PANTHEON CONTROL / SECURITY RESOURCES

Pantheon Control Integration and Deployment Guide

Proposed requirements for connecting agent evidence and existing security controls

Prepared for technical evaluators, security architects, and integration teams · October 2026

This guide describes a proposed integration and deployment approach for Pantheon Control. It is a planning document for discovery and prototype work, not a list of currently supported connectors. The public demo is simulated and does not connect to customer systems.

The central design rule is to keep evidence correlation separate from enforcement. Pantheon may help decide and coordinate, but an existing or purpose-built control must receive the decision on the action path to allow, deny, or hold an operation before it executes.

1. Deployment principles

- Begin with read-only discovery and event collection. Do not request response permissions until a specific use case requires them.
- Record the source of every fact, when it was observed, and whether it is complete or inferred. Keep missing identity, route, and target data visible.
- Use deterministic customer policy for consequential decisions. Model-generated summaries may help an analyst but should not authorize a tool action or destructive response.
- Keep prevention, post-event detection, containment, remediation, and verification as distinct states in the product and audit record.
- Design for least privilege, tenant isolation, encrypted transport and storage, auditable approval, bounded retention, and a documented failure mode.

2. Reference deployment model

A reference deployment has four logical parts. Customer systems remain the authoritative sources and enforcement points. A Pantheon service may normalize evidence, relate identities to resources and policies, and stage approved response. The final deployment location, network boundaries, tenancy model, and data residency need to be chosen with customers before implementation.

Part	Responsibilities	Design questions
Agent and tool paths	MCP clients and servers, application functions, connectors, browsers, code runtimes, workflows, and delegated agents.	Which route is used? Who owns it? What identity acts? Where can a request be intercepted?
Customer source systems	Identity provider, endpoint and workload tools, cloud, SaaS, API, data, vulnerability, network, and recovery systems.	Which APIs and audit events are available? Which actions can the customer authorize?
Pantheon evidence and policy plane	Inventory, event normalization, relationship context, deterministic policy evaluation, case records, and response coordination.	What data is necessary? Where is it processed and stored? How are tenants isolated and administrators controlled?

Part	Responsibilities	Design questions
Enforcement and response controls	MCP gateway or server, application/API authorization, agent host, identity, endpoint, cloud, workflow, SaaS, and data controls.	How does the control receive a decision? Does it return an acknowledgment? What happens on timeout or bypass?

3. Integration inventory

Integration area	Evidence to collect	Possible action or boundary
Remote MCP	Client, server, transport, tool/resource names, call metadata, principal, target, result.	Gateway/server may deny or hold a routed request; non-routed calls bypass it.
Local MCP stdio	Agent host, process launch, local server config, tool name, identity, file/network scope.	Requires host, runtime, endpoint, or application control; network visibility alone is insufficient.
APIs and function calls	Model-request record, actual application execution, service identity, API target, result.	Enforce in the tool runtime, API gateway, application, or destination authorization.
SaaS connectors	Connector owner, OAuth grant, actual principal, scopes, target records, action result.	Use connector policy or SaaS-native controls; inherited user access may be broader than intended.
Browser agents	Managed session, authenticated identity, site, UI action, target result.	Use managed browser/session policy or application authorization; API logs may not show UI events.
Shell, CI, and containers	Host/job/runtime, command, process tree, secrets references, cloud principal, changed resource.	Use endpoint, container, CI, secret broker, and cloud IAM controls; inspect child processes and environment access.
Workflow and delegation	Trigger, run ID, step graph, parent/child agent lineage, queue, identity transitions.	Use orchestrator approvals and per-step checks; asynchronous actions need full run tracing.
Identity and access	Human sponsor, service/workload principal, grant, token scope, role changes, session status.	Read-only inspection first; narrowly revoke or reduce scope only with customer authorization.
Endpoint, cloud, data, vulnerability, recovery	Asset, exposure, runtime alerts, data class, access path, patch state, backup and restore status.	Correlate evidence and stage actions in the source tool; verify the source state after the change.

4. Event and relationship data

A useful action record links the request to the operation that actually ran. Keep source event identifiers so an analyst can return to the original system. Store observed values separately from enrichment and policy conclusions.

Field group	Minimum proposed fields
Event	Event ID, event time, source system, source record reference, ingestion time, schema version.
Agent and owner	Agent ID, runtime ID, owner, human sponsor when known, model/application label if supplied by source.

Field group	Minimum proposed fields
Identity	Actual principal, identity type, delegated scope or grant reference, authentication/session reference.
Route and operation	Path type, server/tool/API/function/connector, requested operation, executed operation, downstream status.
Target and context	Resource identifier, environment, data classification when available, destination, transaction size or volume.
Decision and response	Policy ID/version, decision, decision reason, approval actor/time, enforcement point, acknowledgment, response and verification state.
Confidence and gaps	Source confidence, missing values, enrichment source, timestamps, unresolved relationship.

Do not silently infer that a model request was executed, that a user identity was the effective principal, or that a deny succeeded. Store a clear state such as requested, observed, allowed, denied with enforcement receipt, approval held, response staged, completed, or verified.

5. Data handling requirements

Data category	Recommended treatment for a prototype
Agent and tool metadata	Collect identifiers and configuration needed to inventory owners, runtimes, servers, tools, and permissions. Avoid collecting prompts or full message bodies by default.
Action telemetry	Prefer event metadata and a source pointer. Collect content only when the use case requires it, the customer approves it, and retention/access controls are defined.
Identity and permission data	Use stable references and scopes needed to resolve effective access. Avoid storing bearer tokens, passwords, private keys, or reusable secrets.
Sensitive data context	Use source classifications, labels, resource references, or derived categories where possible. Do not copy customer records into the evidence plane merely to label their sensitivity.
Policy and approval records	Retain policy version, input facts, decision, approver, time, enforcement receipt, and response outcome for the customer-defined period.
Operational and diagnostic logs	Minimize payloads; restrict administrator access; define retention, export, deletion, and incident-response procedures before production use.

Before production, publish a data inventory that states each field, purpose, source, storage location, access role, retention period, and deletion method. Document encryption, tenant isolation, subprocessors if any, cross-border handling if applicable, and customer export options. These are requirements to implement and validate; this guide does not assert that Pantheon currently meets them.

6. Permissions and response governance

Permission tier	Examples	Control requirement
Read-only discovery	List agents, grants, assets, tool configurations, and audit events.	Use scoped service identities; record source and last-sync time; make coverage gaps visible.

Permission tier	Examples	Control requirement
Policy evaluation	Evaluate a proposed action against customer rules.	No write permission is required to run shadow decisions; preserve policy version and inputs.
Approval workflow	Request authorization for a high-impact deny, pause, revoke, isolate, or configuration change.	Show scope, expected effect, evidence, rollback or recovery option, and named approver before execution.
Response execution	Disable a connector, revoke a scoped session, isolate a host, patch a service, or alter workflow access.	Require a separate, least-privilege credential; verify source-system acknowledgment and resulting state.

Response actions should be narrow and reversible where practical. Define which actions may run automatically, which require approval, and how service outages, stale evidence, conflicting policy, or unavailable enforcement are handled. Do not label a post-event alert or recommendation as a prevented action.

7. Deployment and validation sequence

Phase	Work	Acceptance evidence
0. Select	Name one business workflow, agent owner, runtime, route, target, and risk boundary.	Customer-approved scope, source list, excluded routes, data and success criteria.
1. Discover	Connect read-only inventory and identity sources for the selected path.	Owner confirms the inventory; unknowns and sync age are explicit.
2. Trace	Ingest test events and link requests to actual principals, targets, and outcomes.	Measured attribution, source fidelity, duplicates, missing fields, and event latency.
3. Shadow policy	Evaluate test actions without changing production behavior.	Analyst agreement, false-positive/exception rate, decision latency, and reason quality.
4. Enforce in test	Test one deny and one approval hold at the actual on-path control.	Control receipt precedes downstream action; bypass, timeout, fail-open/closed, and rollback behavior documented.
5. Respond and verify	Execute one approved containment or remediation in a controlled environment.	Approval, scope, source-system receipt, post-change readback, and workflow regression check.

8. Failure modes to test before production

- MCP request bypasses the managed gateway or uses a local stdio server.
- The agent uses a direct API, SaaS connector, browser session, or delegated agent route outside the chosen control.
- Identity or target resolution is missing, stale, or ambiguous.
- The policy service or enforcement point is unavailable or times out.
- A deny decision is recorded but the destination still completes the operation.
- A containment action revokes too much access or breaks an approved business workflow.
- A remediation reports success but the source system still exposes the same vulnerable or over-permissioned path.
- An event contains sensitive prompt, customer, or secret material that should not be retained.

9. JADEPUFFER applicability

Sysdig reported JADEPUFFER as an attacker-operated campaign that exploited an internet-facing vulnerable Langflow service, then used credential discovery and infrastructure access to affect other systems. MCP governance alone would not have blocked the initial application exploit. A broader integration plan would need exposure and vulnerability findings, workload/runtime telemetry, identity and secrets context, network reachability, data-system events, and recovery evidence. Any prevention or response claim remains contingent on connected sources and tested controls.

10. Discovery checklist

- Which agent and workflow should the first prototype cover? Who owns it?
- What exact action paths can the agent use today, including local tools and alternate connectors?
- Which principal authorizes each action, and where are its effective permissions defined?
- Which system is the source of truth for tool configuration, identity, target, data classification, and exposure?
- Where can the organization inspect and stop a request before the target executes it?
- What fields may be collected, where may they be processed, and how long may they be retained?
- Which response actions require human approval, and how will success be verified?
- What test will prove the agent’s intended workflow continues to work after the policy change?

References

- NIST, Artificial Intelligence Risk Management Framework 1.0, NIST AI 100-1, https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf. NIST states the framework is voluntary and organizes risk work through Govern, Map, Measure, and Manage.
- Model Context Protocol, Specification, https://modelcontextprotocol.io/specification/2026-07-28.
- OWASP GenAI Security Project, Top 10 for Agentic Applications, https://genai.owasp.org/2025/12/09/owasp-top-10-for-agentic-applications-the-benchmark-for-agentic-security-in-the-age-of-autonomous-ai/.