CEDAR AUTHORIZATION PRIMER

Cedar

Authorization Architecture

How policy decisions become enforced, observable controls

An architecture primer based on Cedar documentation, CNCF project status, and a dated NVIDIA OpenShell engineering case.

FIRST-PARTY SOURCES / INDEPENDENT LUMINITY SYNTHESIS



Cedar Sits Inside a Larger Authorization System

Production assurance depends on request assembly, complete enforcement, current data, controlled change, observable decisions, and bounded failure behavior.

AT A GLANCE

3
system duties
authenticate, authorize, enforce

3
deployment patterns
embedded, local, hosted

4
state domains
identity, policy, entity, schema

1
enforcement inventory
every protected action

Runtime Control System 01

Applications assemble trusted requests, call a decision point, enforce the result, and retain evidence.

Assurance Boundary 02

Policy correctness cannot compensate for missing enforcement, stale inputs, or uncontrolled delivery.

IN THIS PRIMER

Key Points

01 **Authorization Layer**
Identity, mapping, context, and request assembly

02 **Enforcement Path**
Interception, decision handling, execution, and evidence

03 **Deployment Patterns**
Embedded, local, and hosted decision points

04 **Policy and Entity Distribution**
Versioned state, refresh, rollback, and revocation

05 **Delivery and Assurance**
Validation, review, release, observation, and rollback

06 **Agentic Systems**
Delegation, tools, credentials, endpoints, and ceilings

07 **Adoption and Acceptance**
Coverage, observability, ownership, and residual gaps

Cedar authorizes. The surrounding system authenticates, supplies facts, invokes Cedar, and enforces the decision.

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar documentation and CNCF project page · verified Nov 13, 2026

Cedar Authorization Architecture

Cedar can externalize policy decisions only after the application establishes identity, maps the action, and assembles trusted data.

01

What the application establishes

- ✓ Authenticated identity — UPSTREAM
- ✓ Action mapping — EXPLICIT
- ✓ Resource lookup — TYPED
- ✓ Relevant entity slice — CURRENT
- ! Missing or untrusted input — MATERIAL

Application Logic → Externalized Decision

02

How the request reaches Cedar

Identity · principal | Action · operation | Resource + context

1 **Assemble** — Identity and facts

2 **Normalize** — Typed values

3 **Authorize** — Policy decision

4 **Enforce** — Guard operation

IDENTITY
Authentication remains outside Cedar

MAPPING
Application operations become Cedar actions

INPUTS
Facts require provenance, shape, and freshness

03

Four Architecture Boundaries

Authentication Is Separate — AUTHN

The application establishes who or what is requesting access.

Mapping Is Explicit — MAP

Protected operations map to named Cedar actions and resources.

Freshness Is Visible — STATE

Identity, policy, entity, and schema state remain distinguishable.

Enforcement Is Owned — PEP

The caller must apply Cedar's result before the operation executes.

ARCHITECTURE BOUNDARY

- Authenticate upstream
- Map actions explicitly
- Supply trusted facts
- Enforce every result
- Retain provenance

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar Authorization and Cedar Security · verified Nov 13, 2026

Cedar Authorization Architecture

A policy decision becomes a control only when every protected operation passes through an enforcement point.

01

What the enforcement point does

- ✓ Request intercepted — BEFORE ACTION
- ✓ Decision returned — ALLOW OR DENY
- ✓ Diagnostics handled — OBSERVABLE
- ✓ Operation guarded — ATOMIC
- ! Bypassed operation — UNCONTROLLED

Policy Decision Point → Policy Enforcement Point

02

How a protected action executes

Entry · request boundary | Decision · Cedar result | Exit · guarded operation



COVERAGE
Inventory every protected action

COMPOSITION
Compound operations may need multiple checks

EVIDENCE
Decision, version, and disposition form the record

03

Four Enforcement Interpretations

Every Protected Action — INVENTORY

An uncovered route remains outside the authorization control system.

Compound Authorization — COMPOSE

Multi-step operations need checks at each material authorization boundary.

Deny Path — FAIL

Denials and evaluation errors must block the protected operation safely.

Evidence Event — AUDIT

Record the request identity, decision, versions, and enforcement disposition.

ENFORCEMENT RULE

- Inventory operations
- Intercept before action
- Block on deny
- Handle errors safely
- Record disposition

Luminity Digital
INTELLIGENCE, ENGINEERED, DELIVERED.

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar Authorization Architecture

Embedded, local, and hosted decision points trade latency, isolation, operating ownership, and failure behavior.

01

What changes by topology

✓ Latency path	NEAR OR REMOTE
✓ Failure domain	LOCAL OR SHARED
✓ Scaling model	APP OR SERVICE
✓ State distribution	PULL OR SERVE
! Unbounded outage behavior	UNSAFE

One Policy Model → Different Runtime Topologies

02

Three deployment patterns

Embedded SDK | Local Decision Point | Hosted Authorization Service



EMBEDDED

Low call overhead, application-owned lifecycle

LOCAL

Near-workload service with separately refreshed state

HOSTED

Shared policy service with network dependency

03

Four Topology Checks

Decision Proximity — LATENCY

Choose a path that meets the operation's latency and availability needs.

State Ownership — OWNER

Assign who distributes, refreshes, verifies, and rolls back policy state.

Isolation Boundary — FAILURE

Separate application failures from shared authorization failures deliberately.

Experimental Agent — STATUS

The Cedar Local Agent repository labels its crate experimental.

TOPOLOGY CHECK

- Set latency budget
- Model failure domain
- Assign state ownership
- Bound stale behavior
- Test recovery

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar Authorization Architecture

Authorization state is distributed state. Policy, entity, schema, and identity freshness require separate controls.

01

What must remain current

- ✓ Policy version — IDENTIFIED
- ✓ Entity version — IDENTIFIED
- ✓ Schema version — COMPATIBLE
- ✓ Identity validity — TIME-BOUNDED
- ! Stale or unknown state — ALARM

Published State → Runtime Decision Point

02

How state reaches the runtime

Policy · signed version | Entities · current slice | Schema · compatible model

1 **Publish** — Versioned state

2 **Verify** — Source and integrity

3 **Refresh** — Within bound

4 **Serve** — Known version

PROVENANCE
Know who produced each version

FRESHNESS
Age is measured against an explicit bound

REVOCATION
Urgent changes need a faster path than routine refresh

03

Four Distribution Controls

Policy Provenance — SOURCE

Bind each deployed policy set to an approved source and immutable version.

Entity Freshness — DATA

Measure how old decision-relevant entity data can become safely.

Stale-State Alarm — ALARM

Surface refresh failures and the version still serving decisions.

Rollback and Revocation — RESPONSE

Prepare both routine rollback and urgent invalidation paths.

FRESHNESS RULE

- Version each domain
- Verify provenance
- Measure state age
- Alarm on refresh failure
- Test revocation

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar Local Agent and Cedar Security · verified Nov 13, 2026

Cedar Authorization Architecture

Schema validation and policy analysis improve quality, but controlled delivery and runtime observation complete the assurance system.

01

What the delivery path controls

- ✓ Model and author — TRACEABLE
- ✓ Validate and test — REPEATABLE
- ✓ Review and approve — INDEPENDENT
- ✓ Release and observe — VERSIONED
- ! Unreviewed change — BLOCKED

Policy Change → Governed Release

02

How a policy reaches production

Author · typed model | Assure · tests + analysis | Release · controlled version

1 **Author** — Policy change

2 **Validate** — Schema and tests

3 **Approve** — Independent review

4 **Deploy** — Observe and roll back

VALIDATION	ASSURANCE	RUNTIME
Find type, action, and attribute errors	**Test intended outcomes and review model changes**	**Observe decisions, failures, and enforcement disposition**

03

Four Release Controls

Schema Migration — MODEL

Treat model changes as migrations with compatibility and revalidation checks.

Independent Review — REVIEW

Separate policy authorship from approval for material access changes.

Retained Evidence — RECORD

Keep tests, analysis, approval, version, and deployment records together.

Rollback Ready — BACKOUT

Define triggers, authority, and a verified path to the last known-good set.

RELEASE GATE

- Validate the model
- Test intended outcomes
- Approve independently
- Observe runtime
- Keep rollback ready

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar Policy Validation and Luminity Delivery Model · verified Nov 13, 2026

Cedar Authorization Architecture

Agentic systems add delegation, tool calls, credentials, and network actions that require distinct authorization decisions.

01

What enters the agent boundary

- ✓ User and agent identity — DISTINCT
- ✓ Sub-agent delegation — BOUNDED
- ✓ Tool and endpoint — NAMED
- ✓ Credential use — PURPOSE-LIMITED
- ! Ambient authority — UNBOUNDED

Delegated Intent → Per-Action Decision

02

How an agent action is constrained

Principal · agent or sub-agent | Action · tool or network use | Resource · credential or endpoint

1 **Delegate** — Bound authority

2 **Check** — Per action

3 **Invoke** — Tool or endpoint

4 **Record** — Decision and use

"Cedar balances expressivity, readability, and decidability."

NVIDIA OpenShell engineering note · Nov 9, 2026

RFC + implementation OPEN development builds only

03

Four Agentic Interpretations

Per-Action Authorization — ACTION

Authorize material tool, credential, data, and network actions separately.

Connection vs Request — LAYER

Opening a connection need not authorize an inspected request over it.

Least-Privilege Ceilings — CEILING

Agents and sub-agents remain within explicit authority ceilings.

MCP Schema Work — MCP

Cedar for Agents includes tooling to derive Cedar schemas from MCP tools.

DEVELOPMENTAL CASE

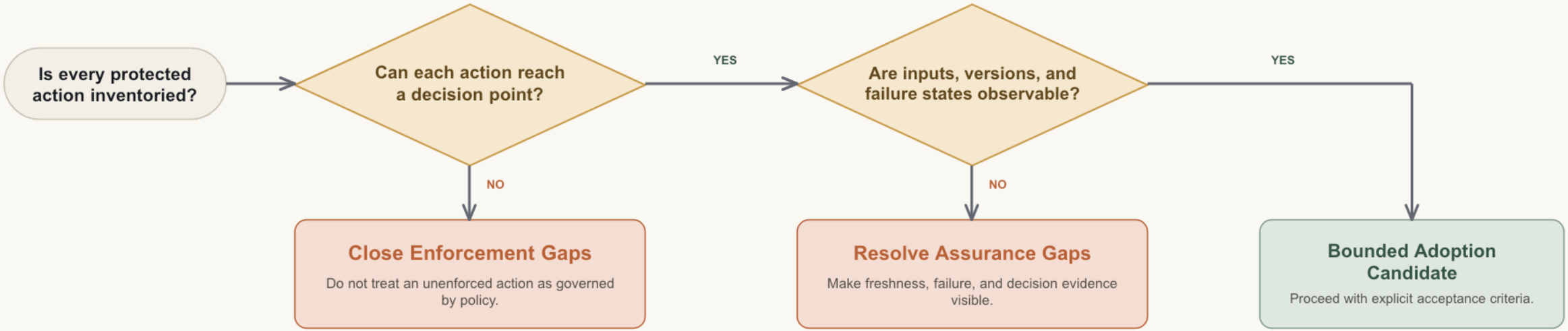
- OpenShell RFC is open
- Implementation PR is open
- Stable builds reject Cedar policies
- Treat as evidence, not adoption

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar for Agents and NVIDIA OpenShell Engineering Note · verified Nov 13, 2026

Cedar Authorization Architecture

Adopt Cedar when enforcement coverage, state controls, operating ownership, and failure behavior are explicit and testable.



RETAIN THESE CONTROLS WITH EVERY RELEASE

Source Ownership — OWNER

Name owners for identity, policy, entity, schema, and enforcement state.

Availability and Latency — SLO

Set measurable decision budgets and bounded failure behavior.

Change and Incident Contro — CONTROL

Keep approval, rollout, revocation, rollback, and incident evidence.

Residual Gaps — ACCEPT

Record uncovered operations, stale-state tolerance, and accepted exceptions.

ACCEPTANCE RULE

- Prove enforcement coverage
- Observe state and failures
- Assign operating owners
- Accept residual risk explicitly

Luminity Digital
INTELLIGENCE, ENGINEERED, DELIVERED.

© 2026 Luminity Digital, Inc. | Independent synthesis of Cedar first-party sources.

Cedar Security and Luminity Acceptance Model · verified Nov 13, 2026