

Network Management Operations
Internet-Draft
Intended status: Informational
Expires: 28 August 2026

L. C. Rodríguez
D. Lopez
A. Mendez
Telefonica
24 February 2026

Model for distributed authorization policy sharing
draft-cabanillas-nmop-authz-policy-sharing-model-01

Abstract

This document defines mechanisms and conventions for the representation, lifecycle management, and distribution of authorization policies in distributed and automated environments. It specifies a consistent, machine-readable, and interoperable framework that enables policies to be validated, versioned, exchanged, and removed across heterogeneous systems and domains.

The framework defines how authorization policies, expressed in declarative Policy-as-Code (PaC) languages, are encapsulated, managed, and distributed using YANG as the canonical representation format. This separation allows independent evolution of policy languages, enforcement architectures, and trust models.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://LuciaCabanillasRodriguez.github.io/authz-policy-sharing-model/draft-cabanillas-nmop-authz-policy-sharing-model.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-cabanillas-nmop-authz-policy-sharing-model/>.

Discussion of this document takes place on the Network Management Operations Working Group mailing list (<mailto:nmop@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/nmop/>. Subscribe at <https://www.ietf.org/mailman/listinfo/nmop/>.

Source for this draft and an issue tracker can be found at <https://github.com/LuciaCabanillasRodriguez/authz-policy-sharing-model>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Conventions and Definitions	4
3. Requirements for Policy Management	4
4. Policy-as-Code and Declarative Policy Languages	5
5. Policy representation in YANG	5
6. Architecture Overview	8
6.1. Functional Roles	8
6.2. Functional Interaction	9
7. Other Models	10
8. Security Considerations	11
9. IANA Considerations	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Acknowledgments	12

Authors' Addresses	12
--------------------	----

1. Introduction

The increasing complexity and automation of distributed systems, such as programmable networks, multi-cloud platforms, and intent-based infrastructures, require scalable and interoperable mechanisms for managing authorization policies. In these environments, policies are no longer confined to static configuration files or manually managed. Instead, they are dynamic artifacts that must be created, validated, distributed, updated, and eliminated programmatically.

Authorization policies increasingly govern not only access control but also operational behavior, compliance requirements, and governance constraints across multiple domains. These policies may be authored in one domain, distributed through another, and enforced in many. As a result, consistent handling of policies throughout their lifecycle becomes critical.

Existing approaches often rely on system-specific policy formats and ad hoc distribution mechanisms, leading to several challenges:

- * Fragmentation: Incompatible representations and semantics hinder interoperability and auditability.
- * Limited lifecycle control: Many systems lack standardized mechanisms for validation, versioning, and decommissioning.
- * Trust ambiguity: In multi-domain environments, it is often unclear whether a policy source is authorized or trustworthy.
- * Governance gaps: Systems frequently lack mechanisms to verify whether a policy author is permitted to define policies for a specific domain.

To address these challenges, this document defines a structured and interoperable framework for representing and managing authorization policies as governed artifacts. YANG is used as the canonical representation format for policy artifacts. A YANG-defined policy encapsulates its metadata together with embedded declarative Policy-as-Code content, which is treated as opaque executable logic. This structured representation enables schema-based validation, immutable semantic versioning, and verifiable lifecycle transitions while remaining agnostic to the underlying evaluation engine.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC2119 [RFC8174] when, and only when, they appear in all capitals, as shown here.

- * Policy: A rule or set of rules that define behavior, access, or operational constraints within a system.
- * Authorization policy: A policy that governs access or permissions based on user/agent, resource, or environmental attributes.
- * Policy lifecycle: The set of stages through which a policy passes, including creation, validation, versioning, distribution, update, and removal.
- * Policy-as-Code (PaC): A paradigm in which policies are represented as declarative code artifacts, allowing automation, versioning, and testing.

3. Requirements for Policy Management

Systems that manage or exchange authorization policies across domains MUST satisfy the following requirements:

- * Granularity: Policies SHOULD be able to express fine-grained authorization rules over users, resources, and contextual conditions.
- * Lifecycle control: Policies MUST support creation, versioning, validation, and removal.
- * Interoperability: Policy representations SHOULD be portable and interpretable across different administrative domains.
- * Verifiability: The framework MUST provide mechanisms to verify policy integrity and provenance.

4. Policy-as-Code and Declarative Policy Languages

Declarative Policy-as-Code (PaC) languages, such as Rego [Rego], Cedar [Cedar], or ALFA [ALFA], are widely used to express authorization logic in distributed systems. Although these languages differ in syntax, evaluation models, and execution environments, they share common lifecycle and governance requirements when used in distributed and multi-domain environments.

This framework treats PaC content as opaque executable logic embedded within a YANG-defined policy artifact. The YANG representation does not interpret, validate, or constrain the internal semantics of the policy language. Instead, it provides a structured and interoperable container for lifecycle governance, version control, provenance binding, and distribution.

By separating policy handling from policy semantics, including evaluation logic and decision outcomes, this framework enables interoperability without constraining innovation in policy languages or enforcement technologies.

Example in Rego syntax:

```
package example
# Allow read access if the user has the "read" role
default allow = false
allow {
    input.user.role == "read"
}
```

The policy logic above is treated as opaque content by the YANG representation. Its evaluation behavior is determined exclusively by the target execution environment, while lifecycle and governance properties such as version and ownership are managed independently.

5. Policy representation in YANG

YANG provides a structured and schema-driven mechanism for representing authorization policies as managed and governed artifacts. In this framework, YANG serves as the canonical container format for policy definitions, encapsulating:

- * Policy metadata, including owner, description and version.
- * The declarative language used to express the policy logic.
- * The embedded Policy-as-Code (PaC) content, treated as opaque data.

* Optional leaf for validation and provenance.

Each policy instance MUST include a semantic version following a controlled versioning scheme (for example, Git-style tags such as v1.0.0). Version values MUST uniquely identify immutable policy content. Once a specific version is stored, it MUST NOT be modified. Any change to the policy logic or its governance metadata MUST result in the creation of a new version. Versioning is therefore a fundamental requirement of Policy-as-Code governance. Maintaining historical versions enables controlled updates, rollback to previous versions, auditability, and forensic analysis in case of misconfiguration or dispute.

In addition, each policy instance MUST include an explicit owner attribute that identifies the authority responsible for the policy definition, ensuring accountability within and across domains. By associating a policy with a clearly identified authority, the framework enables governance controls and allows systems to determine whether the policy source is authorized within a given scope. The owner attribute MUST be expressed as a URI that uniquely identifies the authoritative entity responsible for the policy.

The specific URI structure is determined by the administrative environment. Ownership metadata MAY be cryptographically linked to the policy provenance, enabling verification against the provenance signature key. This mechanism ensures that the policy origin and integrity can be independently verified and trusted.

The YANG model below illustrates a simplified structure for representing authorization policies as managed artifacts:

```
module authz-policy {
  namespace "urn:ietf:params:xml:ns:yang:authz-policy";
  prefix pac;
  organization
    "IETF NMOP";
  contact
    "WG Web:    <https://datatracker.ietf.org/wg/nmop/>
    WG List:    <mailto:nmop@ietf.org>
  Authors:
    Lucía Cabanillas <mailto:lucia.cabanillasrodriguez@telefonica.com>
    Diego López <mailto:diego.r.lopez@telefonica.com>
    Ana Méndez Pérez <mailto:ana.mendezperez@telefonica.com>";
  import ietf-yang-provenance {
    prefix iyangprov;
  }
  description
    "Illustrative YANG model for representing authorization policies as managed artifacts";
```

```
revision 2026-02-10 {
  description
    "Second revision";
  reference
    "RFC XXXX: Model for distributed authorization policy sharing";
}
container policy {
  leaf description {
    type string;
    description
      "Optional human-readable description of the policy";
  }
  leaf language {
    type enumeration {
      enum rego {
        description "The policy is written in Rego syntax";
      }
      enum cedar {
        description "The policy is written in Cedar syntax";
      }
      enum alfa {
        description "The policy is defined in ALFA format";
      }
    }
    mandatory true;
    description
      "Specifies the language used to express the policy";
  }
  leaf pac {
    type string;
    mandatory true;
    description
      "Example:      package example
      # Allow read access if the user has the 'read' role
      default allow = false
      allow {
        input.user.role == \"read\"
      }";
  }
  leaf owner {
    type uri;
    mandatory true;
    description
      "URI identifying the authoritative entity responsible for this policy";
  }
  leaf version {
    type string;
    mandatory true;
  }
}
```

```
        description
            "Semantic version of the policy following Git-style tags (e.g., v1.0.0)";
    }
    leaf policy-provenance {
        type iyangprov:provenance-signature;
        description
            "Signature proving provenance of the policy";
    }
}
```

This YANG snippet demonstrates how policy content can be represented as structured data while keeping the logic in a declarative format. By explicitly indicating the language, management systems can validate and process policies appropriately, enabling interoperability between tools and engines.

6. Architecture Overview

Policy management in distributed environments relies on a set of functional components that cooperate to define, govern, distribute, evaluate, and enforce authorization policies across administrative domains [RFC2904]. This framework adopts that functional separation while introducing structured policy representation and lifecycle governance based on YANG.

6.1. Functional Roles

Policy Author: The entity (human or automated system/agent) responsible for creating the policy definition. The author produces a YANG-encoded policy document that includes metadata (description, version, language, owner) and the actual declarative rule.

Policy Administration Point (PAP): The Policy Administration Point manages the lifecycle and governance of policy artifacts. Upon receiving a YANG-encoded policy, it performs schema validation and, where applicable, verifies provenance using cryptographic mechanisms such as those described in [I-D.ietf-opsawg-yang-provenance].

The PAP enforces version uniqueness and immutability, ensuring that existing versions are never modified. Any change to policy content or metadata results in the creation of a new version. The PAP maintains historical versions to support rollback and audit requirements.

Before accepting or distributing a policy artifact, the Policy Administration Point (PAP) MAY perform an authorization verification step. In this step, the PAP queries a Policy Decision Point to determine whether the declared owner or submitting entity is authorized to define or modify policies within the relevant domain.

All lifecycle events, including creation, update, activation, rollback, and decommissioning, MUST be recorded in an append-only accounting ledger to ensure traceability and auditability.

Once validated and authorized, the PAP distributes the executable policy artifact to the appropriate decision components.

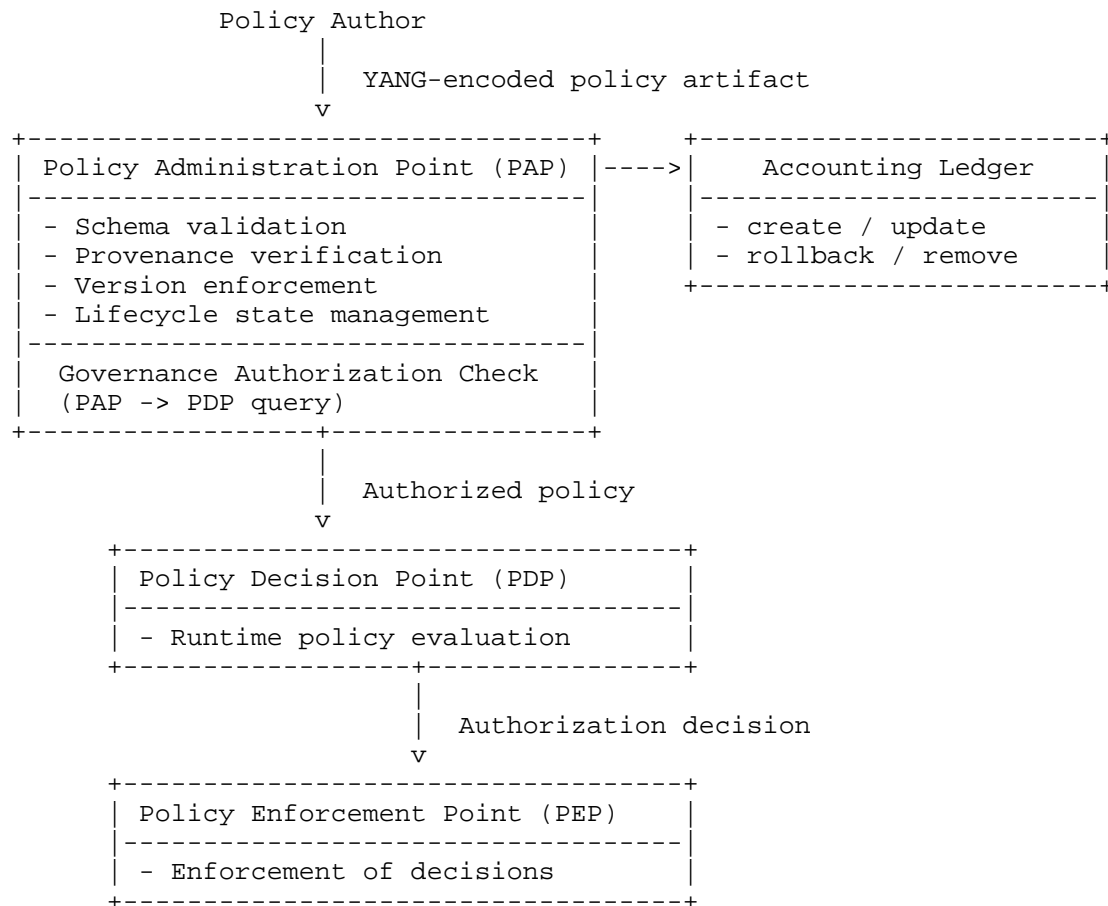
Policy Decision Point (PDP): The Policy Decision Point evaluates policy logic at runtime. It receives validated policy artifacts and processes authorization requests based on contextual attributes, claims, or environmental information. The PDP produces authorization decisions such as Permit or Deny, optionally including obligations or advice.

Policy Enforcement Point (PEP): The PEP enforces the decisions issued by the PDP. Enforcement may include granting or denying access, applying configurations, or triggering operational actions.

6.2. Functional Interaction

The interaction among the architectural components reflects a strict separation between policy governance and runtime decision-making. Policy artifacts are first validated, governed, and recorded as managed entities before being distributed for evaluation.

The following diagram illustrates the logical interaction flow:



7. Other Models

Existing data models demonstrate that YANG can be effectively used to carry authorization-related information in operational environments.

The OpenConfig gNSI authorization model [OCgNSI] defines a YANG module that represents metadata associated with gRPC authorization policies installed on network devices. That model focuses on device-level state and observability, including policy versioning, creation time, and success/failure counters collected during authorization evaluation.

This document is complementary to that approach. While the OpenConfig model concentrates on operational visibility for a specific enforcement technology, the framework defined here focuses on the representation, lifecycle management, provenance, and distribution of authorization policy artifacts across systems and administrative domains.

8. Security Considerations

Ensuring the integrity, authenticity, and provenance of policy data is critical to prevent unauthorized modification. Policies SHOULD include cryptographic protection mechanisms that allow their origin and validity to be verified.

The mechanisms defined in [I-D.ietf-opsawg-yang-provenance] — Applying COSE Signatures for YANG Data Provenance — provide a suitable foundation for these protections. That document specifies how COSE signatures [RFC9052] are used to include digital signatures within the YANG data, enabling, in this case, verifiable provenance and ensuring the integrity of the YANG-based distributed authorization policy sharing model proposed in this draft.

When such provenance mechanisms are applied to policy definitions, each policy instance can include a verifiable signature providing proof of origin and integrity of the provided policy. By treating policies as signed, versioned artifacts, this framework reduces the risk associated with automated and cross-domain policy exchange, including the additional risks in multi-domain deployments such as determining whether a policy has been modified in transit and whether the policy issuer is authorized to define policies applicable to the receiving domain.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[I-D.ietf-opsawg-yang-provenance]
Lopez, D., Pastor, A., Feng, A. H., Prez, A. M., and H. Birkholz, "Applying COSE Signatures for YANG Data Provenance", Work in Progress, Internet-Draft, draft-ietf-opsawg-yang-provenance-03, 3 November 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-yang-provenance-03>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, DOI 10.17487/RFC2904, August 2000, <<https://www.rfc-editor.org/rfc/rfc2904>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <<https://www.rfc-editor.org/rfc/rfc9052>>.

10.2. Informative References

- [ALFA] "ALFA (Abbreviated Language For Authorization)", n.d., <<https://alfa.guide/alfa-authorization-language/>>.
- [Cedar] "Cedar Policy Language", n.d., <<https://docs.cedarpolicy.com/>>.
- [OCgNSI] "OpenConfig gNSI Authorization Model", n.d., <<https://www.netconfcentral.org/modules/openconfig-gnsi-authz/2024-02-13/source/raw/>>.
- [Rego] "A Policy Language for Open Policy Agent", n.d., <<https://www.openpolicyagent.org/docs/latest/policy-language/>>.

Acknowledgments

This document is based on work partially funded by the iTrust6G project (Grant agreement N 101097083) and the ROBUST-6G project (Grant agreement N 101139068).

Authors' Addresses

Luca Cabanillas Rodrguez
Telefnica
Email: lucia.cabanillasrodriguez@telefonica.com

Diego Lopez
Telefnica
Email: diego.r.lopez@telefonica.com

Ana Mendez
Telefnica
Email: ana.mendezperez@telefonica.com