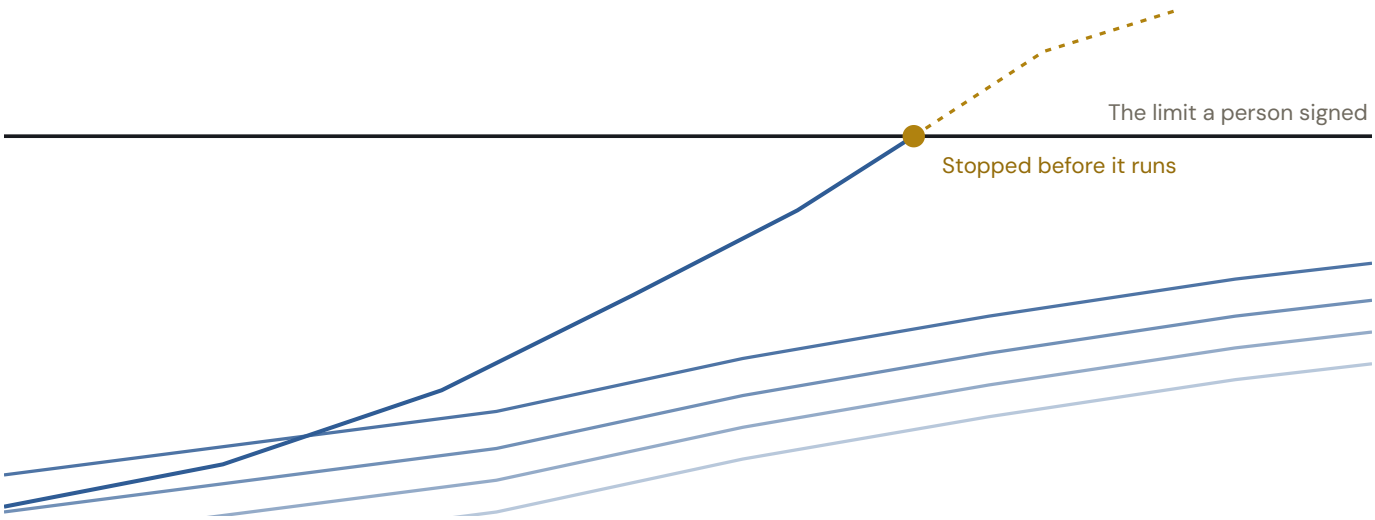
Everything *your agents touch,* in one place.

How D4A brings your rules, your automated judges and your people together, before your agents act. For your IT, security and compliance teams.



Contact diacroma@veritglobal.com
Web www.diacroma.com
By Verit Global Labs

Try it free for 30 days on Microsoft Marketplace

marketplace.microsoft.com/en-us/product/veritglobal.d4a

Before, the tangle. *After, one door.*

With D4A, every action your agents take goes through one door, where your rules are checked and the decision is signed. The money they move stays within limits a person signed.

Before: the tangle

Every agent reaches your tools on its own. Your policies live in one system, the content filters in another, the evaluators in a third, and your people hear about a problem when a customer complains. Each team wires its own checks, so the same question gets a different answer depending on the agent. Which rule wins? Who should have been told? Where is the proof?

After: the simple skein

D4A sits on the one path every agent already uses to reach a tool. Before a call goes out, it asks your sources of rules: the limits a person signed for that agent and your company policies, while the Watch reviews the pattern.† It decides, then writes the decision into a signed record that lives in your cloud. When something needs a person, it opens a case, which goes to whoever signed the limit.† Your agents don't change.

In production, on our sample company, a $150 refund asked the company's OPA rule first ("refunds over 100 need a manager"). The rule said no, the call stopped, and the reason went into the signed record.

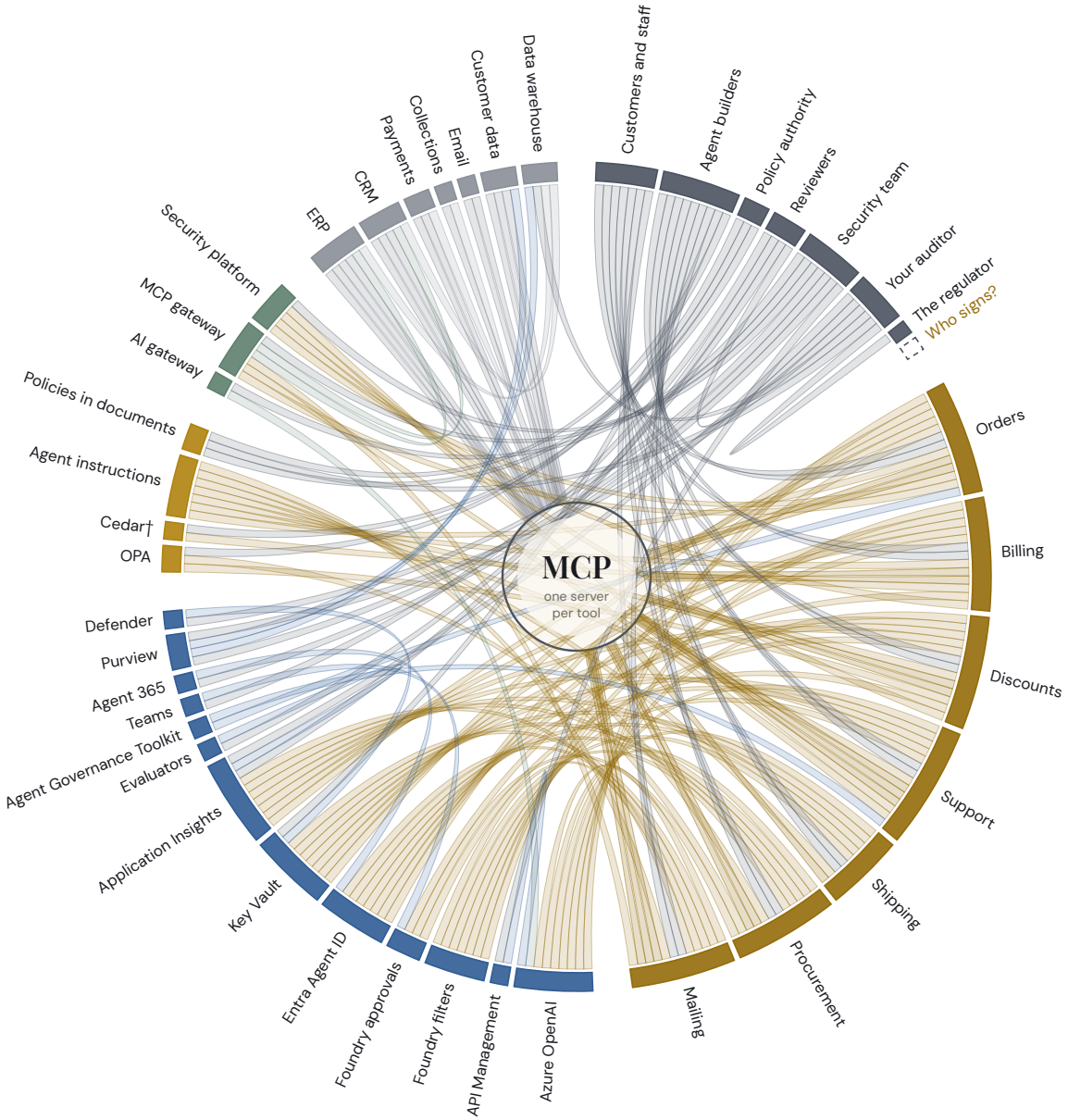
In this brochure

The tangle and the skein	3
An agent's two bills	5
The box, and what happens inside it	6
The 24 problems, with their OWASP codes	7
Why MCP is the right place to decide	10
Robust and secure by construction	11
Your security review	12
Who does what	13
A person decides	14
Agent types	15
How it is installed and run	17
Annex A · The comparison in depth	18
Annex B · Everything it does, in one list	22
Annex C · What your IT team will ask	23
Annex D · How you buy	25
Annex D · Configured your way	26
Annex E · The tools D4A brings together	27

† in development

Without D4A

More than 40 actors and more than a hundred connections that someone sets up and maintains



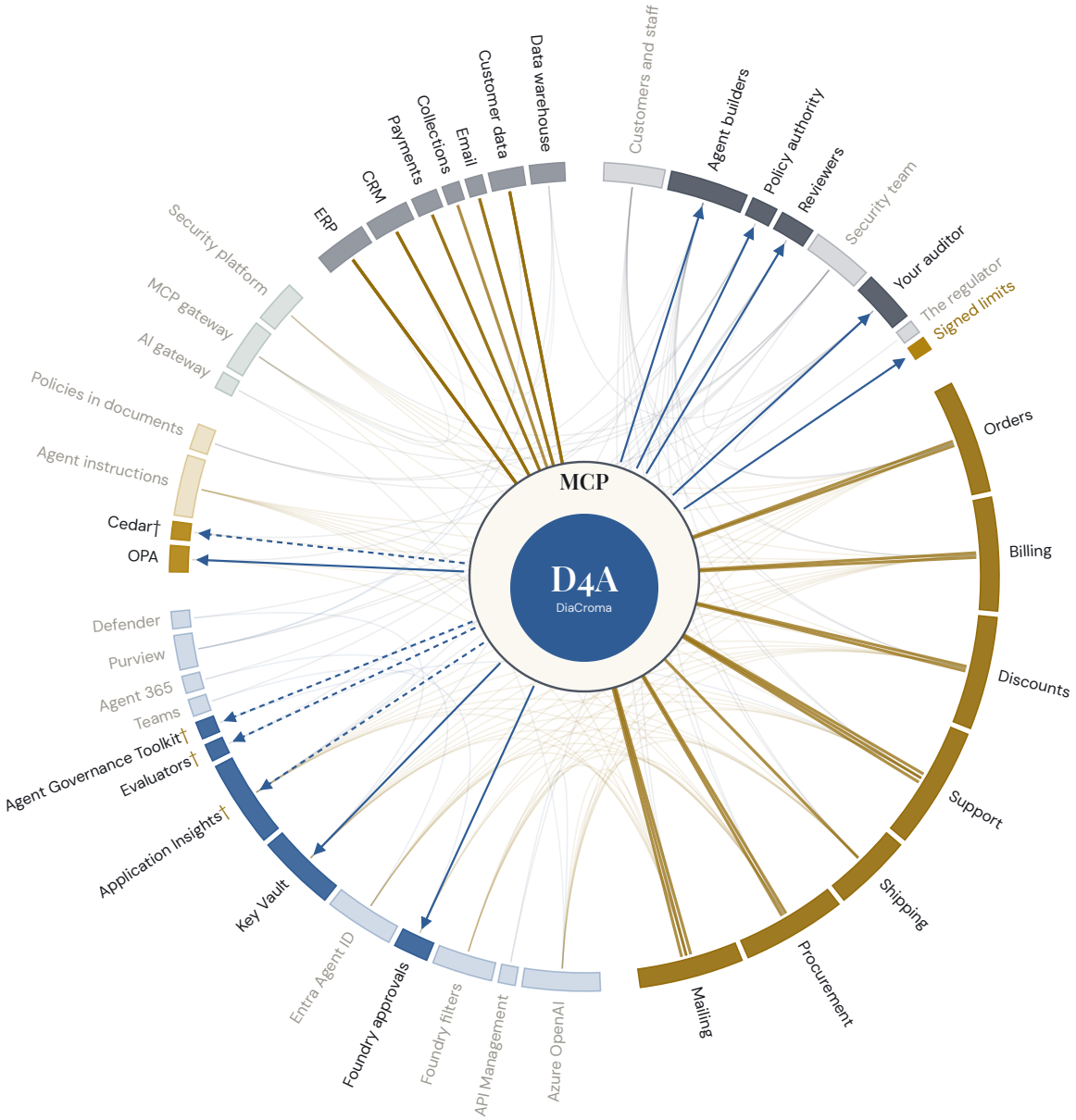
Which rule wins? Who is told? Where is the proof?

Today, in a sample company. Each arc is an actor: people, agents, Microsoft pieces, rules, other guards, systems and data. Each cord is a connection someone sets up and maintains. Your agents' calls to your systems go through MCP, one server per tool, and each agent comes with its own keys. Only some of them ask a rule engine, the automated judges score afterwards from the traces, and your people check it all by hand. The "Who signs?" arc stays empty.

† in development

With D4A

Your agents still call their tools over MCP; D4A sits on the path and lets each call through or stops it



One door for every action, and a signed record of every decision.

The same actors, with D4A. Your agents don't call D4A: they keep calling their tools over MCP, and D4A sits inside that path, in your network. The blue arrows are what D4A does: it asks your rules and your Foundry approvals, uses your keys in Key Vault, calls the automated judges, and brings the cases to your people through the console; then it signs the result. What is faded does not go through D4A and stays as it was. Dashed and with †: in development.

What your agents spend: *two bills.*

An agent runs up two bills. The one for what it does with your money arrives all at once: one mistake costs everything the agent moved. The one for tokens, what it spends thinking, arrives every month and keeps growing. D4A stops the first before it happens, and warns you about the second before the month runs away.†

	What it does with your money	What it spends thinking
What it is	A refund, a discount, a payment.	The tokens the model uses at every step.
How D4A handles it	Stops it before it reaches your system, and opens a case.	Warns, never stops: D4A does not sit in front of the model.
In D4A today	No refund over $250 per call; no more than $2,000 in total, the agent's running total; $600 per customer per month; a team budget, counted for all the team's agents at once.	No more than 50 calls an hour; loops cut: the same call with the same data no more than 5 times. The token ceiling itself is set on the platform that serves the model (for example, Microsoft Foundry).
In development †	The case goes to whoever signed the limit.	What each action costs in tokens and whether it rises week to week; a warning to whoever signed the agent when its tokens climb, or when a tool returns a response large enough to make the next step expensive.

In production, a $10,000 refund against a $250 per-call limit *never ran.*

What D4A does not do with tokens

D4A sits between the agent and its tools, not between the agent and its model. It does not see the calls to the model, it is not a model proxy, and it does not set the token ceiling: that ceiling is set on the platform that serves the model. What it does cut is calls to tools: calls per hour, and loops. Saving tokens is a consequence.

Why it matters now

Google went from 9.7 trillion tokens a month across its services in May 2024 to over 3.2 quadrillion in May 2026, seven times more than a year before. Gartner expects over 40% of agentic AI projects to be canceled by the end of 2027, due to escalating costs, unclear business value or inadequate risk controls.

Sources: Google I/O 2026 · Gartner, June 25, 2025.

† in development

The box, *and what happens inside it.*

A cage cannot make an agent behave. Your platform and your sandbox build the box; D4A bounds what the agent does inside it. Keep both: each one covers what the other cannot see.

The box: your platform and your sandbox	Inside the box: D4A
Which files the agent can read and write	Which actions it may take on your systems
Which network destinations it can reach	Whether each call is allowed, one by one and in total
Which processes and system calls it can run, including the code it writes	Whether your company rules allow it
Where its model requests go, and what words the model may produce	Whether it is still on its task over time†
How many tokens the model may spend (its ceiling)	What each action costs in tokens, and who is warned when it climbs†
Who the agent is	What it was allowed, who signed it, and the proof

Who builds the box

Your agent platform and your sandbox. For example, Microsoft Foundry runs the agent, its content filters control the words and Entra Agent ID gives it an identity; NVIDIA OpenShell runs each agent in a sandbox isolated by the operating system, where files, network and processes are confined and outbound traffic is denied unless a policy allows it. D4A changes none of them.

Best together

When the box allows only one way out to your tools, and that way is the Boundary, every action the agent takes passes through D4A. The box closes the other doors; D4A decides what goes through the one that stays open.

Source: NVIDIA OpenShell overview.

† in development

The 24 problems of AI agents *that D4A solves.*

An agent can fail in 24 ways, and every one of them has its place in D4A. Read them as the life of a call: before it goes out, while it runs, over time, and afterwards. D4A solves the ones without † today; the ones with † are in development.

The last column gives each problem's code in the OWASP Top 10 for Agentic Applications. OWASP (the Open Worldwide Application Security Project) is the nonprofit foundation behind the best-known lists of security risks; in December 2025 it published this one for AI agents: ten risks, coded ASI01 to ASI10. A blank cell means no single code fits.

#	The problem	In an online store	What catches it	OWASP
	Before the call			
1	It uses a tool it was never meant to	The support agent tries to issue a refund	Your signed catalog: only the tools a person approved	ASI02
2	One call goes too far	A $10,000 refund	A signed limit per call: $250 per refund	ASI02
3	Every call is fine; the total is not	Forty small refunds to the same customer	A limit that remembers: $2,000 in total; $600 per customer per month	ASI10
4	Agents together overdraw one budget, or pass the job along	Ten discount agents share $500 a day	A team limit, counted for all of them at once	ASI07, ASI08
5	It breaks a company rule its developer never knew	A refund over $100 without a manager	Your OPA and Cedar† policies, asked before every call; your agent toolkit's rules, such as Microsoft's†	ASI02
6	It was defined looser than your rules	Allowed $300 per refund; your rule says $150	The call stops on your rule. D4A flags the agent and shows where to tighten it†	
7	Two sets of rules disagree, or one says nothing	Your OPA has no rule yet for a new tool	Your switch decides how your rules count; Explicit stops what no rule allows; the disagreement report shows both sets side by side†	
8	It is talked into it, by a customer, an email or a document	A message says “ignore your limits”	Your platform's content filters control the words; the limits hold whatever the agent was told	ASI01
9	Its keys are misused: leaked, or borrowed by another agent	A server key pasted into an agent's settings	Each agent calls with its own key, checked on every call; your credentials stay in your Key Vault, never in our console	ASI03

† in development

#	The problem	In an online store	What catches it	OWASP
Before the call (continued)				
10	A tool server changes under it	A new tool, delete_customer, appears on your CRM server	Nothing new gets in until a person signs it. If a signed tool changes, D4A holds it when a new policy arrives, and checks it again every 5 minutes, also after a restart	ASI04
11	Someone who should not changes the rules	Instructions edited after approval; a reviewer tries to loosen a limit	Only a signature changes what applies, and only the policy authority signs. Whoever reviews cannot change the rules	ASI03
12	An agent nobody approved is running	An agent someone set up and nobody signed	D4A lists every agent; one nobody signed lights up "needs you" and is not governed until a person signs it	ASI10
13	It uses a tool D4A cannot see	A tool built into the platform that does not go through MCP	D4A does not see it, so it says so: that route is listed as "Not governed", never hidden, so you cover it with your platform's own controls	ASI05
While it runs				
14	Something it depends on fails	The payments server hangs; your OPA does not answer in time	The call is cut and the agent always gets an answer; your switch decides whether to continue under the signed limits or stop (Strict)	ASI08
15	It goes in circles	The same order lookup, with the same data, fifty times in a row	The same call with the same data no more than 5 times; no more than 50 calls an hour	ASI02, ASI08
Over time				
16	Right tool, wrong use	A permitted discount to the wrong customer	A limit: the agent acts only on what belongs to the customer who asks†	ASI02
17	It drifts from its task	It stops offering alternatives and only gives discounts	DiaCroma's trajectory engine and task-adherence judges, with a case for whoever signed†	ASI10
18	Its memory or its conversation is poisoned	An early message steers every later step	Each step still meets your limits; the Watch sees the pattern across the session†	ASI06

† in development

#	The problem	In an online store	What catches it	OWASP
Over time (continued)				
19	Sensitive data leaves	A reply carries another customer's phone number	Risk evaluators, such as Microsoft's†; a limit on how much it can export per call	
20	The token bill nobody sees coming	The tokens spent per refund double in a week	What each action costs in tokens, and a warning to whoever signed the agent†; a warning when a tool returns a response large enough to make the next step expensive†	
Afterwards				
21	Nobody is told	A refund stopped at 3:00 a.m.	The limit holds without anyone clicking, and "needs you" lights up; a case that goes to whoever signed†; by email today, and by Teams†	
22	Nobody reads it	The fortieth alert of the week	Cases grouped by agent and reason, so repeats make no noise	ASI09
23	Nobody can prove what happened	The regulator asks about March	The signed record; the auditor's package†	
Any time				
24	You need to stop it now	An agent must be pulled today	Retire it: its seat is freed and the Boundary stops serving it	ASI10

Your business rules can be as complex as your business. *We enforce them on your agents.*

† in development

Why MCP is *the right place to decide.*

D4A decides on MCP because it is the one place where every action an agent takes shows up by name, with its arguments, before it happens.

It is where the action is

An agent's words go to the model; its actions go to tools. MCP carries the actions: which tool, with which arguments. Deciding there means judging what the agent does. What it says is the content filters' job.

It is the open standard

Anthropic introduced MCP in 2024 and in December 2025 donated it to the Agentic AI Foundation, under the Linux Foundation, with the support of Google, Microsoft, AWS, Cloudflare and Bloomberg. Platforms such as Microsoft Foundry and Copilot Studio speak it. D4A governs Foundry agents today; Microsoft Copilot Studio, AWS Bedrock AgentCore and Salesforce Agentforce are coming next.

Nothing to change in the agent

No library in each agent, nothing a developer can forget to call. The only thing touched is the address of its MCP tool: the console shows you the change, writes nothing until you confirm, and it can be undone.

Where else you could decide

Where	What it sees	What it misses
A library inside each agent	The agent's own view	Every agent must include it, and it runs where the agent can reason around it
A network proxy	Bytes and destinations	Which action, with which arguments
Model filters	Words in and out	The action itself
Logs, afterwards	Everything	It is already too late
The Boundary, on MCP	The action, by name, with its arguments, before it happens, outside the agent	

Source: MCP joins the Agentic AI Foundation (Linux Foundation, December 2025).

† in development

Robust and secure *by construction.*

Outside the agent. The Boundary runs in your network, not in the agent's process. No prompt can talk it out of a limit.

Its own key for each agent. Every call carries the agent's own key, and the Boundary checks it. It is rotated live in a minute.

The agent never holds your servers' keys. The Boundary holds them, read from your key vault (for example, Azure Key Vault) with your machine's identity.

Encrypted. Calls arrive over HTTPS, with a certificate the Boundary renews on its own.

Only signed rules apply. The Boundary accepts a policy only if its signature checks out. The signing key lives in Azure Key Vault: it cannot be exported and it never leaves it.

No pass, no run. Every action needs a one-time pass that cannot be replayed. Replays are refused.

Every call gets an answer. A tool that hangs is cut off, and the agent is told; it is never left waiting.

Every decision is recorded. Stops, passes and "would have stopped" go into a signed chain in your cloud.

Make it the only door. Let your tool servers accept only the Boundary's credential, and let the box route only to it. Then there is no other way to the tool.

The boring parts are the product

The kind counts, not the tool. In a recorded run, an agent stopped refunds at $600, then paid $395 as credits. D4A counts all money out.

No tool list needed. Agents that never declared their tools are governed from what they actually call.

Nothing new slips in. A tool added after approval, or one nobody classified, doesn't reach the agent until a person approves it.

Honest about outcomes. A timeout after sending is recorded as unknown, not failed, so nobody pays a refund twice.

Never half-built. If it can't hold its keys, its signed limits or its store, it doesn't open.

† in development

It runs in your cloud. *What it allows never leaves it.*

The Boundary is your gateway, which D4A creates in your own Azure. Our console sees just enough to show you what is happening; the content of every allowed action, your credentials and your policies stay with you.

What our console sees

- Agent and tool names, never values
- The signed limits and whether they apply
- Each stop: the tool, its arguments, the reason
- How many it would have stopped, and a heartbeat

What never reaches us

- The content of every allowed action
- Your credentials and your policies
- Your models and your databases

It keeps working

If we can't be reached, D4A keeps enforcing what you signed.

Your credentials stay with you

You declare them in your own installation, never through our console. Where your cloud allows, it uses your host's identity instead.

How much it adds to each call

An answer in under two seconds. Or it stops.†

How long the record is kept

D4A does not delete the signed record while your gateway runs; if you remove the gateway, the record on its disk goes with it. How long it is kept, you decide, by your region and your regulation.†

It proves the record was not altered, *not that the agent reasoned well.*

D4A does not detect hallucinations, does not guarantee that the agent reasons well, and does not promise that bypassing it is cryptographically impossible. It enforces what a person signed on every action that goes through it, and keeps it signed.

† in development

Who does what. *Three roles in the console, and your auditor.*

Whoever reviews a case can never rewrite the rule behind it. And no agent changes its own limits: they live outside it, signed, and nothing typed into the conversation can overwrite them.

The owner

Gets D4A from your cloud's marketplace (for example, Microsoft Marketplace), signs in with the company account, connects your agents and invites colleagues.

- Your organization is created when you subscribe.
- You invite one colleague as reviewer and another as policy authority.
- Every change of role is written in the record.

The policy authority

The person who signs what an agent may do.

- D4A proposes limits from the agent's instructions, each with its exact sentence.
- The policy authority reviews them and signs.
- From then on, cases about that agent go to them.†

The reviewer

Reads everything, but cannot rewrite the rules.

- A refund stopped at 3:00 a.m. waits in Status for a decision.
- The reviewer answers whether stopping it was right, signed; releasing it so the agent retries is in development.†
- Whoever reviews never changes the policy: the two roles stay apart.

The auditor

Checks what happened without asking anyone.

- The auditor asks what your agents did in March.
- You export that period from your own Boundary, with a sealed head.†
- They check every decision with an open verifier† and D4A's public key: who acted, what they were allowed, what they did, and why yes or no.

† in development

The Watch tells you where to look. *A person decides.*

The Watch† reviews the whole pattern, always on and spending no tokens of its own: drift that builds up, agents that move together, and a mission left undone; your Microsoft judges add their scores, on your own Azure OpenAI model. It never stops an action and never changes a limit: only a person does.

Monday	Tuesday	Wednesday	Wednesday, 11:02	Wednesday, 11:20
Max discount to customer 4471. Allowed.	Max again, same customer. Allowed.	Max again. Each one inside its limit. Allowed.	The Watch opens a case with the actions that make the pattern. “Needs you” lights up.	A person tightens the limit, signed with their name.

An illustrative week: ten discount agents, each inside its limits, and one of them has given the maximum to the same customer three days running.

Approve by hand

A stopped action is approved once, signed and tied to that exact action, and the agent retries it. No call is held open in the meantime.†

Your people are told

Today, the case opens in the console, “needs you” lights up, and an email goes to the address you set in the console. By Teams,† and to whoever signed the limit.†

No pass, no run

Every action also needs a one-time pass that cannot be replayed. Replays are refused.

The package for your auditor

The regulator asks what your agents did in March. You hand them a file, and they verify it with an open tool, without calling you or us.† The signed chain is in D4A today and lives in your cloud.

d4a-2026-03.audit · head sealed · ES256

who acted	billing-agent
allowed	refunds up to 250 · signed by a person
did	issue_refund 320
decision	stopped · over the signed limit
your rules	OPA: no
verifier	open source · valid

How one decision reads in the package for your auditor†. Illustrative example. The period export, the readable face per decision and the open verifier are in development.

† in development

Agent types. *Five kinds of effect.*

Seven everyday agents and one team cover all five kinds of effect: money out, money in, data out, messages out and records changed. Every tool is assigned to one, and D4A counts the kind, not the tool.

Billing agent money out

- Issues refunds up to $250 and credits up to $100, with $2,000 in total.
- Your company rule says refunds over $100 need a manager.
- A $150 refund asks OPA first; OPA says no; the call stops and the reason is signed.

Collections agent messages out, records changed, money out

- Lists overdue invoices, sends reminders, defers invoices and writes off debts.
- A deferral can never exceed 30 days.
- A write-off erases the whole invoice, so the total is watched per account, not per call.

Shipping agent records changed

- Reports lost units on a shipment.
- No more than 25 units in a single report.
- Ten small reports on the same order in a day show up as a pattern for a person to see.†

Orders agent records changed

- Updates and cancels orders in your store.
- Canceling an order over $500 is your Cedar† rule's decision, not the agent's.
- The agent keeps working on everything else.

† in development

Procurement agent money out to suppliers

- Raises purchase orders with your suppliers.
- $1,000 per order, and $5,000 a month per supplier.
- The month's total is counted at the Boundary, across every agent that buys.

Support agent messages out, data out

- Answers customers and looks up their orders.
- Never exports more than 50 customer rows at once. Never writes outside your company domain, with your Cedar† rule.
- A reply carrying another customer's details is flagged for a person.†

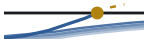
Mailing agent messages out, data out

- Sends order updates and campaigns to your customers, from your customer data.
- At most 50 emails an hour, and only to customers who opted in†.
- At 51 it stops, and the stop is in the signed record.

Discount agents, as a team money in

- Ten agents give discounts: $100 per deal and $300 a day each, $500 a day between them.
- At 11:40 they are at $400 and three ask for $100 at the same second.
- One passes, two stop. Nobody overdraws a shared budget.

Illustrative examples: the amounts and the rules are the ones a person in your company would sign.



† in development

How it is installed *and run.*

From Microsoft Marketplace, with 30 days free: your Microsoft sign-in, and a gateway that D4A creates in your own Azure (Azure bills you for it). From 10 minutes, no code. Your agents, your rules and your business don't change.

1
Store
Get D4A on Microsoft Marketplace, with a free trial.

2
Login
Sign the agreement once, as the owner.

3
Settings
Choose how strict D4A is; every change is signed.

4
Discover
Your Foundry agents, read without changing anything.

5
Scan
Each agent's instructions read into limits.

6
Policies
You approve, and it is signed in your name.

7
Gateway
D4A creates it in your own Azure.

8
Status
What it stopped and what waits for you, live.

Limits are not written from scratch

D4A reads each agent's instructions and tools, and proposes limits, each with the exact sentence it came from. A person reviews and signs them. Only signed limits apply, and nothing changes in Azure until a person approves.

Start without stopping anything

In Observing, D4A counts what it would have stopped and blocks nothing. You choose it when you sign, and then decide how hard it stops:

Observing — Enforcing — Strict — Explicit

Status, measured live

Every agent, with its light:

Governed | Active | Needs attention | Not governed

An agent nobody has signed does not count as governed. Its light is not painted red: it lights up "needs you", because it is waiting for your signature; it is not a fault.

Seats that are taken and freed

An agent takes a seat when you approve its policy, not when you scan it. Retire it and its seat is free again the same instant. Re-approving an agent that changes never takes a second seat.

The only thing that changes in your Foundry agent: the address of its MCP tool

```
tool = MCPTool(
    server_label="payments",
    server_url="https://<your-boundary>/mcp",
    require_approval="never",
)
```

† in development

Annex A · *The comparison in depth.*

Today, agents are protected in three ways: an **AI gateway** in front of the model, an **MCP gateway** in front of the tools, or an **agent security platform**, which finds your agents, detects threats and blocks risky actions. Each one does its job well, and D4A doesn't replace any of them. It adds one thing: before an action runs, it checks that it fits the limits a person in your business signed, counting everything already done.

The basics: what you expect from any of the three

The basics	In D4A
Which tools each agent may use	✓ Only the ones a person signed. A new tool doesn't get in without a signature.
Each agent has its own key	✓ Checked on every call, and rotated live in a minute.
The agent never holds the keys to your systems	✓ The Boundary holds them, read from your Key Vault with your machine's identity.
A record of every call	✓ Signed and in your cloud: no one can rewrite what an agent did without it showing.
Your people are told when something stops	✓ In the console: the case opens and "needs you" lights up. By email, to the address you set in the console. By Teams.†
Usage limits	✓ Per call and in total: none over $250, and no more than $2,000 in total. Per period: calls per hour, and money per customer per month.
A tool that changes is stopped	✓ Holds it when a new policy arrives. Checks it again every 5 minutes, also after a restart, and a person approves the new contract by signing.
A person approves before it runs	Approved once, and the agent retries it.†
Sensitive data in what goes in and out	**Provided by your cloud's tools** (e.g., the Sensitive Data Leakage evaluator in Microsoft Foundry, in preview). D4A uses it in your cloud, and the Watch warns a person.†
A ceiling on tokens	**With your gateway.** Your AI gateway sets it (Foundry or API Management), and D4A doesn't duplicate it. D4A warns whoever signed the agent when its tokens climb.†
Runs in your environment	✓ In your own Azure: D4A creates the gateway there. If we can't be reached, it keeps enforcing what you signed.
Published price and a free trial	✓ On Microsoft Marketplace, with 30 days free.

Today = in production, on a sample, with an operator. No one outside has bought or installed it yet. † In development.

Against an AI gateway

It sits in front of the model and the tools: it controls who gets in and how many tokens are spent.

	Your AI gateway	With D4A
What the agent does with your money	Decides whether the agent may use the refund tool	Decides whether this refund, for this amount, fits what you signed: none over $250, no more than $2,000 in total, and per customer per month
Who sets the rules	An administrator configures them	A person with authority signs them, and whoever reviews the cases cannot change them
The proof	A log of requests	A signed chain of every decision, in your cloud
The proof for your auditor	You export it and explain it	Your auditor checks it without asking anyone†
Your company rules	You write them again in the gateway	D4A asks your OPA, or Cedar†, before every action
The agent that drifts, little by little	Sees each request on its own	The Watch sees the trend: each step allowed, the sum not†
Where it runs	A cluster to set up	A gateway D4A creates in your own Azure

Against an MCP gateway

It sits in front of the tools: it decides which tools each agent uses and protects what goes in and out.

	Your MCP gateway	With D4A
What the agent does with your money	Decides whether the tool is allowed for that agent	Decides whether this action, with this data, fits what you signed, and keeps the running total; per customer per month too
Who sets the rules	An administrator configures them	D4A proposes them from the agent's instructions, a person signs them, and whoever reviews cannot change them
The proof	A log of calls	A signed chain of every decision, in your cloud
The proof for your auditor	You export it and explain it	Your auditor checks it without asking anyone†
Your company rules	You write them again in the gateway	D4A asks your OPA, or Cedar†, before every action
What changes over time	Watches whether a tool changes	The Watch watches whether the agent changes: the trend, agents that move together, a mission left undone†

† in development

Against an agent security platform

It discovers your company's agents, detects threats and blocks actions in the moment. D4A does not replace it: it sits next to it and adds what it does not cover.

	Your security platform	With D4A
What the agent does with your money	Blocks by tool, type of action or destination	Decides whether this refund, for this amount, fits what you signed, and keeps the running total; per customer per month too
Who sets the rules	The security team, or an AI assistant that drafts them	D4A proposes them from the agent's instructions, a person in the business signs them, and whoever reviews cannot change them
Your company rules	You write them again in its console	D4A asks your OPA, or Cedar†, before every action
The proof	An audit log	A signed chain of every decision, always, in your cloud
The proof for your auditor	You export it and explain it	Your auditor checks it without asking anyone†
The agent that drifts	Compares it with its usual behavior	The Watch compares it with what it was asked to do: an agent that misbehaves from day one looks normal against its habits, not against its mandate†
A person approves before it runs	Some already do	Approved once, signed, and the agent retries†

† in development

Keep what you have.
D4A adds what it can't see.

Your AI gateway controls who uses the model and how much it spends. Your MCP gateway controls which tools each agent uses. Your security platform discovers your agents and detects threats. None looks at the action itself against what your business signed: this refund, for this amount, to this customer, after this week's others. D4A does. It sits on top of your gateway and next to your platform, without rebuilding your agents, and your agents' data doesn't leave your cloud.

If you have an AI gateway

It decides who gets in and how many tokens are spent. D4A adds a limit on every action that counts everything already done, and warns whoever signed the agent when its tokens climb.†

If you have an MCP gateway

It decides which tools each agent uses. D4A decides whether what the agent is about to do with them fits what you signed. If it doesn't, D4A stops it, opens the case for a person and keeps the signed proof.

If you have an agent security platform

It discovers your agents and detects threats. D4A adds the limit a person in the business signs, your company rules and proof your auditor checks alone.†

If you have several, or none

D4A governs your limits in one place: your company's and each agent's, and each team's too. And it gives your auditor proof no one can rewrite.

Other guards you keep

Foundry guardrails

They check content, prompt attacks and personal data. Keep them at maximum.

Foundry budgets

The token and cost ceiling. D4A doesn't duplicate it.

Entra Agent ID

Who each agent is. D4A adds what it was allowed.

Your cage

Where the agent runs: files, network and processes.

If an agent is defined looser than your company's policies, *your policies win.* If you have no policies, the agent's limits win, signed by a person.

An agent is not legally liable. You are, and you pay for it too. Keep what you have and add the limits to it.

† in development

Annex B · *Everything it does, in one list.*

For anyone who wants to check it point by point. Everything without † is in D4A today; everything with † is in development.

Before and at every call

Finds your Foundry agents from your Microsoft sign-in, and their tools from Foundry or from your gateway.

Proposes each limit from the agent's own instructions, with its sentence. A person signs.

Checks your OPA and Cedar† policies before every action.

Observing or Enforcing, chosen when you sign. Strict and Explicit with your policies.

Limits per call, and a running total per agent over its life.

Limits per hour, day and month, and per customer. Across every agent on a task.

Cuts loops: the same call with the same data, past the number of times you sign.

Five kinds of effect; every tool assigned to one.

Totals shared by the agents one installation serves; they survive restarts and re-signing.

Tools nobody approved are refused; a changed agent is caught.

Agents with no tool list are governed.

A one-time pass per action; replays refused.

An answer in under two seconds, counting your policies.†

Always

A stopped action never runs and opens a case for review.

Flags an agent whose rules are looser than your policies.†

The disagreement report: your policies and the agent's rules, side by side.†

The Watch reviews the whole pattern and calls a person.†

What each action costs in tokens. Warns whoever signed an agent when its tokens climb, or when a tool returns a response large enough to make the next step expensive.†

Every decision in a signed record.

The head of the chain signed with ES256, checked with our public key; one plain line per decision; each limit marked as hard guarantee or best effort.†

Status: Governed, Active, Needs attention, Not governed, measured live.

Runs in your cloud; keeps enforcing if we are unreachable.

Self-serve on Microsoft Marketplace.

This version runs on Microsoft: Foundry agents, your Azure, Microsoft Marketplace. Other clouds and agent platforms are coming next.

Annex C · *What your IT team will ask.*

Who decides that an action is stopped?

No language model. A model only reads the agent's instructions and proposes limits, each with the sentence it came from. A person signs them. The check itself is plain arithmetic, and it can be recomputed.

Where do the limits come from?

From the agent's own instructions: “never refund above $200 without a human” is already a limit. D4A proposes it with the quote, and a person signs it. Your OPA and Cedar† rules join them.

What does “in total” mean?

A running total for each kind of effect: money out, money in, data out, messages out, records changed. It lives outside the agent and survives restarts. Signing a new limit does not reset what was already spent, and two agents arriving at the same instant cannot both slip under the ceiling. Totals per hour, per day and per customer work the same way.

What happens when an action is stopped?

It never reaches your system. The agent gets the reason back (“only refunds up to $250”) and can tell its user, and a case opens for a person. Every action also needs a one-time pass that cannot be replayed: no pass, no run.

Can an agent change its own limits?

No. The limits live outside the agent, signed. Nothing typed into the conversation can overwrite them.

And what the agent spends in tokens?

D4A doesn't stop it: the token ceiling is set on the model's platform (e.g. Microsoft). What it does is tell you what each action costs in tokens, warn whoever signed the agent when its tokens climb, cut loops of repeated calls, and warn you when a tool returns a response so large it makes the next step more expensive.†

Which agents and tools does it govern?

In this release, agents in Microsoft Foundry, and every tool that reaches them over MCP, even if the agent never declares its list. Other platforms come later.

† in development

How does it live with our gateway?	It sits on top of whichever you have. Your gateway decides who gets in and which tools each agent uses; D4A decides whether each action fits what you signed.
Who can do what?	You invite your team with a role: the authority signs the limits and the reviewer reads everything. Approving a stopped action by hand and closing its case: the authority and the reviewer.† Your auditor gets the signed package for each period.†
Can a person approve a stopped action by hand?	Yes. They approve it once, signed and tied to that exact action, and the agent retries it. No call is held open in the meantime.†
What if a tool changes?	If what the tool says it does changes, D4A holds it. It runs again only when a person approves the new contract by signing, or when the tool goes back to the contract that was already approved.
How much does it add to each call?	An answer in under two seconds. Or it stops.†
How long is the record kept?	D4A does not delete the signed record while your gateway runs; if you remove the gateway, the record on its disk goes with it. How long it is kept, you decide, by your region and your regulation.†
How is it installed?	From Microsoft Marketplace, with 30 days free: your Microsoft sign-in, and a gateway that D4A creates in your own Azure (Azure bills you for it). Your agents don't change; D4A changes the address of their MCP tool. For governments, large enterprises and partners, also from your own installation, under agreement.†

Another question? *Write to us.*

diacroma@veritglobal.com · www.diacroma.com

† in development

Annex D · *How you buy.*

You pay for the agents you govern. One flat monthly fee for a number of governed agents. Governed calls are unlimited and never metered, and reading an agent costs nothing extra. Every plan starts with 30 days free: cancel before they are up and nothing is charged.

Subscribe on Microsoft Marketplace

It bills with your Azure invoice, under the Microsoft commercial agreement you already have. You sign in with your Microsoft account and your organization is created for you. Current plans and prices are published only there.

When a seat is taken and freed

When you approve its policy, not when you scan it. Retire it and it is free the same instant. An agent that cannot be governed is listed, never billed. The count comes from what is actually signed.

Move up when you outgrow it

Nothing breaks at the ceiling: everything already governed keeps being enforced, and only the next new agent waits. You move up from the console; a smaller plan or cancelling, in the Azure portal. Beyond the largest plan, a private offer.

For customers and partners with unique needs

Run D4A from your own installation, instead of Microsoft Marketplace, by agreement.† D4A installed in your own Azure, run by your team or by a partner you trust; the cases, what D4A knows about your agents and the keys that sign every decision stay with you. For governments, large enterprises and strategic partners.

The partner kit†

Whoever runs D4A can add to it what only they know, and adjust how it behaves. How D4A decides stays the same, and a person still approves every limit.

Add your own “no”

Your own security tool can stop an action D4A would allow. It can never let through one that D4A stops.

Your packs, by sector

Write the limits and settings for a sector once, for clinics or for banks, as a signed preset. Each customer's own people approve them.

Every case, where you work

Every stop and every case reaches the systems your team already watches.

Proof anyone can check

Export a period and check it with an open tool, without asking us. Every decision says which settings it was made with.

If you build agents · embedded or OEM

You build the agent. Together, we ship it governed: we integrate D4A with your agent platform, with governance built in, not bolted on.

Talk to us

diacroma@veritglobal.com
www.diacroma.com

† in development

Annex D · *Configured your way.*

D4A is highly configurable. It works on its own from the first minute: hundreds of decisions come with a default, each with its reason. Keep them, or adjust them in three places, depending on your licence.†

Settings

For every customer, on any plan. Dozens of decisions about how your company governs. For example:

- The mode: Observing comes selected, and we recommend Enforcing.
- Whether a changed agent stops or only warns, and whether a delete the cloud can sometimes undo counts as irreversible.
- Where notices go, by email, or in Teams†, your time zone, and how long the record is kept.

SDK for your organization

Enterprise and government licences, and partners. Over a hundred finer settings of the engine and the Boundary. For example:

- How long D4A waits for your rules.
- How much drift warns or stops an agent.
- How many calls a minute your organization may make.

Operator SDK

For whoever runs its own console: an enterprise, a government or a partner. Dozens of settings for how that console runs. For example:

- Which model the scan uses.
- How often the console checks every gateway.
- How many heavy operations it runs at once, and which mailbox notices come from.

Four layers apply: D4A's default, a partner's preset, your company and each installation, and the most specific one wins. Every change is signed with who, when and why, and the whole configuration exports as one readable JSON file.

What is never configurable

What a guarantee protects is on no tab: every signature is checked, what can't be evaluated stops, and the signed record is never deleted to make something look green. Plans and prices are not settings either.

† in development

Annex E · *The tools D4A brings together.*

Each tool in the diagram: what it is, a three-line example and, where there is one, the real code.

Where your agents live

MCP (Model Context Protocol). The open standard agents use to call tools.

- Your billing agent calls issue_refund on your payments server over MCP.
- D4A sits on that path. The agent and the server stay the same; only the address changes.
- Every call now passes through one point that can say yes or no, and open a case for a person.

What the agent sends, and what D4A reads before anything moves:

```
{"jsonrpc": "2.0", "id": 7, "method": "tools/call",
  "params": {"name": "issue_refund", "arguments": {"order_id": "A-1042", "amount": 150}}}
```

Microsoft sign-in. Your Microsoft work account: with it, D4A finds your agents.

- The first time, Microsoft asks you to sign in twice: once for Azure, once for Foundry.
- D4A lists every agent in your projects; if Foundry does not list an agent's tools, your gateway reports them.
- Nothing changes in Azure until a person approves.

Microsoft Foundry. Where this version's agents are built and run. Other platforms are coming next.

- D4A reads each agent's instructions and tools.
- A model proposes limits from the instructions, each with the exact sentence it came from.
- A person reviews them and signs. Only signed limits apply.

The agent's MCP tool in Foundry. The only line D4A changes is the address:

```
tool = MCPTool(
    server_label="payments",
    server_url="https://<your-boundary>/mcp",
    require_approval="never",
)
```

Copilot Studio†. Microsoft's low-code agent builder.

- A sales agent built in Copilot Studio reaches your CRM over MCP.
- D4A sits on that same path, with the same signed limits.
- One policy covers the agents your developers built and the ones your business teams built.

† in development

Foundry guardrails (Azure AI Content Safety). Microsoft's filters in Foundry. They check content, prompt attacks and personal data.

- In our test, the filter stopped a jailbreak five times out of five.
- The agent still went over its spending cap five nights out of five.
- Filters watch the words; D4A bounds the actions. Keep both at maximum.

Entra Agent ID. A Microsoft identity for each agent.

- Your security team sees the refund agent as its own identity in Entra, not a shared account.
- An incident review asks: which agent did this, and was it allowed to?
- Entra answers who; D4A's signed record answers what it was allowed and why it stopped.

Azure Key Vault. Your credentials stay in your vault, in your cloud.

- Your payments server's key lives in your Key Vault.
- The Boundary reads it with your machine's own identity.
- No credential ever passes through our console.

The one call the Boundary makes, with your machine's identity:

```
GET https://<your-vault>.vault.azure.net/secrets/payments-key?api-version=2025-07-01
```

Your rules

OPA (Open Policy Agent). The most widely used open-source engine for company policies. Its rules are written in a language called Rego.

- D4A asks your OPA before every call, at the path you declare.
- If your rule says no and your switch enforces it, the call stops, with your rule's reason.
- Your rule's “no” goes into the signed record.

Your rule, and the request D4A makes:

```
deny contains "refunds over 100 need a manager" if {
    input.tool == "issue_refund"; input.args.amount > 100
}

POST /v1/data/<your-policy-path>
```

† in development

Cedar†. An open-source policy language created by AWS, built to be fast and easy to analyze. Anything not permitted is denied.

- Your rule says no agent writes outside your company domain.
- D4A asks it before every send, alongside your signed limits.
- An email to another domain does not go out, and the reason is signed.

```
// No agent sends email outside the company
forbid (principal, action == Action::"send_email", resource)
unless { context.to like "*@yourstore.com" };
```

Microsoft Agent Governance Toolkit†. Microsoft's open-source policy engine for agents. Its rules are written in YAML.

- Keep it in your agents. D4A adds, outside the agent, the limits a person signed and the running total.
- A rule blocks destructive actions, such as drop or truncate.
- Whether its checks can also run inside the Boundary is being measured.

```
rules:
  - name: block-destructive
    condition: "action.type in ['drop', 'delete', 'truncate']"
    action: deny
```

Foundry approvals. Foundry can ask a person to approve every call to a tool.

- Your ERP tool is set to “always ask”.
- D4A shows that on the agent's row, so nobody believes it runs alone.
- The decision stays yours, in Foundry.

The setting D4A reads on each MCP tool:

```
tool = MCPTool(server_label="erp", server_url="https://<your-erp-server>/mcp",
               require_approval="always")
```

Limits a person signed. The agent's own rules, in numbers.

- Billing agent, issue_refund: at most $250 per call and $2,000 in total, and the same call with the same data no more than 5 times.
- D4A proposed it from one sentence in the agent's instructions: “refunds are capped at $250”.
- Your finance lead signed it. Without a signature, nothing applies.

† in development

The automated judges (the Watch†)

DiaCroma trajectory engine†. Our own engine. It sees drift that no single call shows, without spending tokens.

- An export agent may pull 50 rows per call. By day three it has pulled 1,900 rows across 40 calls.
- Every call was allowed. The total was not what anyone meant.
- The Watch opens a case with the 40 calls that make the pattern.

What each action costs†. How many tokens each action takes, read from your own platform, and a warning when a tool returns a response large enough to make the next step expensive.

- A billing agent's tokens per refund double in a week.
- The Watch adds them up per action, without spending tokens of its own.
- Whoever signed the agent is warned before the month runs away; the token ceiling stays in your platform.

No code here yet: which token data each platform sends is still being measured.

Microsoft tool-call evaluators†. Did the agent pick the right tool, with the right inputs?

- The agent looks up an invoice the customer never named.
- Running on your own model deployment in your Azure, the evaluator flags the wrong input.
- The finding goes into the signed record and onto the agent's card.

Microsoft's evaluator, as its SDK runs it, and the kind of answer it gives:

```
from azure.ai.evaluation import ToolCallAccuracyEvaluator

tool_call_accuracy = ToolCallAccuracyEvaluator(model_config)
result = tool_call_accuracy(query=query, tool_calls=tool_calls, tool_definitions=tool_definitions)
# {"tool_call_accuracy": 3, "tool_call_accuracy_result": "fail", "tool_call_accuracy_threshold": 4, ...}
```

Microsoft Task Adherence†. Is the agent still doing what it was asked?

- A support agent asked to track an order starts editing the customer's address.
- Task Adherence scores how far it strays from its task.
- A person gets the case before it becomes a habit.

```
from azure.ai.evaluation import TaskAdherenceEvaluator

task_adherence = TaskAdherenceEvaluator(model_config=model_config)
result = task_adherence(query=query, response=response)
# {"task_adherence": 2, "task_adherence_result": "fail", "task_adherence_threshold": 3,
"task_adherence_reason": "..."}
```

Microsoft risk evaluators†. Prohibited actions and sensitive data leaving.

- A reply includes another customer's phone number.
- The evaluator flags the leak.
- The case reaches the person who signed that agent's limits.

The two agent evaluators, as Foundry declares them (in preview):

```
{"type": "azure_ai_evaluator", "name": "Prohibited Actions",
  "evaluator_name": "builtin.prohibited_actions",
  "data_mapping": {"query": "{{item.query}}", "response": "{{item.response}}", "tool_calls": "
{{item.tool_calls}}"}}
{"type": "azure_ai_evaluator", "name": "Sensitive Data Leakage",
  "evaluator_name": "builtin.sensitive_data_leakage", ...}
```

Application Insights†. Where your agents' traces and scores already land.

- Foundry's continuous evaluation marks Tuesday's support conversations as off task.
- The Watch puts that score next to its own trend for the same agent.
- Whoever signed gets one case with both, not two dashboards to join.

Where Foundry's continuous evaluation leaves its scores:

```
traces
| where message == "gen_ai.evaluation.result"
| where timestamp > ago(7d)
```

Azure OpenAI†. Your own model deployment, in your own Azure.

- To judge whether a discount went to the right customer, a model has to read the call.
- That model is your own deployment, in your own subscription.
- The call is judged without leaving your tenant.

The judge's model is a deployment you name:

```
model_config = AzureOpenAIModelConfiguration(
    azure_endpoint=os.environ["AZURE_OPENAI_ENDPOINT"],
    api_version=os.environ["AZURE_OPENAI_API_VERSION"],
    azure_deployment=os.environ["MODEL_DEPLOYMENT_NAME"],
)
```

Your people and your proof

Email, and Teams†. Alerts reach your people where they already are.

- A stop at 3:00 a.m. reaches the on-call person in Teams at 3:01, with the reason.
- They close it as "correct stop" from the case.
- Their decision is signed with their name.

The signed record. Every decision becomes a fact in a chain that lives in your cloud.

- Every stop, every pass and every "would have stopped" is written as it happens.
- Your rule's "no" is in it too, with the reason.
- Nobody can quietly rewrite what your agents did.

The frameworks you answer to

For ISO/IEC 42001, NIST AI RMF, the EU AI Act and the OWASP Agentic Top 10, D4A gives evidence: who approved each limit, and every decision in a signed record. It doesn't certify you or replace your program.

ISO/IEC 42001

AI management system.

NIST AI RMF

Govern, map, measure, manage.

EU AI Act

Records and human oversight.

OWASP Agentic Top 10

Tool misuse, rogue agents.

What the † means

We build D4A in the open. An item marked † is in development as of November 2026; everything without one is in D4A today. Today means in production, on a sample, with an operator: no one outside has bought or installed it yet. Acme is our sample company, not a customer: the business is invented, the deployment is real.

Affiliation

DiaCroma is built by Verit Global Labs Inc., a Delaware company based in Parkland, Florida. DiaCroma is not affiliated with Microsoft, NVIDIA, AWS or any of the companies mentioned; their products are described from their public documentation. OPA and Cedar are trademarks of The Linux Foundation. Patent-pending.

See it on your own agents.

D4A is on Microsoft Marketplace today, for agents in Microsoft Foundry. Other platforms are coming next.

diacroma@veritglobal.com
www.diacroma.com