SAWBILL

WHITE PAPER / 02

CONTROL

From Authorization to Effect

How Sawbill governs agent execution

How Sawbill places a control boundary between what an agent proposes and what may change a real system.

READERS

Platform architects / Security leaders / DevOps and SRE leaders / AI platform leaders

Contents

From Authorization to Effect

The issue in one minute

A useful agent must do more than produce text. It must read a repository, execute tools, run validations, and sometimes request a Git, cloud, or application mutation. Giving it the required credentials directly, however, conflates its cognitive capability with a right to act.

Sawbill introduces a structural separation:

> **the agent proposes and produces; a separate admission path decides; a controlled enforcement point holds the credential and performs the effect.**

This separation increases autonomy without turning every agent into a standing administrator.

Autonomy is not authority

In a conventional agentic chain, an orchestrator invokes a model, supplies tools, and considers the task complete when the provider returns `Completed`. That approach leaves several questions unanswered:

- does the executed work package still match the authorized request?
- are policies, branches, budgets, and identities still valid?
- can the worker call an external provider directly?
- does a timeout mean failure, success, or an unknown result?
- can a retry unknowingly repeat an effect that already occurred?

Sawbill does not treat execution as a simple job queue. It treats it as a series of governed transitions, each with an owner and an explicit ceiling on its guarantees.

The Sawbill execution path

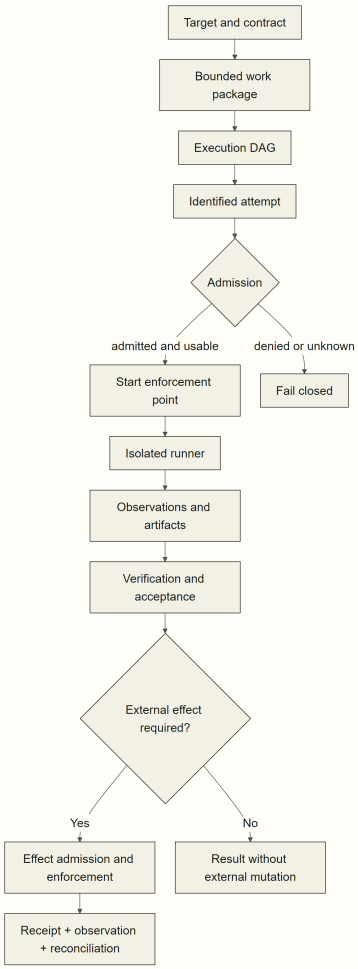

Figure 1 - Sawbill

This path avoids two extremes: an agent with no practical capability, or an agent holding general credentials and monitored only after the fact.

1. Turn intent into bounded work

The work package describes what an execution may attempt. It binds, among other things:

- an objective and exit criteria;
- the exact inputs;
- the read and write scope;
- admissible tools or capabilities;
- policies and verification profiles;
- time, cost, and data limits;
- explicitly permitted or prohibited effects.

The package is not an enriched prompt. It forms a boundary between the governed mission and the cognitive choices left to the agent.

An execution DAG then orders packages through explicit dependencies. This DAG replaces neither the system architecture nor the graph of requirements. It answers only one question: in what order may this work be attempted?

2. An attempt is an invocation, not an eternal task

An attempt has its own identity. It represents a physical invocation under a precise set of inputs, policies, and capabilities.

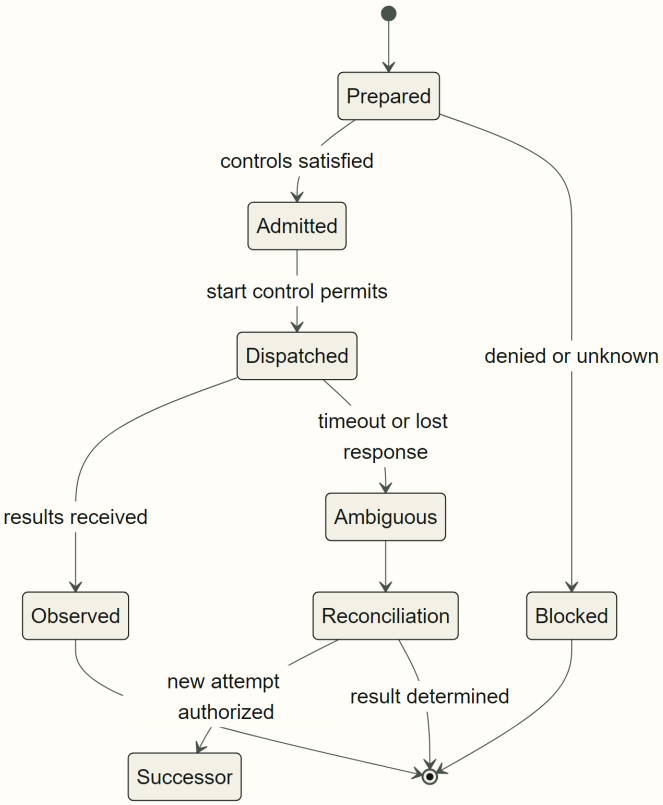

Figure 2 - Sawbill

A retry does not rewrite the previous attempt. A provider change or restart creates a successor. The new context does not automatically recover old authorizations, budgets, or capabilities.

This rule preserves history and prevents a technical restart from becoming a silent replay of authority.

3. Admission decides; the enforcement point controls

Sawbill distinguishes a policy decision from its physical enforcement:

- the admission engine evaluates exact inputs and produces a historical decision;
- the Policy Enforcement Point, or PEP, rereads current state before start;
- the runner executes only the capability entrusted to it;
- the coordinator organizes the procedure without owning a second authority path.

This architecture follows the spirit of zero trust: no implicit trust is granted because the worker runs on the right network, in the right cluster, or under the right service account. NIST SP 800-207 already distinguishes authentication and authorization before resource access. Sawbill extends the discipline to agentic attempts and their exact objects.

At dispatch time, the PEP can verify:

- workload identity and qualification;
- the package version and its inputs;

- current policy and authorization validity;
- the expected branch, head, or resource state;
- the budget and capabilities that remain usable;
- the absence of invalidation since the initial decision.

A historically positive decision is therefore insufficient when context has changed.

4. Integrate Sawbill without replacing the delivery chain

Sawbill sits between the tools that propose or execute work and the resources whose mutation commits the organization. Orchestrators, runtimes, CI systems, runners, and cloud platforms can remain in place. Their role changes: they receive a bounded mission and return observations; they no longer carry authority, acceptance, or the right of effect on their own.

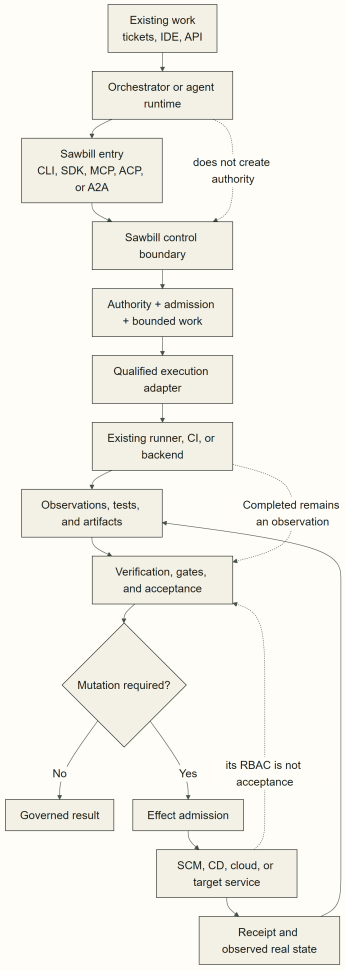

Figure 3 - Sawbill

The boundary shift is precise. Before Sawbill, the orchestrator often owns the prompt, context, tools, and effect credentials. With Sawbill, it retains the cognitive flow, while authority, admission, evidence, and effect pass through distinct and verifiable controls.

Integration families

Delivery-chain point	Interfaces and endpoint examples	Role retained by Sawbill
Work management	APIs and webhooks from Linear, Plane, GitHub, GitLab, or Azure DevOps	A ticket remains a projection; it becomes neither a contract nor a verdict
Agent runtimes	CLI, SDK, MCP, ACP, A2A; runtimes such as OpenCode or Microsoft Agent Framework	The adapter transports bounded execution; the runtime does not self-admit
Orchestration and compute	Workflow backends, CI, containers, Kubernetes, or batch	External status remains an observation to correlate and verify
SCM and delivery	Repositories, checks, pipelines, and deployment systems	Merge, release, and deployment remain separately admitted effects
Identity, secrets, and keys	IdP, workload identity, secrets vault, KMS, or HSM	Credentials remain behind their boundary; they do not become business authority
Observability	Logs, traces, audit logs, and event buses	Signals are attributed, redacted, and bound before becoming usable evidence

These names illustrate documented integration endpoints. They are neither architectural dependencies nor a version-by-version availability matrix. Effective compatibility always depends on the selected adapter's profile and qualification.

5. The agent does not hold the effect credential

The most important control appears when work must produce an external mutation: push a commit, merge a branch, change a cloud resource, send a message, or trigger a deployment.

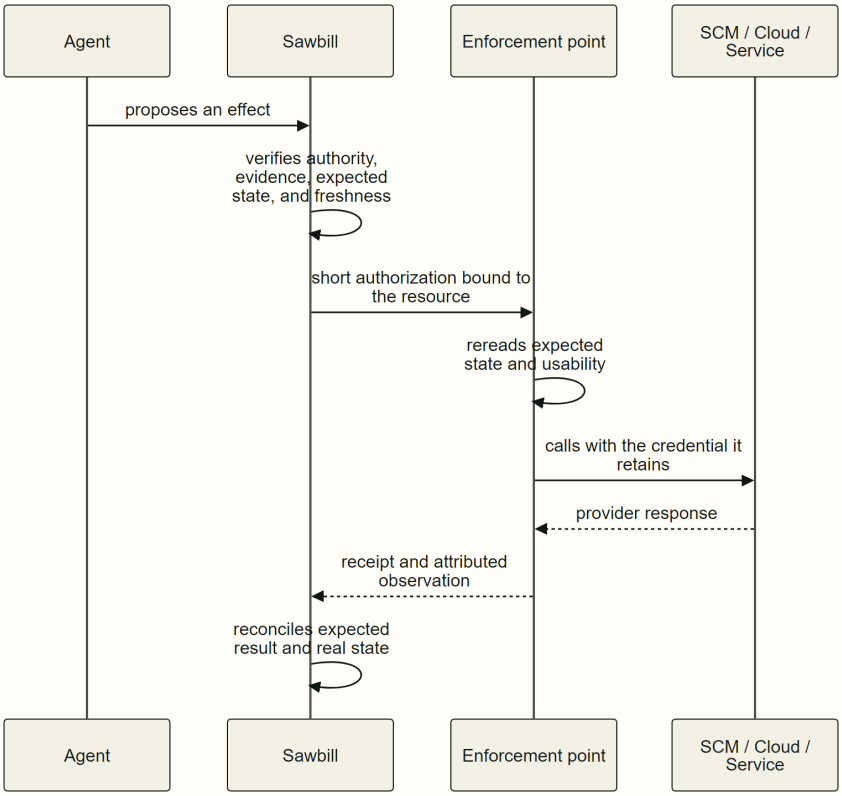

Figure 4 - Sawbill

The runner does not receive the secret and cannot bypass the enforcement point. The effect capability is short-lived and bound to a purpose, resource, and expected state. It does not become a general bearer token.

This separation reduces the blast radius of a hostile prompt, compromised tool, or agent that leaves its scope.

6. An effect has its own admission

Authorization to execute an agent is not authorization to modify a resource. Sawbill therefore separates:

1. admission of the attempt;
2. production and verification of its results;
3. admission of the external effect;
4. enforcement of the effect at the resource control point.

An agent can compile, test, or prepare a change without gaining the right to publish it. Likewise, an accepted result may require a new decision before deployment.

Control applies to expected state. A request to "change the branch if its head is X" is denied if the head has become Y. This check protects against valid decisions made about a world that no longer exists.

7. Provider success remains an observation

An HTTP 2xx status means a request was received or processed according to the service's semantics. It does not prove that the business objective was achieved. RFC 9110 defines HTTP semantics, not Sawbill acceptance semantics.

Sawbill therefore retains separately:

- persistence of the request;
- provider acceptance of the start;
- observation of a process;
- the service response;
- real state observed after the effect;
- the assurance or acceptance decision.

A `Completed` status from a workflow engine, CI system, or agent runtime does not automatically become a positive gate.

8. Ambiguity triggers reconciliation, not replay

Remote systems produce ambiguous results: the request may have been applied even though the response was lost. Retrying immediately is dangerous, especially for a non-idempotent mutation.

Sawbill opens a reconciliation case:

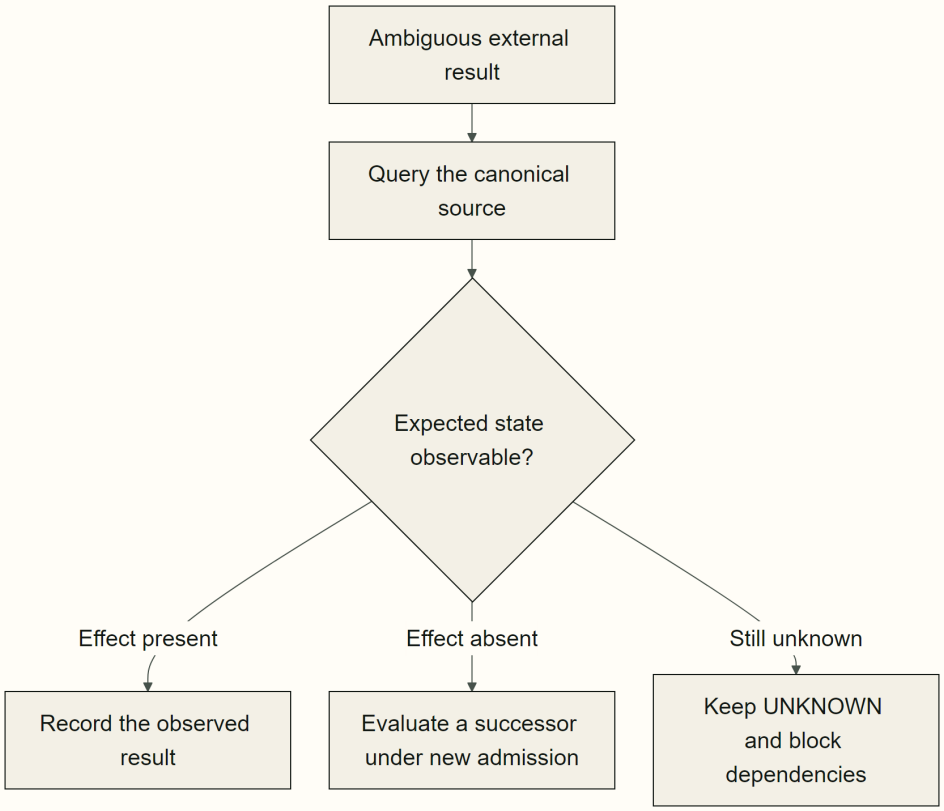

Figure 5 - Sawbill

Uncertainty is neither optimistic success nor a failure that automatically permits a retry. It remains visible until a qualified observation allows the chain to continue.

9. Parallelize without adding individual permissions

Two packages that are separately admissible are not necessarily admissible together. They may share a resource, exceed an aggregate budget, or produce non-commutative effects.

Sawbill therefore evaluates a parallel set as a set: dependencies, conflicts, capabilities, budgets, and effects are checked at the same cut. Selection remains a scheduling proposal, never additional authority.

This property becomes essential when an orchestrator launches several specialist agents across the same portfolio of repositories.

Scenario: prepare and publish the data export

Return to the customer data export capability.

1. The plan creates separate packages for the API, access control, tests, and deployment configuration.
2. The development agent receives only the sources and tools needed for its package.
3. Admission verifies the input versions, budget, and delegated authority.
4. The enforcement point starts an isolated runner without giving it a production credential.
5. The runner produces a proposal, tests, and observations.
6. Quality and security controls are evaluated separately.
7. An effect request proposes merging the exact commit onto the expected head.

8 The SCM adapter retains the credential, rereads the head, and performs the mutation.

9 A receipt is retained, then the branch and deployment are observed.

10 If the provider response is ambiguous, reconciliation precedes any new attempt.

The chain lets the agent work quickly while preventing a scope violation from immediately becoming a real-world effect.

Enterprise value

Autonomy with a smaller blast radius

Agents can explore and produce without receiving general secrets. Sensitive effects remain concentrated in hardened enforcement points.

Consistent control across tools

A local runtime, workflow engine, CI system, or remote backend remains an execution adapter. Changing providers does not change authority, admission, or acceptance semantics.

Explainable recovery

Attempts, retries, and ambiguous effects remain distinct. Operations can reconstruct what was requested, observed, and reconciled without inventing a result from a partial log.

Security before effect

Sawbill does not stop at detecting an action after the fact. It places controls on the path to the mutation.

What each signal actually allows you to conclude

Signal	Permitted conclusion	Prohibited conclusion
Package selected	The work is a scheduling candidate	It may start
Positive admission	Exact inputs satisfied policy at the cutoff	The context is still fresh
Dispatch accepted	The enforcement point submitted a bounded start	The work is correct or complete
Provider `Completed`	The provider reported its state	Sawbill gates are satisfied
Effect receipt	A precise request was observed under the profile	The real objective was achieved
Real state observed	A condition was found within a scope	The complete deliverable is acceptable

Official references

Source	Status	Contribution used	Limit retained by Sawbill
NIST SP 800-207 - Zero Trust Architecture	Final NIST publication	Resource-centered access decisions without implicit network trust	Sawbill adds bindings for agentic actions, attempts, and effects

Source	Status	Contribution used	Limit retained by Sawbill
NIST SP 800-218 - Secure Software Development Framework 1.1	Final NIST publication	Secure software development practices and shared vocabulary	The framework does not provide Sawbill runtime admission
RFC 9110 - HTTP Semantics	Internet Standard	Exact interpretation of HTTP requests and responses	An HTTP status does not become a business verdict

Adopting a standard does not automatically transfer its guarantees. Every integration retains an explicit contract and a ceiling on its guarantees.