What an Owned AI Stack Buys You

Benefits and Risk Management

Whitepaper, November 2026

Executive Summary

A CIO who puts an AI gateway in front of every model call is not buying a technology. They are buying back six kinds of control that the current way of adopting AI quietly gives away: control over which vendors you depend on, what you spend, what gets used, what is secure, what happens to your data, and what quality your systems deliver.

This paper takes those six areas one at a time. For each one it sets out what you gain, what you avoid, and what it actually takes to get there, because the third part is the one that usually decides whether a programme succeeds.

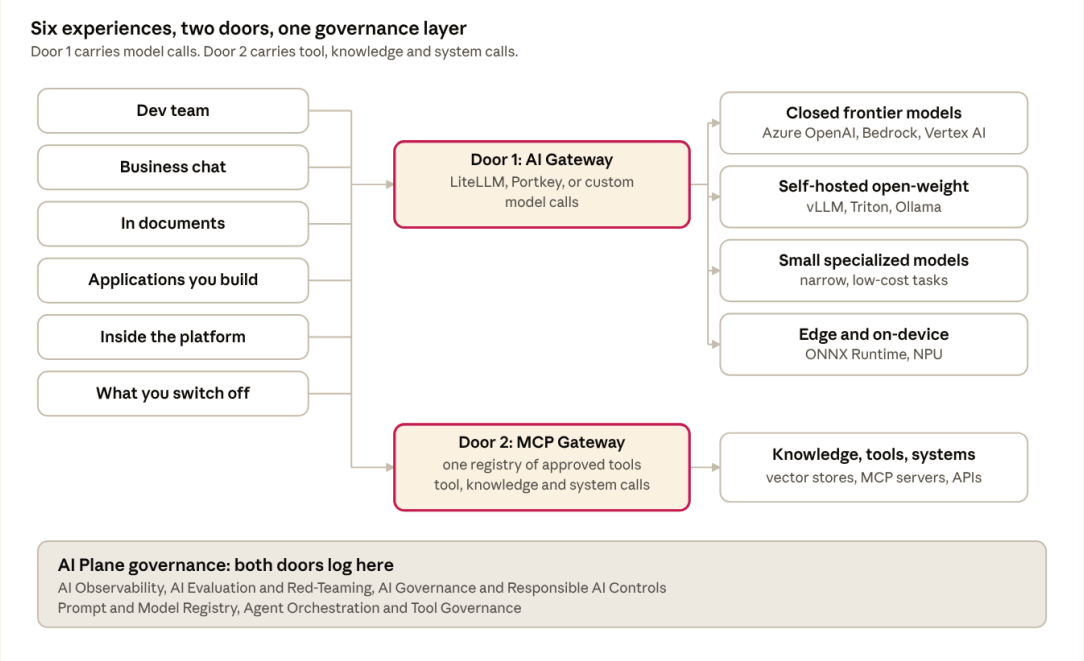
It also makes one honest admission, in a section of its own. The gateway becomes a single point of failure in your estate. That is a real risk and it deserves a real answer, which is the same answer you already apply to your identity provider and your primary databases.

The first section explains the concept in two pages, for a reader who has not read the companion paper. Everything after that assumes it.

The Concept, in Two Pages

Most enterprises adopt AI the way they adopted software-as-a-service: one decision at a time. A department buys an assistant. A development team adopts a coding tool. A product team wires a model API into an application. Each decision is defensible on its own. The result is an estate of direct, point-to-point connections from inside your organization to outside it, each with its own credentials, its own price list, its own roadmap and, usually, no shared log.

The alternative is to treat AI as a platform capability rather than a set of purchases. In practice that means putting a layer in the middle that you own, and making everything behind it replaceable. Enterprise IT has done this before with data platforms, API gateways and identity providers. The AI version of that layer is the gateway.



The architecture in one picture: six entry points, two governed doors, four options in the model inference layer, one governance layer.

There are two kinds of AI traffic in an enterprise, and architecturally they get two doors.

- **Door 1 is the AI Gateway.** Every model call goes through it. It maps a logical model name to a physical model in the inference layer, applies the approved model list and the budget, provides a fallback when a provider fails, and records every call with the caller, the team, the model, the tokens and the cost.
- **Door 2 is the MCP Gateway.** Every call to a tool, a knowledge source or an internal system goes through it. It holds one registry of approved tools, gives each agent its own non-human identity with least privilege, and makes sure that data a tool returns follows the access rights of the person who asked.

On stage we compress this into one sentence: put all of your AI through one door. That is the right instruction. The precise version is that there are two doors in the architecture, and whether they are one product or two is an implementation choice. Some products cover both;

LiteLLM, for example, offers model routing and MCP gateway features in one product, according to its own materials. A gateway written as custom code can also combine both. What matters is that nothing reaches a model or an internal system without passing through infrastructure you own.

Underneath both doors sits one governance layer: one observability instance that receives the logs from both doors, an evaluation and red-teaming gate that a new agent or model version passes before wide rollout, and a registry that versions prompts and model configurations the way you version source code.

Six things use this architecture. Five are entry points: development teams through their IDE agents, employees through one business chat application, people working inside Word and Outlook through an add-in, the applications your own teams build, and the platform tooling you already run that adds an AI step. The sixth is not a way of using AI at all. It is a loop, fed by the logs from both doors, that tells you what to switch off.

That is the concept. The companion paper, **The Architecture Behind an Owned AI Stack**, draws draws each of the six on the same template with the building blocks in order, and is the right place to go for the architectural detail. The rest of this paper assumes the architecture exists and asks the question a CIO actually has to answer: what does it buy you, and what does it cost you to get there.

Vendor and Continuity

This is the benefit that is easiest to dismiss as theoretical, right up to the moment it is not.

What You Gain

- You can change model provider without touching your applications. The application calls a logical name such as "general-chat". Which vendor serves that name is a routing decision at Door 1.
- You can run several providers at once, by workload, by data classification or by region, without multiplying the integration work.
- You can negotiate from a position where leaving is technically possible. A vendor conversation changes when switching is a configuration change rather than a project.
- You keep the prompts, the evaluation sets, the routing policies and the usage history. Those are the assets that make the next migration cheap, and they live in your registry, not in a vendor console.

What You Avoid

- A price increase you have no practical answer to, because every application is wired directly to one provider.
- A model deprecation that breaks production. Providers retire model versions on their own schedule, and an application that names a version directly inherits that schedule.
- A vendor that is acquired, changes its terms, changes its roadmap or disappears. Consolidation in this market is active, and the AI gateway space itself has already seen acquisitions.
- A provider outage becoming your outage, because there is no second path configured.

What It Takes to Get There

Portability is not free, and claiming otherwise is how this benefit gets oversold. Models are not drop-in replacements for each other. A prompt tuned on one model will behave differently on another, so the real enabler of vendor independence is not the gateway alone. It is the gateway plus an evaluation set per use case that tells you, quickly and objectively, whether a different model still does the job. Without that evaluation set, you have the technical ability to switch and no confidence to use it.

The practical rule is that the logical model name must never leak a vendor name into application code, and that every use case with real business impact has an evaluation set from the day it goes live.

Cost

Cost is where most of these conversations start, and it is usually framed as a question about unit price. The structural problem is not the price. It is that nobody can attribute the spend.

What You Gain

- Cost per team, per application and per use case, as a by-product of normal operation rather than as a reporting project.
- Budgets and rate limits that are enforced at the door, before the spend happens, instead of discovered on the invoice.
- Routing of cheap work to cheap models. A large share of enterprise AI work is narrow and repetitive, and a small model does it at a fraction of the cost of a frontier model. Door 1 can make that decision centrally.
- Caching and deduplication at one point in the estate, so the same question asked a thousand times is not paid for a thousand times.
- The data you need to decide whether self-hosting an open-weight model is justified for a specific workload. That decision requires volume data you do not have today.

What You Avoid

- A consumption invoice that cannot be explained, defended or forecast.
- Seat-based licences bought for a whole department and used by a fraction of it.
- A machine-to-machine workload that scales quietly. A pipeline running a thousand times a day makes a thousand model calls a day, and nobody complains until the invoice arrives.
- Paying frontier-model prices for classification, extraction and summarization work that a small model handles.

What It Takes to Get There

Two things, and only one of them is technical. The technical part is tagging: every call must carry a team, an application and a use case, which means identity has to reach the gateway correctly from every entry point. A call that arrives with a shared service account is a call you cannot attribute, and it will be the most expensive one.

The part that is not technical is deciding what the budgets are. A budget per team is a governance decision that someone has to take and defend. Until it exists, the gateway can measure spend but cannot control it, and measurement on its own has never reduced a bill.

Usage and Adoption

This is the benefit CIOs recognize fastest, because it is the one they have been asked about in a board meeting and could not answer.

What You Gain

- An honest answer to the question of who actually uses AI in your organization, broken down by team, by tool and by use case.
- The ability to retire what nobody uses. You cannot switch off what you cannot measure, and today most AI spend is unmeasured at the level where the decision gets taken.
- Visibility of shadow AI. Once there is a sanctioned front door, traffic that does not come through it becomes detectable rather than invisible.
- Evidence for adoption programmes. You can see which departments are stuck, which use cases spread on their own, and which training actually changed behaviour.
- A clean offboarding path. When someone leaves, access through the door is removed once, rather than across a list of tools nobody maintains.

What You Avoid

- Renewing a licence agreement without knowing whether it is used.
- An adoption debate based on anecdotes and on whoever is loudest in the room.
- Staff using a public AI tool with company data because the sanctioned option is slower, worse or absent.
- Discovering a departed employee still has an active AI account six months later.

What It Takes to Get There

The sanctioned path has to be at least as good as the unsanctioned one. This is the single biggest cause of failure in this part of the architecture. If the governed chat application is slower, has an older model or refuses more often than the public tool people already use, they will keep using the public tool and your usage data will describe an estate that does not exist.

Measuring usage also means someone has to act on it. Retiring a licence that a vocal department asked for is a conversation. The data does not remove that conversation, it changes it from an argument about opinions into a review of measured facts.

Security

AI security discussions tend to focus on the model and on what it might say. The larger enterprise exposure is in the plumbing around it.

What You Gain

- One identity path. Every call carries the identity of the person or the agent making it, from your own identity provider, with your own multi-factor and conditional access rules already applied.
- No API keys in application code, in configuration files or in a developer notebook. The application authenticates to your gateway, and only the gateway holds provider credentials.
- A filter on both directions at one point: what may leave the organization in a prompt, and what may come back in a response.
- Non-human identity with least privilege for every agent at Door 2, instead of agents inheriting the rights of whoever deployed them.
- One audit trail that can reconstruct an incident. Which agent, which system, which identity, which arguments, in what order.

What You Avoid

- Credentials scattered across teams, with no central way to rotate or revoke them.
- An agent with broad write access to a production system because that was the quickest way to make a demonstration work.
- Prompt injection reaching a tool that can act, because nothing between the model and the system enforced what that agent is allowed to do.
- An AI incident you cannot reconstruct, which is the version of an incident that cannot be closed.

What It Takes to Get There

Door 2 is where this benefit is earned, and it is the harder of the two doors. The work is not the gateway itself. It is deciding, per system, what an agent may read and what it may change, and then expressing that as scopes rather than as a sentence in a design document.

Start read-only, with a human approving every write, and widen per system using the audit trail from the first phase as evidence. This is slower than teams want, and it is the difference between an agent programme that survives its first incident and one that does not.

Data Governance and Compliance

This is the area where the gap between what an organization believes it does and what it can prove is widest.

What You Gain

- A control point that decides which data classification may reach which model. Confidential data to a self-hosted model, general questions to a closed API, by policy rather than by instruction.
- Residency control. Where a model runs is a routing decision you make, not a property of whichever vendor a team happened to choose.
- Retention and training opt-out enforced centrally, instead of checked once per contract and never again.
- Evidence. An inventory of AI systems, a record of which models are in use and for what, logs of what was sent and returned, and documented evaluation before rollout. These are exactly the artefacts that regulatory obligations ask for, and they fall out of the architecture rather than being produced for an audit.
- Retrieval results that follow the access rights of the person asking, so an assistant cannot become a route around your document permissions.

What You Avoid

- Confidential material in a prompt to a provider whose terms you have not read.
- An assistant that surfaces documents a colleague technically had access to but was never meant to see. This is not a new problem. Enterprises met it a decade ago when Microsoft shipped Delve, which exposed years of over-broad file permissions all at once, and many of them chose to switch the feature off rather than fix the permissions underneath.
- An AI inventory assembled by sending a questionnaire to department heads.
- A compliance position that depends on nobody asking a detailed question.

What It Takes to Get There

A data classification that exists and is applied. The gateway can enforce a rule such as "confidential never leaves the organization" only if something in your estate knows what is confidential. Most organizations have the classification on paper and not in the systems, and that gap becomes visible on the day the rule is switched on.

It also takes a permissions review before an assistant goes live over a document store, not after. Retrieval does not create exposure, it reveals it, and the fastest available fix after the fact is always to switch the assistant off.

Quality and Reliability

The least discussed of the six, and the one that decides whether anybody trusts the output a year from now.

What You Gain

- Pinned model versions, so your behaviour does not change because a provider shipped an update on a Tuesday.
- An evaluation gate. A new model or a new agent is measured against a known set before it reaches users, not after they complain.
- Graceful degradation. When a provider is slow or rate limited, Door 1 falls back to another backend rather than returning an error to a user.
- A prompt registry, so the prompt running in production is versioned, reviewable and attributable, like any other piece of logic.
- Latency and error rates measured per use case in one place, which is the only way to tell a model problem from an integration problem.

What You Avoid

- Silent regression. A model update changes the shape of an output, a downstream parser breaks, and nobody connects the two for a week.
- Prompts edited directly in production by whoever has access to the add-in or the application configuration.
- A rollout decision based on a demonstration rather than on a measurement.
- Users who stop trusting the system because it was wrong twice and nobody could say why.

What It Takes to Get There

Evaluation sets, and the willingness to maintain them. This is genuine work: for each use case that matters, a set of representative inputs and an agreed definition of a good answer. It is the same discipline as a regression test suite, and it decays the same way if nobody owns it.

The payoff is that almost every other benefit in this paper depends on it. Changing vendor, moving work to a cheaper model and approving a new agent all require an objective way to say that quality held. Without evaluation sets, you have the mechanisms and no basis to use them.

The Gateway Is Itself a Dependency

A paper that listed only benefits would not be worth reading, and this is the objection that a good architect raises within the first ten minutes.

If every AI call in the enterprise goes through one door, that door is on the critical path of every AI feature you run. If it is down, all of them are down. If it is slow, all of them are slow. If its configuration is wrong, the error is enterprise-wide rather than local. You have concentrated risk on purpose.

This is true, and it is the same trade-off you have already accepted elsewhere. Your identity provider is a single point of failure. Your primary database is a single point of failure. Your network core is a single point of failure. In each case you did not avoid the dependency, because the alternative, which is every application doing its own thing, is worse in every dimension that matters. What you did instead was manage it.

Managing it means the same short list it has always meant.

- A named owner and a place in the platform team, not a side project of whoever built it.
- An availability target that is written down, with monitoring and alerting against it.
- Redundancy appropriate to that target, and a failover path that has actually been tested rather than designed.
- Capacity planning, because the gateway inherits the growth of every entry point behind it at once.
- Change control on policy, because a routing or budget rule is production configuration with an enterprise-wide blast radius.
- A run budget. The gateway is a platform service and it has a cost, separate from what flows through it.

There is a second, quieter failure mode worth naming. The governance layer does not fail with an outage. It fails by decay: a dashboard nobody reads, an evaluation gate that gets waived under delivery pressure, a tool registry that no longer matches what is actually deployed. The architecture diagram still shows a governed estate while the reality has drifted back to an ungoverned one. That failure mode has no alert, which is why the switch-off loop and a regular review of the registry are part of the design and not optional extras.

How You Know It Worked

Six benefits are worth little without a way to tell whether you got them. These are the measures that follow directly from the architecture, so none of them require a separate reporting effort.

- **Coverage.** The share of AI calls in the organization that pass through Door 1. This is the master measure. Every other number is only as good as this one.
- **Attribution.** The share of spend that can be assigned to a named team, application and use case. Anything unattributed is a gap in identity, not in finance.
- **Cost per unit of work.** Cost per thousand calls or per use case, tracked over time. This is the number that shows whether routing and caching are working, and it should fall even when total spend rises.
- **Active usage.** Users and agents active in the last thirty days against seats and licences paid for. The difference is your retirement list.
- **Time to a new use case.** Working days from approval to production for a use case that uses existing doors. If this is not falling, the platform is not yet a platform.
- **Evaluation coverage.** The share of production use cases with a maintained evaluation set. This predicts your ability to change anything safely.
- **Door 2 scope hygiene.** The number of agent identities with write access, reviewed on a schedule. This number should be known and small.

Report these together and the picture is usually uncomfortable in the first quarter and useful by the second. That is the normal shape, and it is worth telling your stakeholders in advance so that the first report is read as a baseline rather than as a failure.

Where to Start

The order matters more than the speed, because each step makes the next one cheaper.

1. **Build Door 1 with one entry point.** Development teams are the usual choice. Route everything to a closed API and do not self-host anything yet.
2. **Turn on observability on day one.** The logs from the first weeks justify every later decision, and you cannot reconstruct them afterwards.
3. **Move business chat behind Door 1.** This is where shadow AI stops, and where you find out how much of it there was.
4. **Write the budgets and the model policy as code.** Until these exist, the gateway can measure but not control.

5. **Add Door 2, read-only first.** One registry, narrow scopes, a human approving writes, widened per system on evidence.
6. **Run the switch-off loop.** Review usage against policy on a fixed schedule and actually retire something. The first retirement is what makes the loop credible internally.

Conclusion

Every benefit in this paper comes from the same move: putting a layer you own between your organization and the AI it uses, and making everything behind that layer replaceable. Vendor independence, cost attribution, usage visibility, security, data governance and quality control are not six separate programmes. They are six consequences of one architectural decision.

None of it is free. Each of the six sections above ends with what it takes, and in most cases the hard part is not the engineering. It is a governance decision that has to be taken in writing before the engineering can enforce it: what a reasonable budget is, what data may go where, what an agent may change, what good enough means for a given use case.

That is also the reason this is worth doing now rather than later. Those decisions get harder as the estate grows, and every month of unmanaged adoption adds to the list of things that will have to be migrated rather than designed.

This Deserves a Conversation

The six areas in this paper are each a conversation on their own, and which of them matters most depends entirely on your situation. A heavily regulated organization starts from data governance. One with a cost problem starts from attribution. One that has already deployed agents usually has a Door 2 problem it has not named yet.

This is too much to work through in a follow-up email, and interesting enough that it deserves a proper discussion. If you want to go through it against your own estate, with your own architects in the room, we are happy to sit down with you. No obligation and no deck.

Diederik Wyffels, Managing Partner, 45 Degrees. diederik.wyffels@45degrees.be

Bert Teuwen, Managing Partner, 45 Degrees. bert.teuwen@45degrees.be

Companion papers: **The Architecture Behind an Owned AI Stack**, the six entry points and the two doors in detail, and **The AI Nobody Is Selling You**, on why the structural alternative exists and what it is worth.