TECHNICAL REFERENCE ARCHITECTURE

Choosing the right controls for AI agents

VeriProof SDK, Agent Gateway, or both

Document ID VP-ARCH-2026-03 | November 2026 | Version 1.6

For enterprise architects, engineering leaders, and governance teams

Executive summary

A capable agent gateway does not remove the need for application-level control. The right integration depends on where an action can still be stopped and which facts the policy needs at that point.

The VeriProof SDK places policy checks inside an application, around the calls and business actions the team chooses to protect, and supplies business facts as typed industry values. The Agent Gateway checks supported model, Model Context Protocol (MCP), and agent-to-agent (A2A) traffic on approved routes. When a business action reaches a supported tool route with the context policy needs, the gateway can govern that action as well.

Use both when an application needs traffic controls on its routes and business controls inside its own code. For example, the gateway protects model access and tool traffic while the SDK checks an internal purchase order or account change. VeriProof coordinates the two automatically. An action that both paths see is checked once and escalated once, and each path keeps the checks that only it can perform.

Contents

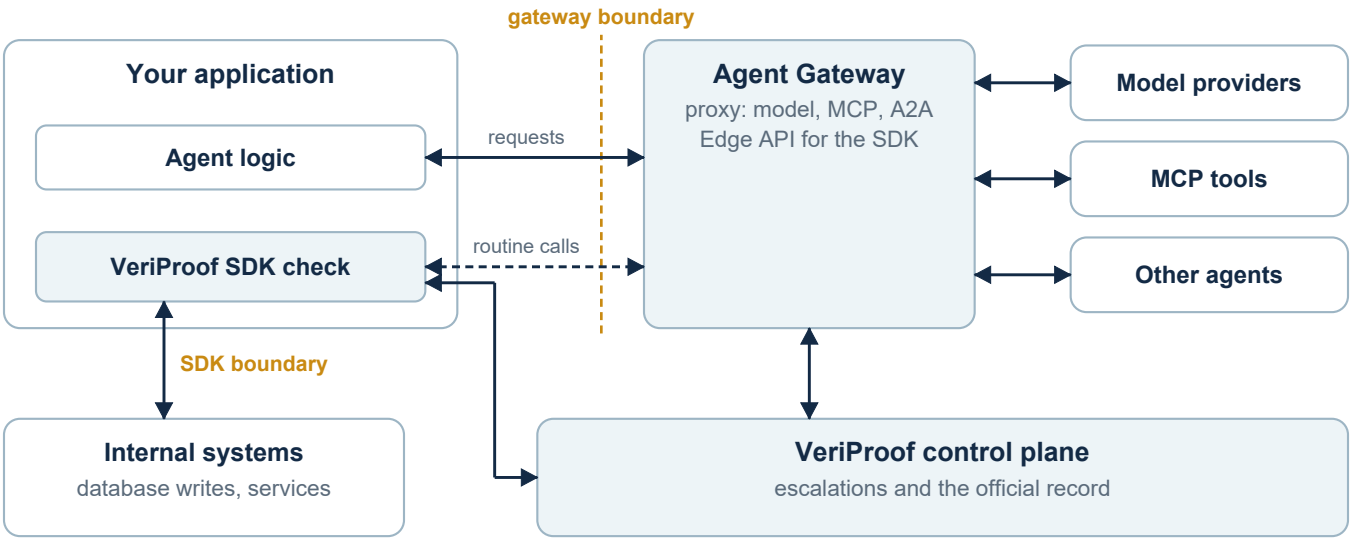
01 Start with the enforcement boundary 2
02 Where the SDK adds value 3
03 Where the gateway adds value 5
04 Coverage and integration choice 6
05 Using both in the same application 7
06 A practical rollout 8
07 Recommendation and references 9

1. Start with the enforcement boundary

Every agent action you govern needs a point where VeriProof can check it and still stop it. That point is the enforcement boundary. Where you place it decides what VeriProof can see and stop. [1, 2]

VeriProof can place the boundary in two locations:

- **Inside the application, with the SDK.** The SDK wraps the operations you choose, such as a database write or a call to an internal service. It sees the business facts the application holds and can stop the action before it runs.
- **On the network route, with the Agent Gateway.** The gateway sits between your agents and the model providers, MCP tools, and other agents they call. It checks each request and response as it passes, with no VeriProof code in the agent.



Both report to the VeriProof control plane, which handles escalations to people and keeps the official record of every decision. [1, 3]

The SDK can also send its routine calls, such as policy checks and the evidence it records, to the gateway's Edge API instead of VeriProof's central service, shown as the dashed line. This keeps that traffic inside your own network and reduces latency, which suits applications that need quick decisions.

What each one sees and stops

- **The SDK** sees the facts your application holds, such as an account's approved limit or a case's current stage, along with the operations you wrap. It stops an operation before it changes a business system.
- **The Agent Gateway** sees each model, MCP, and agent-to-agent request on its routes, and the response that comes back. It stops a request before it reaches its destination and can hold a response until it has been checked.

Start with the boundary closest to your most important action, then add the other when you need what it sees. The rest of this paper compares the two placements and shows when to use one, the other, or both.

2. Where the SDK adds value

The SDK earns its place when the application knows something important that a proxy request does not contain, or when the action happens outside the gateway's routes.

An application may know that a supplier's bank details changed yesterday, a customer has already received a credit, or a case has reached a particular workflow stage. The SDK lets the application supply selected facts to policy and protect the function that changes the business system. [2, 4]

Business facts in the industry's own terms

VeriProof provides 14 industry lexicons and a shared governance vocabulary, with 476 typed fields built on standards such as ACORD, HL7 FHIR, and ISO 20022. Each SDK exposes them as typed values, so an insurance application reports `ClaimAdjudicationOutcome = approved_partial_payment` instead of a label it invented. [7]

Rules test that field directly instead of searching a prompt, and the recorded decision keeps the field and its value. Every value carries a plain-language label and its source standard. Each field lists the regulations it relates to, which the portal uses to suggest compliance and regulatory frameworks for the team to apply. The gateway runs the same rules, but it can use a typed field only when the value reaches it, for example in a tool request's arguments or in context the SDK links to the operation.

What the SDK adds beyond a policy check

- **Business context at the action.** Supply account, case, transaction, and workflow facts without inferring them from a prompt. VeriProof holds a new rule back until the application sends the fields it needs.
- **Coverage of internal operations.** Protect direct database writes, custom service calls, and application functions that do not use a supported proxy protocol.
- **A clear result for every action.** Each call returns whether the action completed, was blocked, was escalated, or timed out, and whether a write may already have happened, so a retry never repeats an uncertain side effect. The gateway returns HTTP status codes and records what it forwarded, but the application has to look up whether a timed-out request went through. [6]
- **Escalations the application can resume.** When a person needs to decide, the SDK hands the application a checkpoint to save, verifies the decision when it arrives, and resumes the workflow once with any permitted edits. [3, 5]
- **Sensitive data masked in the application.** Mask card numbers, account numbers, contact details, and your own patterns before policy runs and before anything is recorded.
- **The evidence behind a decision.** Attach the invoice, report, or screenshot an action relied on, and capture its input and output for the people who check it later.
- **Tested before it goes live.** Run allowed, denied, escalated, and failed decisions through the SDK test harness, with no account or network needed. [2]

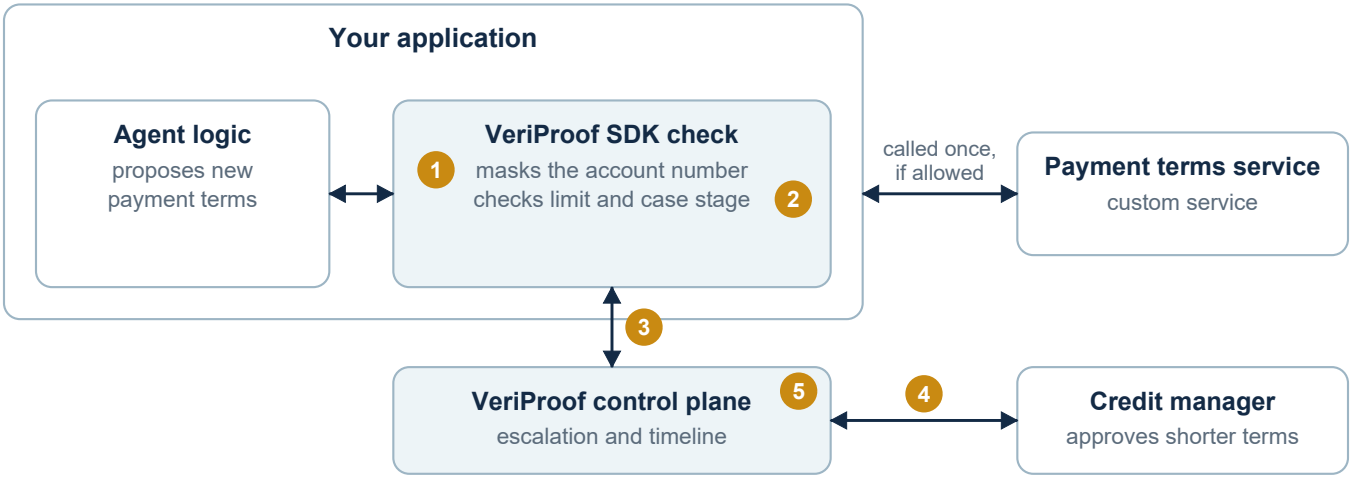
What it takes

Adding the SDK is quick. One `govern()` call wraps each operation you protect and passes the facts its rules need:

```
result = await vp.govern("claim.pay", claim, pay_claim, facts=facts)
```

The team decides which operations to protect, keeps those calls current as the application changes, makes sure no other code path can reach the business system without the check, and provides durable storage wherever a pause can outlast the process. Adapters for frameworks such as LangGraph check the model calls, tool calls, and tasks that framework runs, and a `govern()` call covers anything else the agent does, such as a direct write to an internal system.

Illustrative use case: an internal account update. An agent proposes new payment terms for a customer through a custom service, a call no gateway route carries. Here is how the SDK handles it:



1. **It masks the account number** before anything is checked or recorded.

2. **It checks the change** against the account's approved limit and current case stage, facts only the application holds.

3. **The change exceeds the limit,** so the SDK returns an escalation and the application saves a checkpoint.

4. **A credit manager approves shorter terms.** The application applies that edit and calls the service once.

5. **The record is complete.** The service's confirmation, the credit report the agent relied on, and the model behind the proposal all join the same timeline.

3. Where the gateway adds value

The gateway works best when a platform team can route agents' model, MCP, and agent-to-agent traffic through it, including agents whose code you can't change. One gateway can serve multiple applications in the same environment, applying each application's own rules, so a platform team can run it as a shared service.

The application keeps its existing model or protocol client and needs no VeriProof code. The client may still need endpoint, authentication, or error-handling changes for the chosen profile. [1]

Advantages

- **A controlled traffic path.** Apply policy before forwarding supported requests and before releasing protected responses. Enforce model, data, tool, route, and spending limits at that boundary.
- **Enforcement outside the agent process.** The routed check runs in a separate service. When other routes are blocked, every request the agent sends is checked, with no VeriProof code in the agent.
- **Shared connection administration.** Manage approved destinations and gateway-owned upstream credentials through one runtime instead of separate application implementations.
- **Tool and handoff control.** Mediate approved MCP tool calls and supported agent-to-agent handoffs. When the route carries the complete business request and enough trusted context, it can be the business action's enforcement boundary.
- **No VeriProof code in the client.** Add controls to a supported third-party or existing client that can change its connection settings but cannot embed the VeriProof SDK.
- **A nearby endpoint for the SDK.** Send the SDK's routine calls to the Edge API, close to the application. [1]

What it takes

Gateway coverage follows the routed traffic, as it does for any proxy-based control. An in-process operation, direct database connection, browser action, or alternative outbound route is outside that coverage unless it is separately connected. Standard practice is to make the gateway the agent's only way out. Agent workloads send traffic only to the gateway, which holds the provider credentials so the agent has no key of its own, and where the provider supports it, the provider accepts calls only from the gateway's private endpoint or identity.

Operating a gateway adds a service to maintain, and its capacity, health, storage, updates, credentials, and recovery all need an owner.

The gateway also adds a network step, and inspection or protected buffering can affect response time and streaming. Clients without the SDK must handle the route's escalation, timeout, cancellation, and retry behavior, because the gateway does not hold a connection open while an escalation waits for a person to respond.

Illustrative use case: a purchasing tool behind the gateway. A procurement agent sends purchase-order requests through an approved MCP tool. If the gateway receives the full order, trusted authority, and the facts needed by policy, it can check the request before dispatch. The tool service still creates the order and reports what happened.

4. Coverage and integration choice

Legend: ● **Full support**; ◐ **Partial support**; **n/a** = not applicable; **blank** = not supported.

Full support means the approach covers the requirement within its connected boundaries. Partial support covers only some operations, protocols, or profiles. Not applicable means the requirement does not arise. **Both** means one application using the SDK and the gateway proxy together. [1, 2]

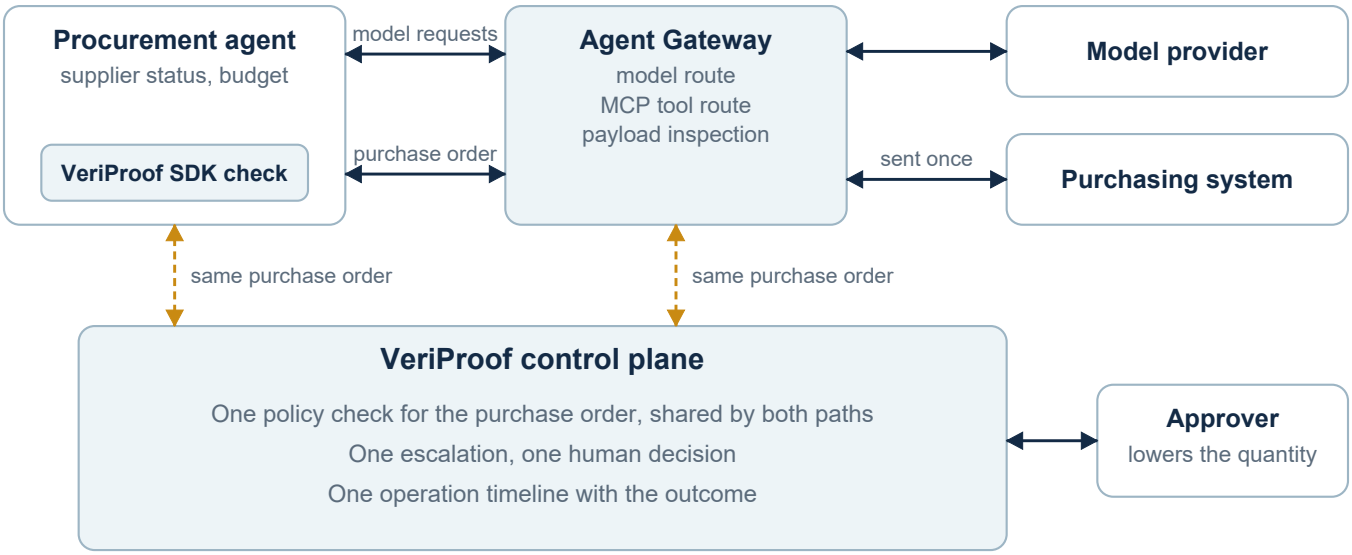
Category	Requirement	SDK	Gateway	Both
Application coverage	Check model inputs and outputs	●	●	●
	Control native MCP and A2A traffic	◐	●	●
	Protect internal functions and database writes	●		●
	Add controls to a client that cannot embed an SDK		●	n/a
Context and control	Use internal records and workflow state in policy	●	◐	●
	Use industry-typed facts in rules and evidence	●	◐	●
	Mask sensitive data before it leaves the application	●		●
	Intercept traffic outside the agent process		●	●
	Keep upstream credentials outside the agent		●	●
Human escalation	Pause protected work for an authorized decision	●	●	●
	Apply authorized edits to business actions	●	◐	●
	Resume multi-step application workflows	●	◐	●
Evidence and recovery	Record decisions, escalations, and reported outcomes	●	●	●
	Attach the documents behind a decision	●		●
	Record the model, token use, and cost of each call	●	●	●
	Recover protected operations after restart	●	●	●
	Resolve uncertain writes in a business system	◐	◐	◐
Automatic coordination	Reuse equivalent SDK and proxy checks	n/a	n/a	●
	Share the same escalation across SDK and proxy	n/a	n/a	●
	Unify SDK and proxy records and accounting	n/a	n/a	●

Reading the partial ratings. The SDK checks MCP and agent-to-agent calls only when the application wraps them. The gateway can use typed facts and internal records, apply edits, and resume workflows only for requests that pass through it; steps inside the application need the SDK. An uncertain write is settled only if the receiving system can confirm it. When a route escalates to a person, a client without the SDK gets a blocked response with a reference to the escalation and can continue after the decision. [2, 3, 5, 6]

5. Using both in the same application

When one application uses both, VeriProof coordinates them for you. Built-in setup checks confirm that the SDK and gateway are ready to work together, and no code is needed to remove duplicate checks or to join their escalations and records.

Consider a procurement agent that submits purchase orders through an approved MCP tool on the gateway. The application also wraps that submission with the SDK, because only the application knows the supplier's current status and the remaining budget. Both paths see the same action at the same phase.



Here is what happens to one purchase order:

1. **The SDK checks the purchase order** with facts only the application has, such as the supplier's status and the remaining budget.

2. **The gateway sees the same purchase order** on its MCP tool route. It reuses VeriProof's decision and still inspects what passes through.

3. **The order needs approval,** so one escalation opens and both paths wait on it.

4. **The approver lowers the quantity.** The revised order is checked once with the new quantity and current facts, then sent once.

5. **The purchasing system confirms the result.** Only its response shows whether the order was created, and that outcome joins the same timeline. [3, 5, 6]

What automatic coordination provides

1. **One check per action.** When both paths see the same action, VeriProof checks it once and reuses only its own verified decision.

2. **One escalation, one execution.** Both paths wait on the same escalation, and the approved action runs once, even across retries and restarts.

3. **One timeline.** The gateway's model, MCP, and agent-to-agent requests link to the SDK action that caused them, each labeled with the boundary that checked it, and the timeline counts the action once.

6. A practical rollout

Start with one agent and its most consequential action, then widen coverage from there.

1. **Map the actions.** List the agent's external requests and business side effects, and mark which route or application function can stop each one.

2. **Name the facts.** For the most consequential action, identify the facts its rules need, where each one comes from, and the system that records the outcome.

3. **Choose the integration.** Use the coverage matrix to pick the SDK, the gateway, or both for that action.

4. **Test the paths that matter.** Try allowance, denial, human escalation, permitted changes, timeouts, restarts, and duplicate delivery at the boundaries the application uses. Run new rules against real activity before they enforce. [4]

5. **Connect both, if you use both.** Connect through the generated setup and confirm the coverage shown in the portal. Then check that one human escalation releases the approved action once and that both paths appear in the action's workflow.

Who owns what

When you use both, each team owns its part. The application team owns business execution, and the platform team runs the gateway and its routes. The governance team writes the rules and decides who handles each human escalation.

7. Recommendation and references

Use the gateway for supported model, MCP, and A2A traffic, including business actions that arrive complete through a tool route. Use the SDK for work the application owns and for decisions that need its internal records. Use both when one application needs both kinds of control.

If one integration already covers an action with the context it needs, add the other only to close a named gap. A client that cannot embed the SDK uses the gateway, and an effect outside every connected boundary needs its own integration.

References

1. **Agent Gateway.** https://docs.veriproof.app/agent-integrations/edge-gateway/

2. **SDKs.** https://docs.veriproof.app/sdk/

3. **Session continuation.** https://docs.veriproof.app/human-review/session-continuation/

4. **Policy evaluations.** https://docs.veriproof.app/sdk/policy-evaluations/

5. **Human input.** https://docs.veriproof.app/human-review/human-input-profiles/

6. **Continuation recovery.** https://docs.veriproof.app/human-review/continuation-recovery/

7. **Industry coverage library.** https://docs.veriproof.app/policy/industry-enums/

Try one workflow with your team

Pick one agent and its most consequential action. Connect it through the SDK, the gateway, or both, then trigger a human escalation and check that the approved action runs once.

Start your free 60-day trial
veriproof.app/start