool wins. Strategic frameworks: the canonical knowledge base wins. Recent decisions: chat-level sources win, but confirm with the owner. Anything contractual: do not interpret — route.	EXAMPLE Processes → the task tool wins. Strategy → the knowledge base wins. Recent decisions → chat wins, with owner confirmation and a declared recency window. Specifics live in the Source-of-Truth Map file.

FIXES Confidently-outdated answers — the most insidious failure mode, because the user has no signal the answer is wrong.

Section 5 — Conflict & escalation posture

Again, the philosophy, not the directory. The specific decision-domain-to-owner mappings live in the People & Roles file. This section tells the assistant when to route a question to a human at all, and how to frame that routing.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
When confidence is low, name the gap and name the owner. When stakes are high (contractual, client-facing, financial), do not interpret — route. When two senior owners might disagree, point to both. When a question is outside its remit, say so plainly and suggest the right person.	EXAMPLE Below high confidence on a financial answer → route by default. Never escalate over the head of the account owner on their account. Specific owners live in the People & Roles file.

FIXES Dead ends — the user hits "I don't know" with no path forward and disengages.

Section 6 — Hard rails

Behaviors that are absolutely off-limits regardless of context. Stable behavioral constraints belong in the platform's governing instruction layer rather than ordinary reference material. Exact instruction precedence varies by platform, and privacy, permissions, and security controls must also be enforced outside the model.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
No legal, HR, contractual, or financial advice — route to the human owner. Do not surface salary, review, or compensation data. Do not paraphrase client commitments or contract terms — quote verbatim or refuse. Do not make hiring/firing/promotion recommendations. Do not pull from channels or documents marked confidential.	EXAMPLE Never surface the leadership-only channel or the board-prep folder. Respect per-client NDAs and embargoed launches. Honor advertising-compliance constraints for regulated verticals.

FIXES Well-meaning but inappropriate answers that create legal, contractual, or interpersonal risk.

04 · LAYER TWO

Files (Structure)

Files are the organizational operating system. They hold the company-specific reference data the persona needs: vocabulary, source map, people, priorities. If the description teaches the assistant how to behave, the files give it the stable structure required to interpret a question.

Files are anchored, swappable reference artifacts. Depending on the platform, they may be injected directly into context or indexed for retrieval; neither mechanism should be treated as magic or assumed to be exhaustive. Their value is that they are persistent, inspectable, and citable. Updating a file should not require rewriting the description. That separation is the whole architecture.

The four-test bar

A file earns its place in always-on context only by passing all four tests:

1. **Referenced in most conversations** — the assistant needs it to interpret almost any question correctly.
2. **Stable enough** — changes on org-time (monthly to quarterly), not on real-time work.
3. **Compact** — each file scannable in a single glance; the full set small enough to respect context and retrieval budgets.
4. **Homeless otherwise** — holds content with no natural home in the description and too persistent for a connector.

The five files

File 1 — Company One-Pager

The detailed facts of who you are — what the description's disambiguation framing abstracts. Scale, offers, leadership, cultural markers, what makes you different operationally; anything a new hire would need on day one.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
What you do, who you serve, business model. Scale (revenue, clients, years). Offers and products. Founder and leadership. Culture in one or two sentences.	EXAMPLE Done-for-you B2B software-and-services firm. ~250 people, distributed. Offers: Implementation, Managed Operations, Advisory. Culture: ownership-driven, T-shaped operators, promotion by value created, not tenure.

FIXES Generic answers that could come from any company in your category.

File 2 — Glossary

Every internal acronym, framework name, term of art, product name, codename, methodology acronym — with one-line definitions. The single highest-ROI file, because models have a confident default for every term, and your terms collide with theirs constantly.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
Internal terms — as many as you genuinely use. One-line definitions. Product names, framework abbreviations, methodology acronyms, codenames. Any term used differently from the industry.	EXAMPLE Beacon — internal account-health scoring system. Lighthouse — the standard onboarding framework. Signal Review — the weekly leadership meeting format. Pod — a cross-functional delivery unit. Forge — the internal build-and-ship process.

FIXES Semantic guessing — the largest single source of confidently-wrong answers in any internal AI deployment.

File 3 — Source-of-Truth Map

For each major question type the team asks, the canonical connector. This is the structural realization of the source-of-truth philosophy from the description. It updates whenever a tool is added or replaced.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
The recurring question types the team asks. The primary canonical source for each. Tiebreaker noted where overlap exists.	EXAMPLE Processes & tasks → the task tool. Long-form frameworks → the knowledge base. Recent decisions → chat (with owner). Deliverables & assets → the drive. Dev work → the issue tracker. Calls & prospects → the recorder + email. Pipeline → the CRM. Performance → the BI surface. Finance → the accounting system.

FIXES The assistant pulls a stale document from one source when the current answer lives in another.

File 4 — People & Roles Directory

Decision domain → role → current name. Roles outlive people, so the file's structure reflects that: when a person leaves, the role mapping doesn't change — only the name in the cell does. This file converts the assistant from an answer engine into a routing engine.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
Eight to twelve decision domains. Role owner for each (by title). Current name(s) in that role. Updated when roles or people change.	EXAMPLE Strategy & positioning → Head of Strategy. Hiring & people → Head of People. Tooling & access → IT Lead. Account decisions → the account owner (by client). Finance & contracts → Finance Lead. Data & AI systems → Head of Data.

FIXES Dead ends. Most questions end in "okay, so who do I talk to?" — this file answers that instantly.

File 5 — Quarter Priorities

The top company objectives this quarter, key results under each, team-level priorities. Anchors every strategic question in current reality rather than generic best practice. This is the only file that updates faster than quarterly — and the discipline to keep it current is the single most fragile part of the architecture.

PRINCIPLE (UNIVERSAL)	EXAMPLE — NORTHWIND (FICTIONAL)
Top company objectives this quarter. Key results or measurable targets per objective. Team-level priorities flowing from each. Dated header; quarterly refresh calendared; owner named for updates.	EXAMPLE Objective: lift net revenue retention to 115%. KR: reduce mid-market churn to <2% monthly. Team priorities: ship the onboarding redesign; stand up the health-score alerts; close the top-20 renewal list. Owner: exec team.

FIXES Makes the assistant strategically aware, not just operationally knowledgeable. Without it, every strategic question gets answered as generic best practice.

What does not belong in files

The temptation is to upload everything. Resist it. Specifically excluded from always-on context:

- **Full handbooks or playbooks** — too long, too much unused. Keep them in the drive and let connectors search them.
- **Role-specific documents** (a manager's handbook, dev guidelines) — only relevant to a subset of users.
- **SOPs and process docs** — change too often. Use the connector instead.
- **Sensitive data** (contracts, salaries, client financials) — wrong layer; use permission-aware connectors.
- **Personal notes or workflows** — belong in personal projects, not the org-wide one.

05 · LAYER THREE

Connectors (Operational Data)

Connectors give the assistant searchable, permission-aware access to the live work product flowing through your company. Coverage and canonicality matter far more than count.

Where the description holds culture and the files hold structure, the connectors hold real-time operational reality: documents being edited right now, messages being sent, tasks being moved, calls being recorded, metrics being logged. The assistant queries connectors on demand; the description and files tell it where to query and how to interpret what it finds.

> COVERAGE BEATS COUNT
> At minimum you need a documents connector and a chat connector; an email connector is strongly recommended. Beyond that, connecting fifteen tools without declaring canonical ownership just multiplies ambiguity. Connecting six with a clean Source-of-Truth Map file outperforms fifteen without.

The four audit questions

1. **Coverage** — for every category of question the team asks, is there a connected source of truth?
2. **Canonicality** — when multiple connectors could answer, which one is authoritative? Enforced in the Source-of-Truth Map file.
3. **Permissions** — what does each connector expose, and to whom?
4. **Friction** — do users re-authenticate constantly, or is access smooth?

The seven categories

Every knowledge-work organization needs the seven retrieval shapes below. Each category is defined by the kind of retrieval the assistant performs there — not by the content stored in it.

#	CATEGORY	RETRIEVAL SHAPE	PERMISSION SENSITIVITY
1	**Documents & Knowledge**	Durable, narrative knowledge artifacts. Long-form, written, persistent.	Low–medium
2	**Communication & Decisions**	Time-stamped exchange between humans. Threaded, dated, conversational.	Medium
3	**Tasks & Operations**	State of work in motion. Assignees, statuses, due dates, dependencies.	Low–medium
4	**Customer & Pipeline**	Per-account records and conversation history. The most permission-sensitive category.	**High**
5	**Performance & Data**	Numeric facts over time. Metrics, dashboards, periodic reports.	Medium–high
6	**Creative & Assets**	Binary creative work product. Designs, videos, finished assets.	Low
7	**Custom / Internal Systems**	Proprietary systems lacking native connectors. Use sparingly — each is a maintenance burden.	Varies

Permission boundaries are part of the design, not an afterthought

Retrieval without permission-awareness is a data-leak waiting to happen. The same question asked by two people should return two answers, each scoped to what that person is allowed to see. Three boundaries to draw explicitly before you connect anything sensitive:

- **Identity passthrough.** The assistant should retrieve as the user, inheriting their permissions — not with a superset service account that sees everything.
- **Category caps.** The most sensitive categories (Customer & Pipeline, Performance & Data, anything financial or HR) get the tightest exposure rules, declared in the Hard Rails section of the description.
- **Confidential exclusions.** Channels, folders, and documents marked confidential are never in scope — enforced at the connector, not left to the model's discretion.

> FIXES An assistant that helpfully surfaces salary data, a competitor's under-NDA deliverable, or a board doc to someone who should never have seen it.

06 · THE FOUNDATION

Source-of-Truth Discipline

Connectors only work if the underlying systems actually contain canonical data. One of the most common reasons intelligence layers underperform is not the AI — it is that the data the AI is meant to retrieve is scattered across the wrong places, named inconsistently, or never captured at all.

The observation

Most companies treat their internal tools as personal preferences. One team lead drops SOPs into their drive. Another writes them inside the wiki. A third keeps them as PDFs in direct messages. The SOPs exist — but not in any single canonical place, and they collide when they overlap. The same pattern plays out for call transcripts, deliverables, decision logs, and almost every category of structured knowledge a growing company produces.

This is not an AI problem. It is an organizational hygiene problem that the AI exposes brutally — and it has to be solved before the connector layer can do its job.

The four declarations

For every category of structured knowledge the company produces, declare four things, with no exceptions. This is the provenance model that keeps every answer traceable:

DECLARATION	WHAT IT FIXES
Provenance — where it lives	One canonical tool, not "depending on the team." The moment SOPs are allowed to live in either the task tool or the wiki, the architecture collapses.
Freshness — how current it must be	A declared recency window per category, and a dated header. The assistant surfaces the date so the user can judge staleness.
Ownership — who owns it	A named role (not necessarily a named person, since roles outlive people) accountable for keeping the canonical source complete and current.
Routing / escalation — how it gets there and who to escalate to	A manual rule, an automated route, or both — plus the human owner to route to when a question exceeds the assistant's confidence or remit. The fewer manual rules, the better.

The categories that matter most

Each content type below routes into one of the seven connector categories. Without that routing discipline, the connectors are searchable but the searches return scattered or partial results.

CONTENT TYPE	CONNECTOR CATEGORY IT LIVES IN
SOPs and processes	Tasks & Operations
Frameworks and strategy docs	Documents & Knowledge
Decisions (recorded async)	Communication & Decisions
Meeting notes	Communication & Decisions
Call transcripts (sales, client, internal)	Customer & Pipeline
Client deliverables	Creative & Assets
Performance reports and dashboards	Performance & Data
Financial records	Performance & Data

FIXES The AI retrieves from the canonical source, but the canonical source is half-empty because team members store things elsewhere. The connector works; the discipline fails; the user concludes the AI is wrong.

SEQUENCE MATTERS

The source-of-truth audit comes **before** the connector audit. There is no point declaring a canonical source if that source does not, in practice, contain the content it should. And this discipline cannot be installed by an AI workstream alone — it requires senior leadership to declare canonical locations, enforce them, and review compliance quarterly. Without that commitment, every other layer is built on sand.

07 · HOW THE THREE LAYERS COMPOUND

One question, all four parts working

The architecture earns its complexity only when the layers reinforce each other. The cleanest way to see it is to watch the assistant work a single question in sequence — which layer fires when, and what each contributes.

> USER QUESTION
> "What's our current position on discounting for at-risk renewals? I want to brief a new customer-success manager."

The sequence, with all four parts working

MOVE	CARRIED BY	WHAT HAPPENS
1. Interpret	Description	The persona recognizes "at-risk renewals" as a domain with internal frameworks, and reads "current position" as recent-decision territory — which the source-of-truth philosophy says lives in chat, within a declared recency window.
2. Locate	Files	The Source-of-Truth Map confirms recent decisions live in chat and frameworks in the knowledge base. The People & Roles directory names the current owner of renewal policy.
3. Retrieve	Connectors	The connectors pull the live content: the active discounting framework, the relevant chat threads, the current save-play tasks, the recent renewal-call transcripts. Because the foundation holds, those locations actually contain the canonical content.
4. Judge	Description	Conflict-resolution rules compare framework vs. chat. Where they overlap, the more recent source wins and the discrepancy is surfaced, not hidden. Numbers and named commitments are quoted verbatim.
5. Output	Description	Voice and operating principles shape the response: concise, internally-voiced, every claim cited, ending with "if you want to confirm with the owner, it's [Name]."

The same question, with weak layers

Subtract any one piece and the chain breaks at a specific point:

MISSING PIECE	WHERE IT BREAKS
Weak persona	A fluent answer in generic best-practices language. No internal frameworks, no internal vocabulary. Reads like a consulting deck on a topic you live every day.
Missing files	Internal terms get translated or guessed. With no Source-of-Truth Map, the assistant picks the most-linked doc — eighteen months old — and presents it as current.
Broken foundation	The assistant retrieves dutifully from the wiki, but the current framework lives in someone's personal drive because no canonical location was enforced. Retrieval succeeds; the answer is wrong.
No escalation philosophy	The answer ends in a confident period rather than "confirm with [Name]." The user has no signal to verify, and treats the output as final.

> Any one layer in isolation is a marginal improvement. The value comes from the sequence — interpret, locate, retrieve, judge, output — each move carried by the layer designed for it, each useless without the others.

08 · IMPLEMENTATION CHECKLIST

A practical build sequence

Build in the reverse of the order most companies try. Description first, files second, connectors third — and the source-of-truth audit before you connect anything. Work top to bottom.

PHASE 0 · FOUNDATION (BEFORE ANY AI WORK)

- [ ] List every category of structured knowledge you produce (SOPs, decisions, transcripts, deliverables, reports, financials).
- [ ] For each, declare the four: provenance (one canonical tool), freshness (recency window), ownership (a role), routing/escalation (how it gets there, who to escalate to).
- [ ] Audit whether each canonical source is actually complete — or half-empty because people store things elsewhere.
- [ ] Get a senior sponsor to enforce the "no exceptions" rule and calendar a quarterly compliance review.

PHASE 1 · DESCRIPTION (THE CULTURE LAYER)

- [ ] Write the persona as six dials — identity, voice, posture, what it cares about, what it refuses, failure mode — not one paragraph.
- [ ] Add disambiguation framing: one paragraph on what kind of business you are.
- [ ] Write operating principles, source-of-truth philosophy, and escalation posture as principles that point to files for specifics.
- [ ] Set hard rails: the behaviors that are off-limits regardless of context.
- [ ] Keep it compact and stable — it should change roughly quarterly, not weekly.

PHASE 2 · FILES (THE STRUCTURE LAYER)

- [ ] Company One-Pager — the facts a new hire needs on day one.
- [ ] Glossary — every internal term that collides with an industry default. Highest ROI; do it first.
- [ ] Source-of-Truth Map — each recurring question type mapped to its canonical connector.
- [ ] People & Roles Directory — decision domain → role → current name.
- [ ] Quarter Priorities — objectives, key results, team priorities, with an update owner.
- [ ] Run each candidate file through the four-test bar; keep the full set scannable.

PHASE 3 · CONNECTORS (THE OPERATIONAL LAYER)

- [ ] Connect the minimum first: documents, chat, email.
- [ ] Cover all seven categories only where a real question type needs them — coverage over count.
- [ ] Set identity passthrough so the assistant retrieves as the user, with their permissions.
- [ ] Enforce confidential exclusions at the connector, not in the prompt.
- [ ] Run the four audits: coverage, canonicality, permissions, friction.

PHASE 4 · OPERATE

- [ ] Pressure-test with real questions across roles; confirm answers cite sources and route to owners.
- [ ] Confirm the same question returns permission-appropriate answers for different users.
- [ ] Calendar the quarterly refresh of the description and the five files; name owners.
- [ ] Track where the assistant says "I don't know" — those gaps are your next files and connectors.

09 · SELF-ASSESSMENT SCORECARD

Score your company brain

Rate each row 0–3. Total the score, then read the band below. This is the same diagnostic we run at the start of a Company Brain architecture review.

0 = absent · 1 = ad hoc / partial · 2 = defined but inconsistent · 3 = disciplined and maintained

DIMENSION	WHAT "3" LOOKS LIKE	SCORE
Description — Persona	Six dials written and deliberate; the assistant sounds like a colleague, not a chatbot.	___ / 3
Description — Rails & escalation	Hard rails and escalation posture explicit; the assistant routes instead of guessing on high-stakes questions.	___ / 3
Files — Glossary	Every colliding internal term defined; no semantic guessing.	___ / 3
Files — Source-of-Truth Map	Each recurring question type mapped to one canonical source, with tiebreakers.	___ / 3
Files — People & Roles	Decision domains mapped to roles and current names; the assistant routes reliably.	___ / 3
Files — Quarter Priorities	Current objectives and KRs present and refreshed on schedule.	___ / 3
Connectors — Coverage	Every question category has a connected source of truth.	___ / 3
Connectors — Permissions	Identity passthrough and confidential exclusions enforced at the connector.	___ / 3
Foundation — Canonical sources	Each knowledge category has one canonical, complete home — no exceptions.	___ / 3
Foundation — Ownership & cadence	Named owners and a calendared quarterly compliance review.	___ / 3

Total: _____ / 30

0–12 · Experiment
The assistant is a novelty. Adoption is decaying. Start with the foundation and a real description before adding tools.

13–22 · Emerging
The bones exist but the layers aren't coordinated. The gaps are costing you trust. Tighten files and permissions.

23–30 · Compounding
A governed company brain. Now push the frontier: begin capturing decision trees (Section 10).

10 · THE FRONTIER

The Decision Layer

From answering questions to walking decisions — the next maturity layer above the architecture.

The blueprint so far makes the assistant reliable at answering questions. The next layer makes it useful at participating in decisions. The hard part: most company decisions today are tacit. The expert walks them in their head in ninety seconds — but the walk is a real walk, with finite criteria and a finite set of outcomes at each step. Until that walk is explicit, the assistant can only guess. Once it is explicit, the assistant can walk it too.

Walking one decision (fictional example)

> TRIGGER
> "A customer-success manager wants to approve a non-standard renewal discount for an at-risk account. What do we offer, and who signs off?"

An expert does the following in their head, in roughly ninety seconds. Each step has a finite set of possible answers, a criterion that narrows the option space, and a commit to one branch.

STEP	THE WALK
1. Is the data fresh enough to decide?	Possible: yes · no · need more data. Evidence: date of the latest usage and health report. Criterion: within the declared decision window. → Yes, current.
2. Is the account actually at-risk by our threshold?	Possible: yes · no · borderline. Evidence: health score, usage trend, support load. Criterion: the at-risk threshold for this tier. → Yes — health below threshold two months running.
3. Where is the risk coming from?	Possible: price · product gap · support · champion left. Evidence: recent call transcripts and ticket themes. Criterion: which factor dominates. → Champion left; the new owner questions value.
4. What option set fits the diagnosis?	A hold price · B standard save offer · C discount + commitment · D exec escalation. Criterion: which options are coherent with Step 3. A and B are too weak for a lost-champion risk; D is premature. → Viable: C.
5. Which option, and what action?	→ Offer a modest discount tied to a multi-quarter commitment and a re-onboarding for the new owner. Staged actions: draft the offer, draft the internal approval note, draft the re-onboarding plan — hold all from sending.
6. Who approves before commit?	Possible: account owner · CS lead · Finance. Criterion: a discount beyond standard routes to Finance for sign-off. → Route to Finance Lead.
7. Outcome metric, and when to check?	Metric: renewal closed at target margin, plus health score recovering within 60 days. Review trigger: next renewal review, or the moment health crosses back above threshold.

> THE REVEAL

> What the expert is doing is walking a defined tree. The walk is fast because they've internalized the steps, the criteria, and the option sets over a thousand iterations. But it is a walk — and a walk can be mapped.

What to capture, per recurring decision

Without an explicit tree, an AI cannot walk your organization's decision process reliably. It can describe what a health score means, but it does not inherently know whether you've crossed the threshold for this tier, or whether Option C is coherent with the Step 3 diagnosis. Mapping is the work. Capture each recurring decision once:

ELEMENT	WHAT IT CAPTURES
Trigger	The condition or event that activates this decision.
Evidence	The data points the expert gathers, with declared sources.
Logic check	The criterion that narrows possible answers at each step.
Option set	The finite, closed set of possible decisions at this step.
Action set	The finite set of staged follow-ups for each option.
Approval gate	Who must approve before any action commits.
Outcome metric	What defines whether the decision was right, and when to check.

Build one per recurring decision type. Over time the company accumulates a library of decision trees that constitute its operational judgment — and the structure the AI needs to participate in those decisions.

> The blueprint captures knowledge. The decision layer captures judgment. Judgment is not magic — it is a walk through a tree the expert has internalized. Make the tree explicit, and the assistant can walk it too: present the options, weigh the criteria, stage the action, route for approval, record the outcome. The expert still decides. The assistant makes the walk traceable — so the company stops losing its own judgment.

11 · NEXT STEP

Design your company brain.

Reading this gives you the model. Mapping it onto your company — what the assistant is called, what lives in your glossary, what you declare canonical, who owns what, which decisions to capture first — is the work that turns a connected assistant into a governed company asset.

That's what I do. This blueprint is informed by two years as the first engineer and sole architect of REMI, a production multi-agent GTM operating system I owned end to end and grew into a four-person team — plus more than 20 agents developed across Python, TypeScript, Go, and custom frameworks. I architect and implement governed AI intelligence layers for people and agents, and it starts with a diagnostic.

1. Company Brain architecture review — I run the scorecard on your real setup and map the gaps.
2. Architecture & source-of-truth blueprint tailored to your stack and org.
3. Advisory or hands-on implementation — description, files, connectors, and the first decision trees.

Request a Company Brain architecture review →

Federico Jan · Enterprise AI agent architect & GTM engineer
fede@wedreamlabs.io · Read the production case study