REFERENCE ARCHITECTURE

Governed execution for autonomous AI.

DecisionGuard replaces standing credentials with single-use, signed execution certificates — turning every agent action into a deterministic, auditable, policy-bound execution.

VERSION	DATE	AUDIENCE	MODEL
v1.1	September 2026	Security architects · CISOs · Platform leads	AAA · DGAC · ToolHub · MCP

DecisionGuard B.V. · decision-guard.com · Patent NL-20260324 (provisional)

Table of Contents

§ 01	Executive Premise	3
§ 02	The Problem	4
§ 03	The DGAC Primitive	5
§ 04	The AAA Model	6
§ 05	Execution Bubbles	7
§ 06	MCP Protocol Integration	8
§ 07	Human Identity & Authority Binding	9
§ 08	Request Lifecycle	10
§ 09	Reference Topology	11
§ 10	DG Edge & Relay	12
§ 11	Browser Authority	13
§ 12	Adoption Modes	14
§ 13	Properties	15
§ 14	Architectural Shift	16
§ 15	Competitive Position	17
§ 16	Business Value at Agentic Speed	18
§ 17	Conclusion	20

§ 01 EXECUTIVE PREMISE

Executive Premise

Modern AI agents reason, plan, and act autonomously. Granting them direct access to production infrastructure — through static credentials, broad scopes, and after-the-fact monitoring — reproduces the failure modes of legacy access control and amplifies them at machine speed.

DecisionGuard introduces a deterministic execution model in which no agent action is possible without a cryptographic decision artifact issued by a governing authority, validated by a local enforcement point, and bound in scope, parameters, and time.

FOUNDATIONAL COMMITMENTS

- **No standing infrastructure access for agents** Every operation routes through a governed execution boundary.
- **Cryptographic gating, not advisory policy** Without a valid certificate, the tool does not run.
- **Decisions precede execution** Every action is bound to an authorization issued before it occurs.
- **Progressive adoption** Evidence-only mode delivers value on day one without forcing inline enforcement.

> **OUTCOME**
> *Safe, auditable, deterministic agentic execution — with provable answers to the question every regulator, auditor, and incident responder eventually asks.*

§ 02 THE PROBLEM

The Problem

Enterprise security evolved to manage human operators. Humans are slow, intentional, supervised, and held accountable through social and legal structures. Autonomous agents are none of these things. They reason unpredictably, operate continuously, and diverge from intent under edge cases their authors never anticipated.

ASSUMPTIONS TRADITIONAL MODELS DEPEND ON

- Static credentials grant stable, long-lived access that can be safely scoped at provisioning time.
- Over-permissioning is tolerated to avoid blocking legitimate workflows.
- Logging and monitoring enable detection and remediation after incidents.
- **Advisory enforcement** policies are observed or recommended, but not cryptographically required for execution.

> **THE FAULT LINE**
> *None of these assumptions hold for autonomous agents. When an agent can read sensitive data and write to production systems, nothing structural prevents it from using one to modify the other under reasoning drift.*

Traditional access control cannot answer the only question that matters at the moment of action: should this specific agent perform this specific action, with these specific arguments, against this specific resource, right now?

THE CONSEQUENCES

- **Over-privileged agents** granted access for what might be needed, not what is explicitly approved.
- **Unpredictable execution** that diverges from operational intent without triggering any structural control.
- **Weak traceability** connecting reasoning, decisions, and resulting actions.
- **Compounding regulatory exposure** under EU AI Act, NIS2, DORA, and sector-specific oversight regimes.

§ 03 THE DGAC PRIMITIVE

The DGAC Primitive

A DGAC (DecisionGuard Action Certificate) is a cryptographically signed, single-use execution certificate that binds a decision in scope, parameters, target, and time. It replaces standing credentials with single-use execution rights. It replaces “what an agent can do” with “what an agent has been explicitly authorized to do, here, now, exactly this way.”

DGAC · SIGNED DECISION ARTIFACT

intent	"Backup VLAN 100 config to controlled archive"
agent	claude-ops-7c1d
tool	network.docker_exec
target	toolhub://eu-west-1/network-lab
constraints	egress:false · max_bytes:65536 · dlp:strict-pii
expires_at	issued_at + 120s (single use)
signature	ed25519:0x9f3a…c7e1

ANATOMY OF A CERTIFICATE

- **Intent** a human-readable statement of what the agent is attempting. The semantic anchor for audit.
- **Tool** the exact capability being invoked. Identifier-bound, no aliasing.
- **Arguments** frozen at issuance. No substitution is permitted between authorization and execution.
- **Target** the resource identity. A certificate issued for one bubble cannot be replayed against another.
- **Constraints** predicates that must hold during execution: egress controls, byte limits, DLP profiles, time windows.
- **Validity window** short-lived TTL measured in minutes or seconds. Expiry is enforced; replay is impossible.
- **Consumption state** a DGAC can be active, held, revoked, or consumed. Execution requires live valid state.
- **Signature** Ed25519 signature over the canonical certificate body, verified at the enforcement point.

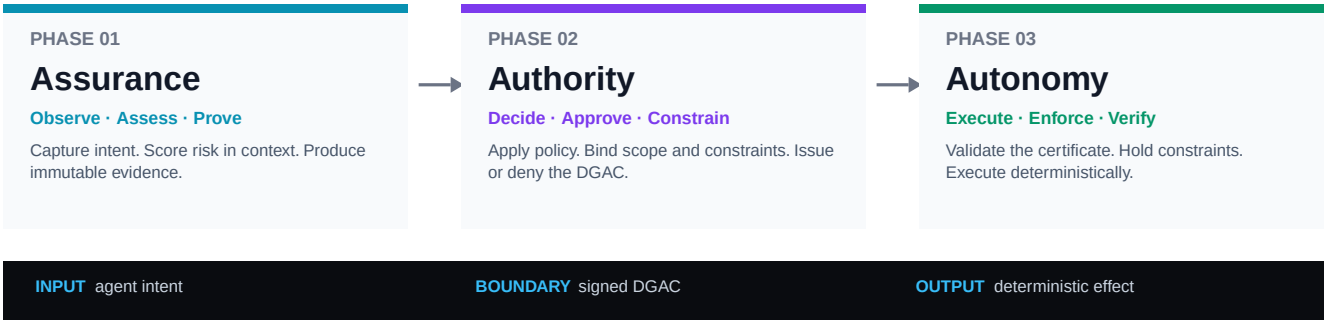
> **EXECUTION RULE**
> *Execution requires a valid DGAC. No DGAC, no execution. This is enforced cryptographically at the tool boundary — not by an advisory layer that can be reasoned around, ignored, or bypassed.*

A DGAC is not a session, a role, or a bearer permission. It is a decision artifact. It proves that a specific action was explicitly authorized at a specific moment, by a specific authority, under specific constraints, and bound to a specific reasoning trace.

§ 04 THE AAA MODEL

The AAA Model

DecisionGuard governance operates across three distinct phases. Each phase has a single responsibility, runs in a different trust domain, and produces an output the next phase depends on. Together they form the AAA Model.



PHASE 01 — ASSURANCE

Captures the agent's proposed action, the surrounding reasoning trace, and the operating context. Risk is assessed against historical baselines, behavioral signatures, and the structural sensitivity of the requested operation. The output is verifiable evidence — independent of any later decision.

PHASE 02 — AUTHORITY

Evaluates the request against policy. Applies fine-grained constraints — argument bounds, target restrictions, time windows, and approval requirements. Issues a DGAC only when every condition is met. Authority is the only phase that can produce certificates; it can also produce DENY, REQUIRE_APPROVAL, or ESCALATE verdicts.

PHASE 03 — AUTONOMY

The local enforcement point — the ToolHub — validates the certificate cryptographically, checks its state live, holds its constraints during execution, and runs the tool. A mandatory enforcement wrapper ensures that no tool can execute without first presenting a valid, unexpired, unmodified DGAC. The earlier vulnerability — an agent receiving a verdict and ignoring it — is structurally closed.

> **SEPARATION OF TRUST**
> *Assurance can be compromised without producing false ALLOWs. Authority can be slow without producing unsafe executions. Autonomy can be hostile and still cannot synthesize a valid certificate. Each phase fails closed.*

§ 05 EXECUTION BOUNDARY

Execution Bubbles

Each ToolHub instance creates a governed execution bubble inside the customer environment — an isolated, cryptographically enforced zone that owns local infrastructure and refuses to execute anything without a valid DGAC. Agents remain outside the bubble and never receive direct infrastructure access.

ISOLATION · LOCAL CONNECTIVITY · CRYPTOGRAPHIC GATING · ZERO LATERAL MOVEMENT

PROPERTIES OF EVERY BUBBLE

- **Local infrastructure connectivity** direct access to databases, APIs, and cloud control planes inside the bubble; latency stays low because the gate is co-located with the resources it protects.
- **Isolated execution boundary** no cross-bubble interference; a compromise in one bubble cannot pivot through DecisionGuard to another.
- **Zero direct agent access** agents cannot reach infrastructure under any path; the only ingress is the DGAC gate.
- **Mandatory enforcement wrapper** every tool invocation passes through a wrapper that refuses to run without a valid certificate. Verdicts cannot be silently ignored by the agent runtime.

MCP Protocol Integration

Agents increasingly reach tools through the Model Context Protocol (MCP) — a standardized handshake for tool discovery and invocation used by Claude Desktop, Claude Code, and a growing set of custom agent frameworks. DecisionGuard implements a governed MCP surface so any MCP-compliant client can call tools through the same DGAC-gated boundary, without a bespoke SDK integration per agent.

The gateway negotiates protocol version, exposes a catalog-driven tool list, and transparently proxies both first-party ToolHub tools and upstream third-party MCP servers — while every tool call is still evaluated against the AAA model before it is allowed to execute. From the agent's point of view, it is simply talking MCP. From the architecture's point of view, nothing invoked over that surface bypasses the DGAC gate.

WHAT THE GATEWAY PROVIDES

- **Standards-based handshake** protocol version negotiation, streamable HTTP and SSE transports, session-scoped identity.
- **Catalog-driven discovery** tools are resolved from the governed catalog, not hardcoded per client.
- **Per-tool replay policy** ONE_TIME certificates for state-changing actions; MULTI_USE for polling and status tools within a job.
- **Upstream MCP proxying** third-party MCP servers are governed the same way as native ToolHub tools.

> **ANY MCP CLIENT, ONE BOUNDARY**
> *An agent built on any MCP-compliant framework inherits full DGAC governance the moment it points at the DecisionGuard gateway — no custom enforcement code required in the agent itself.*

Human Identity & Authority Binding

A decision is only as trustworthy as the identity behind it. Beyond authorizing what an agent may do, DecisionGuard binds every certificate to who — or what — is accountable for the decision: a verified human, a human-delegated mandate, or a service identity operating with no human in the loop.

Identity is resolved from verified sources — SSO/IdP sync (Okta, Entra, Google), on-behalf-of headers validated server-side, and device-authorization binding for headless CLIs and daemons — never from caller-supplied claims alone. A caller cannot assert its own accountability; the platform resolves it.

AUTHORITY MODES PINNED ON EVERY CERTIFICATE

human_direct	Verified human, interactive session (SSO/IdP)
human_delegated	Human-issued mandate to an autonomous agent
service_account	Machine identity, no human in the loop
legacy_unknown	Pre-DGAC callers, flagged for migration

> **HEADLESS BINDING**
> *A device-authorization flow binds a human operator to a browser-less CLI or daemon before it can act — the same non-repudiation guarantee extends to infrastructure that never opens a browser tab.*

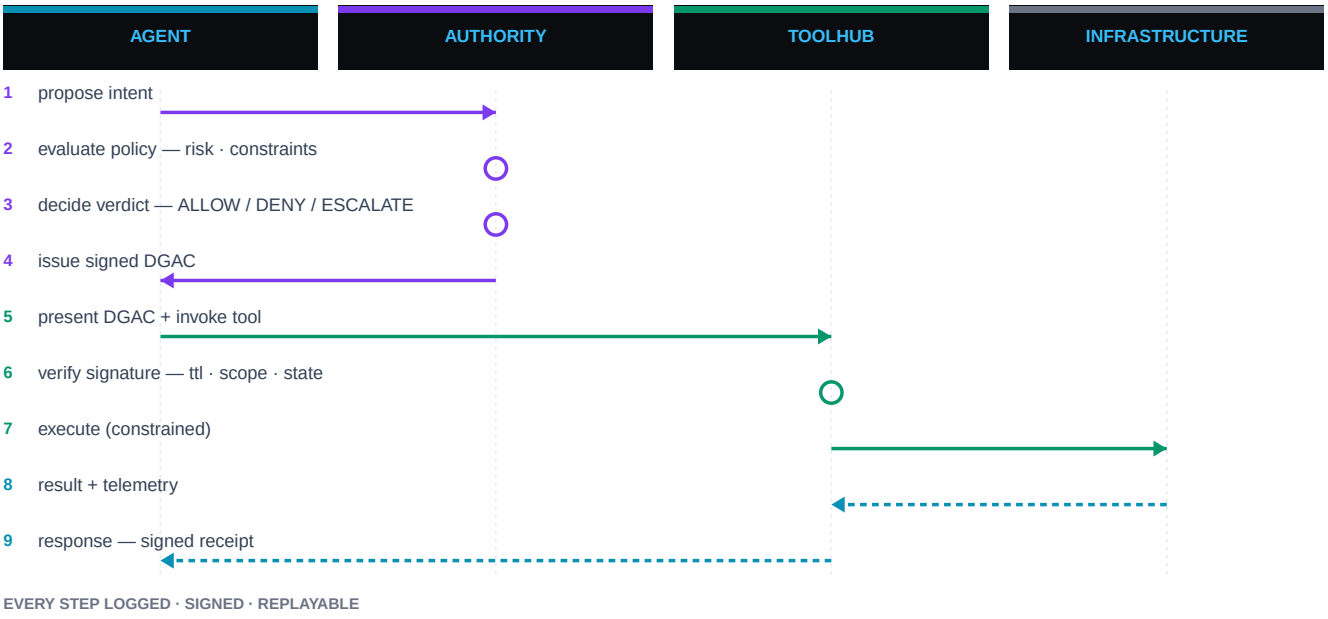
Because authority mode is resolved and signed at issuance — not inferred after the fact — every audit event and every evidence bundle can answer who approved this, and under what authority, without reconstructing intent from logs.

§ 08 FROM INTENT TO EFFECT, IN NINE STEPS.

Request Lifecycle

The full request lifecycle is the operational unit of DecisionGuard. Every governed action follows the same path — across the agent, the Authority, the ToolHub, and the infrastructure it protects — and every step is logged, signed, and replayable.

8.1 — SEQUENCE



Steps 1–4 resolve authorization: the agent proposes intent, Authority evaluates policy and risk, decides a verdict, and — only on ALLOW — issues a signed DGAC. Steps 5–7 are execution: the agent presents the certificate to the ToolHub, which verifies it cryptographically before running the tool under its bound constraints. Steps 8–9 close the loop: the result and telemetry flow back through the ToolHub, and a signed receipt returns to the agent.

8.2 — FAILURE PATHS

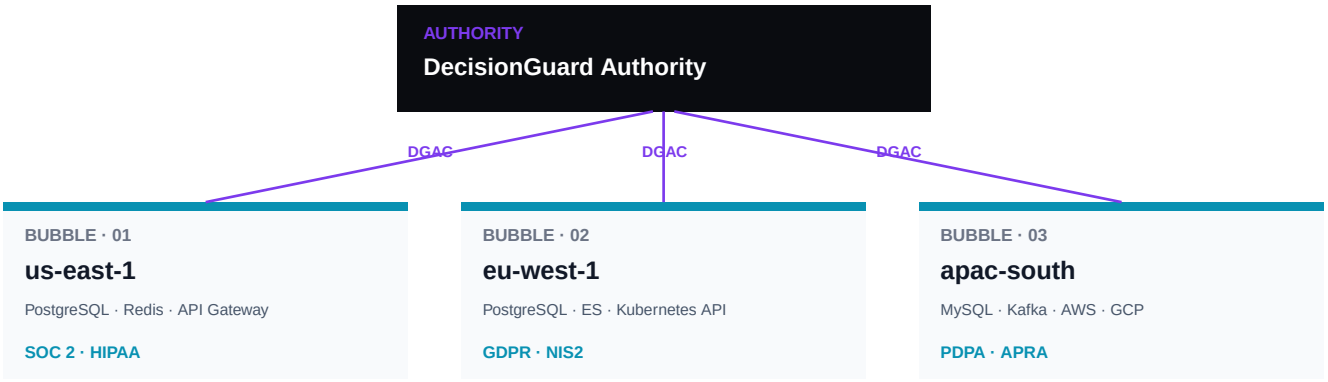
Verification at step 6 is a hard gate, not a warning. Any of the following causes the wrapper to short-circuit before the tool ever runs — and the denial itself becomes a signed audit event linked to the original request:

- **Expired TTL** the certificate's validity window has lapsed; short-lived TTLs make replay outside the window impossible.
- **Modified arguments** the arguments presented at execution no longer match what was frozen into the certificate at issuance.
- **Wrong target** the certificate was issued for a different resource or bubble than the one being invoked.
- **Revoked state** the DGAC's live consumption state is no longer active — it has been revoked, held, or already consumed.
- **Invalid signature** the Ed25519 signature over the canonical certificate body fails verification at the enforcement point.

§ 09 REFERENCE TOPOLOGY

Reference Topology

Policy is centralized; execution is distributed. A single DecisionGuard authority issues certificates; execution happens at any number of regional ToolHubs, each owning a slice of infrastructure with its own residency, latency, and compliance profile.



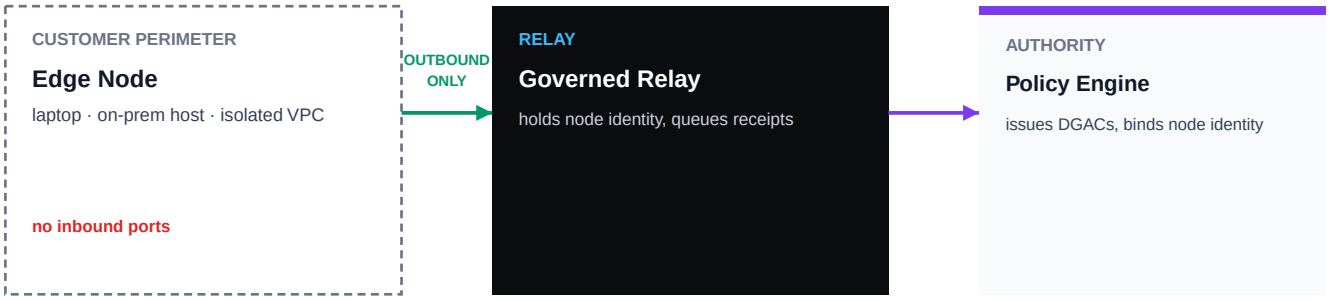
WHAT THIS TOPOLOGY ENABLES

- **Geographic isolation** separate bubbles per region for GDPR, NIS2, and data residency requirements that prohibit cross-border data movement.
- **Infrastructure isolation** each bubble owns its local services; failures and breaches do not propagate.
- **Unified governance** a single authority controls every bubble through identical policy semantics.
- **Horizontal scaling** new bubbles join the topology by registering with the authority; the policy engine never changes shape.

DG Edge & Relay

Not every environment can host a full ToolHub. DG Edge is DecisionGuard's on-premises and endpoint governance agent for Windows, macOS, and Linux. Each enrolled instance is an Edge Node that relays governed local MCP and agent execution over one outbound-only channel, so no inbound firewall rule is ever required.

An Edge Node verifies signed authority artifacts locally, carries a device-attested bound identity, and executes only calls accompanied by a valid DGAC relayed from the central policy engine. If the relay connection drops, it fails closed. DG Edge also builds a privacy-scrubbed local AI asset inventory through process monitoring, eBPF/ETW kernel sensors, disk and credential-store footprint discovery, direct AI API/SDK-key detection, DNS hostname observation, and network inventory; it reconciles these findings with external SSE/AI-discovery feeds such as Zscaler into a unified AI Asset Graph.



WHAT EVERY EDGE NODE GUARANTEES

- **Outbound-only relay** no inbound ports; the Edge Node initiates and holds its single channel to the relay.
- **Bound node identity** device-attested at enrollment and re-verified so a relay key cannot be replayed against a different Edge Node.
- **Local artifact verification** signed authority artifacts are checked on the Edge Node before governed local MCP/agent execution.
- **Fail-closed on disconnect** an Edge Node that loses its relay connection cannot execute; it queues or refuses, never falls back to ungoverned local execution.

> **ZERO INBOUND SURFACE**
> *A compromised network path to an Edge Node cannot be used to push commands in — the node only ever calls out, and only ever acts on signed authority artifacts it fetched itself.*

Browser Authority

Browser Authority is DecisionGuard's governed browser-automation execution surface: a signed extension for Chrome and Edge MV3, with Firefox and Safari companions. Device attestation binds the browser session to an enrolled device, while signed leases constrain authority to a specific tab, origin, and browser.

DEVICE ATTESTATION · CONSEQUENTIAL ACTIONS REQUIRE APPROVAL · BEFORE/AFTER SCREENSHOT EVIDENCE

Every browser action is semantically classified, with prompt-injection and context-drift detection evaluated before execution. Consequential actions are held for approval; continuous recording and before/after screenshot evidence bind the authorized action to its observed result for audit and replay.

BROWSER-SCOPED AUTHORITY AND EVIDENCE

- **Signed, narrow leases** authority is bounded to the approved tab, origin, and browser rather than becoming a standing browser-wide permission.
- **Semantic action classification** the action and surrounding page context are checked for prompt injection and context drift before it is allowed.
- **Consequential-action approval** high-impact browser operations pause for human approval before execution.
- **Continuous evidence** recording and before/after screenshots preserve what the agent saw and what changed.

Adoption Modes

DecisionGuard does not require an enterprise to bet its operations on day one. The same architecture supports three enforcement postures, and tenants progress through them as confidence accumulates.

01	**Evidence**	ToolHub observes every request and produces signed audit evidence. Nothing is blocked. The architecture proves itself through visibility before it earns the right to enforce.	OBSERVE
02	**Conditional Authority**	Low-risk actions auto-approved. Medium-risk actions require human approval. High-risk actions are blocked. Enforcement scales as policy maturity grows.	GRADUATED
03	**Full Enforcement**	Every action requires a valid DGAC. No exceptions. Policy is mandatory. Execution is deterministic. The mandatory enforcement wrapper makes execution impossible without authorization.	ENFORCED

PROGRESSIVE ADOPTION

Start in evidence. Move to conditional authority once policies are tuned. Mature into full enforcement when the organization is ready to live in a deterministic operating model. Each transition is a configuration change, not a re-architecture.

§ 13 PROPERTIES

Properties

The properties below are not features bolted onto a runtime — they are direct consequences of the architecture. They hold whether DecisionGuard is deployed for a single agent or for an enterprise fleet. Governance events also compound into a continuous compliance evidence layer: audits, corrective actions, and management reviews are fed by the same signed decision trail, not assembled as a point-in-time snapshot.

ID	PROPERTY	DESCRIPTION	SUPPORTS
P-01	**Deterministic execution**	No agent can deviate from authorized actions. Every execution is cryptographically bound to a specific decision.	EU AI ACT · ART 14
P-02	**Non-repudiation**	Complete signed trail of who authorized what, when, why, and under which constraints. No plausible deniability.	SOC 2 · CC6.1
P-03	**Full auditability**	Every action linked to a decision; every decision linked to a reasoning trace. Complete chain of custody.	NIST AI RMF · MEASURE
P-04	**Least privilege by design**	Agents receive exactly what they need, exactly when they need it. No standing permissions. Every grant is time-bound.	ZERO TRUST · NIST 800-207
P-05	**Forensic replayability**	Any incident can be reconstructed deterministically from the signed event log and the constellation graph.	DORA · ART 17
P-06	**Data residency honored**	Bubbles are geographic; data and execution stay in their jurisdiction by structure, not by promise.	GDPR · CCPA
P-07	**Continuous compliance evidence**	Every governed decision feeds a living evidence layer — audits, corrective actions, and management reviews — instead of a point-in-time report.	ISO 27001 · SOC 2 TYPE II

§ 14 ARCHITECTURAL SHIFT

From access to authorization. From trust to proof.

THE OLD QUESTION

“Who has access to this resource?”

!’

THE DECISIONGUARD QUESTION

“Was this exact action explicitly authorized — right now, in this exact way?”

That shift — from access-based to decision-based execution — enables safe autonomy without sacrificing control. Agents are free to reason. Execution is bound by cryptographic proof. This is not a control layer. It is a new execution primitive for AI.

§ 15 COMPETITIVE POSITION

A governance plane, not a guardrail.

Most of the market answers one question: is this prompt or output safe? Hyperscaler AI platforms answer it with content filters and identity controls scoped to their own cloud. Specialist guardrail vendors answer it with prompt- and output-layer scanning that works across models but stops at detection. DecisionGuard answers a different question: was this exact action explicitly authorized, right now, in this exact way — and can that be cryptographically proven afterward? That is an execution-layer primitive, not a bigger filter.

CAPABILITY	DecisionGuard	Claude Enterprise	MS AI Foundry	AWS Bedrock	Vertex AI	Guardrail Platforms
Cryptographic, signed certificate required before every tool call executes	✓	–	–	–	–	–
Vendor- and model-agnostic — governs any agent, not just its own	✓	–	±	±	±	✓
Deterministic block at execution, not detect-and-flag after the fact	✓	±	±	±	±	–
Runs across multi-cloud, on-prem, and edge execution hubs	✓	–	–	–	–	±
Full request-to-effect decision graph, signed and independently replayable	✓	±	–	–	–	–
Human-in-the-loop approvals that produce a binding, re-presentable artifact	✓	–	±	±	±	–
Progressive adoption (observe !' conditional !' enforced) without re-architecture	✓	–	–	–	–	–
Native governance of MCP / tool-calling protocols at the dispatch layer	✓	±	±	±	±	±
Compliance evidence (SOC 2, ISO 27001, GDPR) generated continuously from live decisions	✓	±	±	±	±	±
One governance plane across every agent vendor in the org, simultaneously	✓	–	–	–	–	✓

' Native, first-class ± Partial / adjacent – Not offered

Comparison reflects each vendor's public documentation as of August 2026. Hyperscaler platforms continue to expand identity, RBAC, and content-safety controls within their own clouds — real progress, but it governs content and access, scoped to one vendor's models and infrastructure. Specialist guardrail platforms (Lakera, CalypsoAI, Credo AI and similar) are the closest architectural cousins: vendor-agnostic by design, but operating as a prompt/output inspection and risk-inventory layer rather than a cryptographic, pre-execution authorization primitive bound to the tool call itself. DecisionGuard's differentiation is structural: authorization happens once, deterministically, before execution — independent of which model, cloud, or vendor initiated the request.

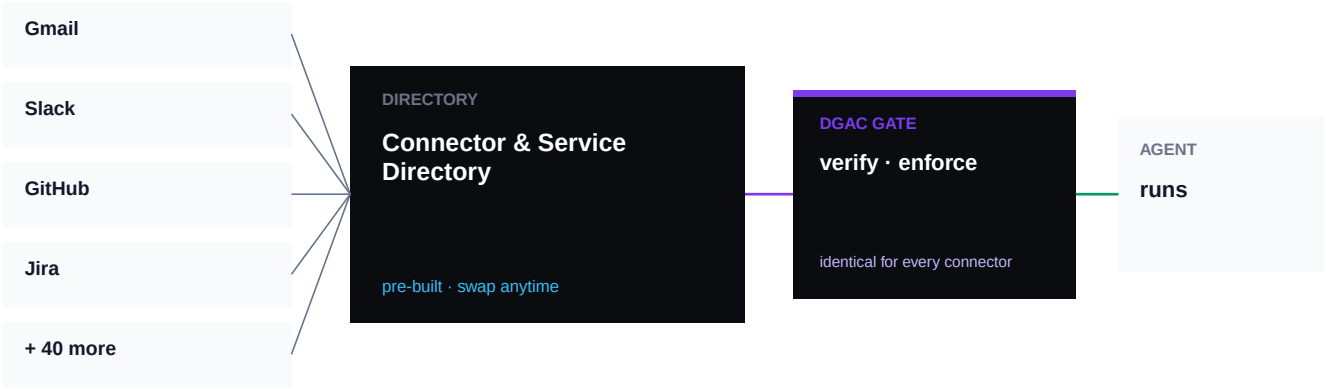
§ 16 BUSINESS VALUE AT AGENTIC SPEED

Velocity is the point, not the tradeoff.

Governance is only half the story, and framing DecisionGuard purely as a control layer undersells what it enables. Every enterprise agent program eventually hits the same wall: agents need dozens of tools, and each new integration has historically meant a bespoke credential, a one-off security review, and weeks of engineering before an agent could touch it. The same platform that authorizes every action in this document also ships a governed Connector and Service Directory — pre-built MCP and REST/API connectors that a team turns on in minutes, not sprints, without giving up any of the authorization guarantees described in the sections above.

16.1 — THE CONNECTOR ECONOMY

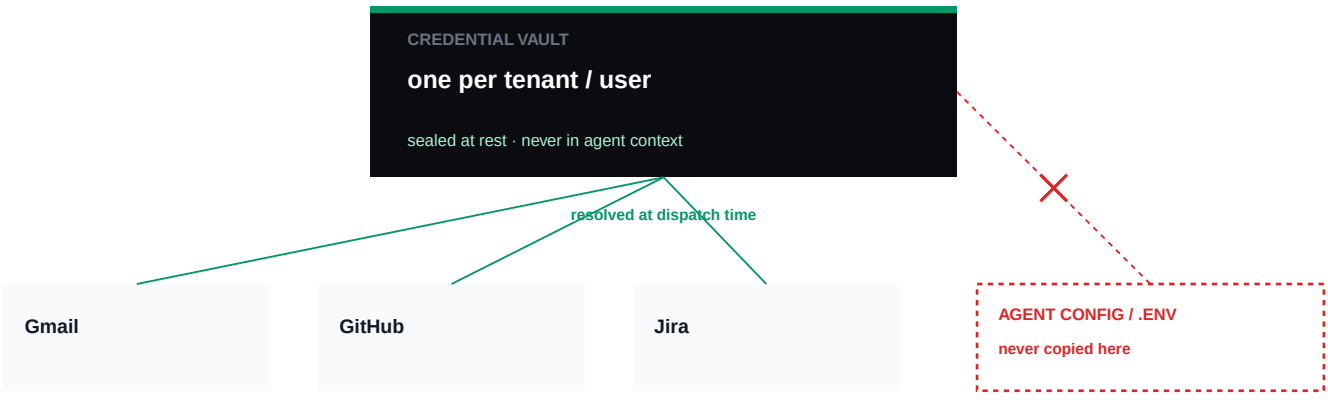
Connectors in the Directory (Gmail, Drive, GitHub, Slack, Jira, and dozens more) are interchangeable by design. Swap Gmail for Outlook, Slack for Teams, or a demo CRM for the production one, and nothing about how an agent is authorized changes — the DGAC gate, the AAA phases, and the audit trail sit underneath every connector identically. Teams are never locked into today's stack; they are free to add, replace, or retire tools as needs change, because governance is bound to the action, not to the vendor that happens to ship it.



SWAP ANY CONNECTOR — THE GOVERNANCE LAYER NEVER CHANGES

16.2 — ONE VAULT, ZERO SPRAWL

That same catalog is where the credential-sprawl problem gets solved, not relocated. Every connection is stored once, per tenant or per user, inside a single governed vault — never copied into agent configs, environment files, CI secrets, or a scattered secrets manager per project. Personal and company-scoped connections stay distinct, and every tool resolves the right credential automatically at dispatch time — at the moment a call is actually authorized to run, not before, and never handed to the agent directly.



ONE VAULT · PERSONAL & COMPANY-SCOPED · NEVER DUPLICATED PER PROJECT

16.3 — GOVERNED AT EVERY HOP

None of that speed comes at the expense of the model this document has built up to this point. A connector call dispatched through the Directory passes through the identical authorization and audit chain as a terminal command or Browser Authority action — the same DGAC gate, the same decision graph, the same replayable evidence. Teams get the speed of a plug-and-play integration marketplace and the assurance of a single governed execution path, because in this architecture the two were never a tradeoff — they are the same primitive, applied twice.

§ 17 CONCLUSION

Autonomy doesn't need a bigger guardrail. It needs a new primitive.

Filtering prompts and scoping access were sufficient when a human clicked every button. They are not sufficient when an agent can plan, delegate, and act across dozens of tools without a human in the loop for each step. DecisionGuard was built for that world: every action — terminal command, Browser Authority action, API call, MCP tool invocation — is authorized once, cryptographically, before it executes, and the resulting decision graph stands as evidence long after the action is done.

That is the architecture this document has described: the DGAC primitive, the AAA model, execution bubbles, identity and authority binding, and a topology that runs the same way in the cloud, on-prem, or at the edge. It is not a competitor to the platforms your teams already build on — it is the authorization layer that makes autonomous execution on top of them provable, auditable, and safe to scale.

decision-guard.com · mathijs@decision-guard.com