THE GROVE FOUNDATION

White Paper

The Telemetry Trap

Why Enterprise AI Contracts Are Missing the Most Dangerous Lock-In Clause in Technology History

Jim Calhoun • Founder, The Grove Foundation • April 2026

Executive Summary

Every major AI lab is building the same thing: a persistent agent that watches how you work, learns your patterns, and acts on your behalf. Anthropic's leaked Conway project is the most visible example, but Google, OpenAI, and Microsoft are all converging on the same architecture. This is not a product trend. It is a platform strategy with a specific lock-in mechanism that enterprise procurement teams are not equipped to negotiate against.

The lock-in is not about data. Enterprises have spent two decades building legal frameworks for data portability. The lock-in is about telemetry—the accumulated behavioral model an always-on agent builds by observing how your people work, what they prioritize, how they communicate, and what they ignore. That model has no export format, no portability standard, and no legal framework governing ownership. When you switch providers, you lose it. Your next agent starts from zero.

This paper makes three arguments:

First: the telemetry layer—not the model layer, not the data layer—is where the deepest vendor lock-in in technology history is being constructed. Enterprises signing AI platform contracts today are negotiating data residency clauses while ignoring the behavioral intelligence those platforms are accumulating about their organizations.

Second: the Autonomaton Pattern, an open architectural standard published under CC BY 4.0, provides a structural alternative: a governance architecture where telemetry stays sovereign, behavioral models remain inspectable, and the entire learning layer is portable by design—not by policy.

Third: enterprise procurement teams have a narrow window—months, not years—to negotiate telemetry portability terms before switching costs become prohibitive. This paper provides the structural framework for that negotiation.

I. The Conway Signal

In March 2026, a packaging error pushed approximately 500,000 lines of Anthropic's Claude Code source to a public registry. Among the internal projects revealed was Conway: a standalone agent environment with persistent memory, extension support through a proprietary .cnw.zip format, external event triggers, and browser integration. Conway is not on Anthropic's public roadmap. It is an internal project that reveals the platform strategy the company has been executing across five surfaces in a single quarter.

The significance is not what Conway does. Always-on agents are inevitable. The significance is what Conway accumulates. After six months of operation, a Conway instance doesn't just store your files and calendar entries. It holds a model of how you work: which emails you respond to immediately, which you ignore, which meetings you reschedule, which Slack threads you monitor, which decisions you delegate and which you keep. That behavioral model is the product of your data plus the provider's compute plus six months of continuous inference. It has no export path. It has no portability standard. And it represents the deepest form of vendor lock-in the technology industry has ever produced.

> *Previous lock-in was about stuff. Files, records, message history. Stuff is painful to migrate but possible. Conway-class agents lock in something different: the accumulated model of how you work. There is no CSV for "how this person thinks."*

The Platform Strategy

Conway does not exist in isolation. It is the capstone of a five-move strategy Anthropic executed in under 90 days: Claude Code Channels (developer messaging), Cowork (non-technical enterprise surface), the Marketplace (procurement layer with partner apps billing through Anthropic), $100M in partner network commitments, and the blocking of third-party tools from subscription access. Every piece pushes in one direction: build inside our walls, use our surfaces, run through our billing.

This is the Active Directory playbook. Microsoft went from operating system to desktop to application layer to enterprise identity in fifteen years. Anthropic is attempting the same arc—model provider to developer tool to enterprise platform to agent operating system—in fifteen months. Conway is the piece that makes everything else sticky, because the persistent agent that holds your organizational memory is the thing you cannot rip out.

The .cnw.zip Paradox

Anthropic published MCP—the Model Context Protocol—as an open standard. OpenAI adopted it. Google adopted it. The Linux Foundation hosts it. The entire premise is universal connectivity between AI tools and data sources. Then Conway adds a proprietary extension layer on top. Extensions packaged as .cnw.zip include custom interfaces and tools that work only inside Conway's environment. They are not portable.

This is the Google Play Services pattern applied to AI. Android is open source. Google Play Services—Maps, payments, push notifications, the Play Store—is proprietary. You can build Android without Google's services. In practice, nobody does. MCP is the open foundation. Conway's extension ecosystem is the proprietary layer on top. Anthropic gets the credibility of an open standard and the commercial advantage of a walled garden.

II. The Telemetry Trap

Enterprise procurement teams have spent twenty years developing frameworks for data portability. GDPR, CCPA, SOC 2, data residency requirements, right-to-delete provisions—the legal infrastructure for "can I take my stuff with me" is mature. Imperfect, but functional.

None of it covers what Conway-class agents accumulate.

Three Layers of Lock-In

To understand the trap, distinguish three layers of what an always-on agent holds:

Layer	What It Contains	Export Status	Legal Framework
Data	Files, messages, calendar, CRM records	Exportable (CSV, API)	GDPR, CCPA, SOC 2
Integration	API connections, workflow configs, tool mappings	Partially exportable	Contract-specific
Behavioral Telemetry	How you work: patterns, priorities, decision timing, communication style	No export path exists	No framework exists

The first two layers are negotiable. The third is not—because no one is negotiating it. Enterprise legal teams are securing data residency while the behavioral model of their entire organization accumulates inside a provider's infrastructure with no portability provision, no ownership clause, and no export format.

The Compounding Problem

Behavioral lock-in compounds. Every day Conway operates, it learns more about how your organization works. At month one, switching means losing convenience. At month six, switching means losing an agent that understands your VP's communication style, your team's deployment cadence, and which Slack channels carry signal. At month twelve, you are not switching AI providers. You are amputating institutional memory.

The switching cost curve is not linear. It is exponential. And unlike data migration—which is painful but bounded—behavioral context migration is currently impossible. Not expensive. Not difficult. Impossible. The format does not exist. The standard does not exist. The receiving system has no way to ingest it.

> *The window for negotiating telemetry terms is before deployment, not after. Every month of always-on operation makes the exit cost higher and the negotiating leverage lower.*

What No One Is Asking in the Contract Review

Having reviewed enterprise AI procurement processes across multiple verticals, a pattern emerges. Legal teams focus on data handling, model liability, and uptime SLAs. They do not ask:

> **Who owns the behavioral model the agent builds about our workflows?**
>
> **In what format can we export the derived context, not just the raw data?**
>
> **What happens to the accumulated intelligence if we terminate?**
>
> **Can we audit what the agent has learned about us—and delete specific inferences?**
>
> **Is there an open format for porting behavioral context to a competing platform?**

These are not theoretical concerns. These are contract terms that should be in every enterprise AI agreement signed in 2026. They are in virtually none of them.

III. The Structural Alternative

On March 31, 2026—the same day Anthropic's source code leak exposed the Conway architecture—the Grove Foundation quietly published the Autonomaton Pattern under CC BY 4.0. The timing was not coincidental. The leak confirmed what we had been building toward for months: that the industry's leading AI lab had constructed internally the exact governance architecture it had no incentive to open-source. Publication could no longer wait for a controlled rollout. The structural alternative needed to be in the public commons, and CIOs needed to be aware of the telemetry trap they're negotiating themselves into.

The Autonomaton Pattern is an open architectural standard for AI governance. It addresses the same problem Conway exploits—persistent agent intelligence—but with the opposite ownership model. Where Conway accumulates behavioral context inside the provider's infrastructure, the Autonomaton Pattern makes that context sovereign by architecture. Not by policy. Not by contract clause. By the structural properties of the system itself.

The Five-Stage Invariant Pipeline

Every Autonomaton instance traverses the same five stages, regardless of domain, model, or provider:

Stage	Name	Function
01	Telemetry	Capture interaction data. Structured, inspectable, owned by the operator.
02	Recognition	Classify intent, assess confidence, determine risk. The Cognitive Router dispatches.
03	Compilation	Assemble context from local knowledge, historical patterns, and domain expertise.
04	Approval	Zone-governed human checkpoint. Green = autonomous. Yellow = supervised. Red = human-only.
05	Execution	Action fires. Output becomes input for the next cycle. Audit trail generated as byproduct.

The pipeline is not a suggestion. It is an invariant. Nothing runs alongside it. No sub-pipelines. No bypasses. This constraint is what makes the pattern auditable, composable, and—critically—portable. An Autonomaton's entire learned behavior is expressed through configuration files and structured telemetry. Not through opaque weights inside a provider's infrastructure.

How the Autonomaton Solves What Conway Exploits

Dimension	Conway-Class Agent	Autonomaton Pattern
Where behavioral model lives	Provider infrastructure	Operator-owned config + telemetry files
Export format	None	Structured JSON telemetry, declarative YAML config
Audit trail	Opaque	Every pipeline decision traceable to a config rule anyone can read
Model portability	Locked to provider	Model-agnostic Cognitive Router; swap providers without rewriting governance
Skill learning	Hidden in weights	Explicit Skill Flywheel: visible, inspectable, deletable
Governance boundaries	Provider-defined	Operator-defined zones (Green/Yellow/Red) in editable config
Extension ecosystem	Proprietary (.cnw.zip)	Open standard (CC BY 4.0), any implementation
Switching cost	Exponential with time	Flat: take your config and telemetry, deploy elsewhere

The Sovereignty Architecture

Three architectural properties make the Autonomaton's portability structural rather than aspirational:

Declarative Governance. Every behavior rule is externalized in configuration files a non-technical domain expert can read and edit. Change the config, the behavior changes. Version the config, the behavior has a history. Audit the config, the behavior is explained. No separate "explainability layer" required. The governance is the architecture.

Sovereign Zones. Every action has an explicit risk classification: Green (autonomous routine), Yellow (supervised proposals), Red (human-only). Zone boundaries are declarative—defined in configuration, not hardcoded. The system earns autonomy through demonstrated reliability. It cannot unilaterally grant itself new authority. Sovereignty is structural, not aspirational.

Feed-First Telemetry. Every interaction generates structured telemetry as its primary output—not as a side effect. This telemetry serves triple duty: learning (feeds the Skill Flywheel), observability (surfaces system health), and compliance (produces audit trails). Because the telemetry is structured, inspectable, and owned by the operator, it is portable. You can take it with you. You can point a new system at it. You can audit every decision the agent ever made.

> *The test: can a non-technical domain expert alter the system's behavior by editing a config file, without a deploy? If the answer is no, the architecture does not belong to you.*

The Skill Flywheel: Visible Learning

Conway-class agents learn in opaque weights. The Autonomaton learns through an explicit, inspectable mechanism called the Skill Flywheel:

The system observes interactions, detects recurring patterns (same intent three or more times in fourteen days), proposes a skill specification the human can read, waits for approval, then executes autonomously on future matches. Usage data refines the skill over time. Skills that stop matching get deprecated.

The behavioral model is not hidden. It is a library with a card catalog. You can inspect every skill. You can correct every pattern. You can delete anything that no longer serves you. When you leave a platform, you take the library—because the skills are configuration, not weights.

IV. The Procurement Failure

The enterprise AI procurement cycle has a structural blind spot. Procurement teams are optimized to negotiate terms for categories they understand: data handling, uptime, liability, integration support. Behavioral telemetry does not fit any existing category. It is not "data" in the way procurement defines data. It is not "software" in the way procurement defines software. It is a new category of organizational asset that is being generated, accumulated, and retained by providers without explicit contractual terms governing its ownership, portability, or deletion.

The Contract Gap

A typical enterprise AI platform agreement in 2026 addresses:

Contract Term	Typical Coverage	Telemetry Gap
Data ownership & residency	Addressed	Covers raw data only, not derived behavioral models
Model output liability	Addressed	Silent on liability for agent actions based on learned patterns
Right to delete	Partially addressed	Covers data deletion; no provision for deleting behavioral inferences
Portability on termination	Partially addressed	Data export in standard formats; no export for learned context
Audit rights	Addressed	Auditable data handling; no audit of what the agent has learned about you
Competitive intelligence	Rarely addressed	No clause preventing provider from training on aggregate workflow patterns

The gap is not ignorance. It is a category error. Procurement teams are applying data-era frameworks to an intelligence-era problem. The asset being generated is not data. It is a behavioral model derived from data—and the difference is existential.

The Timeline Pressure

The window for addressing this is measured in months. Three dynamics are compressing the timeline:

Convergence. Every major lab—Anthropic, OpenAI, Google, Microsoft—is building always-on persistent agents. The architecture is converging. By mid-2027, the persistent agent will be the default enterprise AI interface, not an option.

Compounding. Behavioral lock-in compounds daily. Every day of always-on agent operation adds to the switching cost. Enterprises deploying in H2 2026 without telemetry terms will face material switching costs by H1 2027.

Contract cycles. Enterprise AI contracts are typically 12–36 months. Contracts signed in 2026 without telemetry provisions will not be renegotiable until 2028 or 2029—by which time the behavioral lock-in will be two to three years deep.

> *The negotiating leverage exists right now, before deployment, while providers still need your adoption. It will not exist after twelve months of accumulated behavioral context.*

V. The Structural Intervention

Policy alone cannot solve an architecture problem. Data portability regulations did not prevent cloud lock-in—they made migration legally possible while leaving it technically expensive. The same dynamic will play out with behavioral telemetry unless the intervention is structural.

The Autonomaton Pattern provides that structural intervention. Not as a product. Not as a platform. As an open architectural standard that any enterprise can adopt, any developer can implement, and any regulator can reference.

What the Standard Provides

A portable telemetry format. Structured JSON entries for every interaction: intent, classification, tier, zone, model, cost, outcome, human feedback. This is the behavioral context—and it belongs to the operator, not the provider.

A declarative governance schema. YAML configuration files that define routing rules, zone boundaries, and skill libraries. Any system implementing the Autonomaton Pattern can ingest these files and reproduce the agent's behavior on a different provider.

A model-agnostic routing layer. The Cognitive Router dispatches to tiers, not to specific models. Swap providers without rewriting governance. Your routing rules, zone policies, and audit trails belong to you—not to whichever API you happen to be calling this quarter.

An inspectable learning mechanism. The Skill Flywheel promotes confirmed patterns from expensive inference to cheap local execution. Every skill is a readable specification, not an opaque weight adjustment. Delete a skill and the behavior stops. Export a skill library and the next system inherits your institutional knowledge.

What Enterprises Should Demand Now

Whether or not an enterprise adopts the Autonomaton Pattern, the architectural principles establish a minimum standard for any AI platform contract signed in 2026:

1. Behavioral telemetry ownership. The enterprise owns all derived behavioral models—not just the raw data that generated them. The contract must specify that workflow patterns, priority models, communication style inferences, and decision-timing data belong to the customer.

2. Structured export on termination. On contract termination, the provider must deliver behavioral context in a structured, machine-readable format (not a data dump of

raw logs). The format must be sufficient for a competing system to reproduce the agent's learned behavior.

3. Inference audit rights. The enterprise has the right to audit not just what data the agent accessed, but what inferences it drew—what patterns it detected, what priorities it assigned, what behavioral models it constructed.

4. Inference deletion. Beyond data deletion, the enterprise can demand deletion of specific behavioral inferences. If the agent learned something about your organizational patterns that you don't want retained, you can remove it.

5. Aggregate learning restrictions. The contract must prohibit the provider from using your organization's behavioral telemetry to train models, improve products for other customers, or generate aggregate intelligence about your industry vertical.

6. Architecture disclosure. The provider must disclose the architecture of the behavioral learning system: what is captured, how it is stored, what is derived, and what would be lost on termination.

VI. The Precedent

This has happened before. The structural pattern is identical to every previous platform capture cycle in technology history:

TCP/IP was not built by AT&T. The shipping container was not designed by a shipping line. The entities that benefit most from proprietary infrastructure never build the open standard that replaces it. The open standard comes from outside the incumbents—from an institution with no revenue model tied to lock-in.

The Linux Foundation generates $300 million per year. Red Hat is a $5 billion company. Neither writes code. They write standards, certify compliance, convene ecosystems, and publish the research that keeps the open-source supply chain credible. The Grove Foundation runs the same playbook for cognitive sovereignty that Linux ran for compute sovereignty.

The Regulatory Reframe

Current AI regulation asks the wrong question. The EU AI Act, Executive Order 14110, and most regulatory frameworks focus on: which model did you use? What data did it train on? Is the output biased? These are important questions. They are not the structural question.

The structural question is: how is the system governed? Who controls the routing? Who owns the telemetry? Who can audit the behavioral model? Who can port it to a competitor? The Autonomaton Pattern shifts the regulatory frame from model accountability to architectural accountability—from asking “what model did you use” to asking “who controls the governance layer.”

An open governance standard published under CC BY 4.0 cannot be captured by a single vendor. It cannot be preempted by regulation because it is already in the public commons. It creates the structural condition for a competitive market in AI agents—a market where the switching costs are bounded by architecture, not compounded by behavioral lock-in.

VII. The Window

The industry is in a transition between two eras. The era of model competition—benchmarks, context windows, training runs—is giving way to the era of persistence competition: who owns the always-on layer, the agent that accumulates context across every session and becomes harder to leave every day.

The margins between frontier models have compressed to the point where the model itself is no longer the primary competitive axis. The strategic logic has shifted: stop winning on model releases and start winning on the layer that makes customers sticky regardless of which model sits underneath. The persistent agent layer—the thing that holds your memory, your context, your workflows—is the actual product. The model is the loss leader.

This is the moment—right now, in 2026—when the terms get set. The contracts being signed this year will determine whether enterprise AI operates on open architectural standards with portable behavioral context, or on proprietary platforms where the deepest form of organizational intelligence ever generated is locked inside a single provider's infrastructure.

> *The telemetry layer is the new moat. The question is whether you're building the moat or trapped inside it.*

The Autonomaton Pattern is not the only answer. It is the open answer. Three files and a loop. A routing config, a zones schema, a structured telemetry log, and a pipeline that traverses them in order. It proves that sovereign, portable, auditable AI governance is not a theoretical aspiration. It is an architectural property you can implement this quarter.

The window is open. It is closing. And every enterprise that deploys an always-on agent without negotiating telemetry terms is making a decision they will spend years trying to reverse.

—

The Grove Foundation publishes open architectural standards for AI governance. The Autonomaton Pattern is available under CC BY 4.0 at the-grove.ai.

Contact: jim@the-grove.ai