Network Management Operations
Internet-Draft
Intended status: Informational
Expires: 20 April 2026

L. C. Rodr鱈guez
D. Lopez
A. Mendez
Telefonica

17 October 2025

Model for distributed authorization policy sharing
draft-cabanillas-nmop-authz-policy-sharing-model-00

Abstract

This document defines mechanisms and conventions for the representation and sharing of authorization policies in distributed and automated environments. It specifies the foundational elements required to express policies in a consistent, machine-readable, and interoperable manner, enabling fine-grained control and context-aware evaluation.

The framework supports the complete policy lifecycle, including creation, validation, versioning, distribution, and decommissioning, with YANG serving as the canonical representation format. It also establishes the relationship between policy representation and the structure of tokens used in enforcement and authorization exchanges, ensuring coherent and dynamic policy evaluation across heterogeneous systems.

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

Copyright Notice

Table of Contents

1. Introduction

The increasing complexity and automation of distributed systems, particularly in areas such as network operations, multi-cloud orchestration, and service automation, demand more precise, interoperable, and dynamic management of authorization policies.

Mechanisms based on static configurations or manual administration interfaces no longer provide the scalability, consistency, or adaptability required in current operational environments.

Infrastructures, such as programmable networks, multi-cloud platforms, and intent-based systems, require that policy enforcement components be capable of evaluating policies automatically and contextually, using structured representations that can be validated, exchanged, and updated programmatically. In such environments, policies may be distributed and enforced through various control elements — for example, domain-specific controllers, service orchestrators, or autonomous agents operating at the edge.

Policies are no longer limited to simple access control or configuration parameters; they now define operational behavior, compliance, and governance across multiple administrative and technological domains. These policies may be expressed and applied at any level — from low-level configuration directives and resource constraints to high-level intents that describe desired operational outcomes in declarative form.

However, the absence of a standardized representation model introduces several persistent challenges:

* Fragmentation: Different systems implement incompatible policy formats and semantics, hindering interoperability and auditability.

* Limited granularity: Many policy models lack the expressiveness needed to capture contextual or fine-grained conditions.

* Lifecycle gaps: The lack of versioning, validation, and decommissioning mechanisms increases operational and compliance risks.

To address these issues, this document defines a set of mechanisms and requirements for consistent policy representation, sharing, and evaluation, enabling interoperability among systems that rely on policies for authorization and decision-making.

Within this framework, YANG serves as the canonical representation format for policies. By defining the policy' s metadata, structure, language, and logic in YANG, the Policy Administration Point (PAP) can validate and manage the policy lifecycle, then transform the YANG description into executable policy modules for one or more Policy Decision Points (PDPs).

It enables:

* Provenance verification , through cryptographic signatures that bind policy content to its origin and authority.

* Schema-based validation , ensuring that policy attributes and logical structures comply with agreed models.

* Lifecycle consistency , allowing creation, update, and retirement to be managed under uniform semantics.

In this framework, YANG acts as the source of truth for policy metadata and content, while Policy-as-Code (PaC) approaches provide the executable layer that translates these definitions into enforceable logic.

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 evaluation: The process of determining whether a request complies with a defined policy.

* Context-aware policy: A policy that adapts its evaluation outcome based on runtime context, such as environmental or identity-specific data.

* 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

Policy systems operating across multiple domains MUST satisfy the following requirements:

* Granularity: Ability to express fine-grained authorization rules over users, resources, and contextual conditions.

* Context-awareness: Support for evaluation based on attributes such as device state, network condition, or environment.

* Token alignment: Tokens used for enforcement (e.g., WIMSE tokens) MUST include the contextual fields and claims required for correct evaluation.

* Lifecycle control: Policies MUST support creation, versioning, validation, and retirement.

* Interoperability: Policy representations SHOULD be portable and interpretable across different administrative domains.

4. Policy-as-Code and Declarative Policy Languages

The Policy-as-Code model represents policies in a declarative format, allowing them to be defined, validated, and deployed programmatically. Declarative languages such as Rego are well suited for expressing logical authorization conditions in executable form. However, a key interoperability challenge lies in defining what a policy can evaluate — and consequently, what contextual information must be available at runtime. Without a clear, standardized understanding of the minimum evaluable elements, tokens (such as WIMSE tokens) may omit essential claims, leading to inconsistent authorization outcomes.

Therefore, the YANG representation described in this framework:

* Specifies the language (e.g., Rego, ALFA, Cedar) and its expected evaluation scope.

* Defines, via schema constraints, which attributes or contextual fields a PDP must expect to receive.

Example in Rego syntax:

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

This example illustrates a minimal policy that depends on a single contextual attribute (user.role). The associated YANG model would specify that this attribute is required for correct evaluation, and thus the corresponding WIMSE token MUST include this claim.

5. Policy representation in YANG

YANG provides a structured, schema-driven representation that defines:

* The policy metadata (identifier, description, version).

* The declarative language used to express the policy (e.g., Rego).

* The logical rule content.

* Optionally, validation and provenance extensions.

This canonical form ensures policies can be validated, versioned, and translated programmatically.

The example below illustrates a minimal model that links declarative logic with its structural:

```
module authz-policy {
    namespace "urn:ietf:params:xml:ns:yang:authz-policy";
    prefix pex;
    organization
        "IETF NMOP";
    contact
        "WG Web:   <https://datatracker.ietf.org/wg/nmop/>
        WG List:  <mailto:nmop@ietf.org>
    Authors:
        Lucia Cabanillas <mailto:lucia.cabanillasrodriguez@telefonica.com>
        Diego Lopez <mailto:diego.r.lopez@telefonica.com>
        Ana Mndez Prez <mailto:ana.mendezperez@telefonica.com>";
```

```
    description
        "A simple illustrative model for representing a declarative policy, including its
 language and rule content.";
    revision 2025-10-15 {
        description
            "First revision";
        reference
            "RFC XXXX: Model for distributed authorization policy sharing";
     }
    container policy {
        leaf id {
            type string;
            description
                "Unique identifier for the policy instance.";
            }
        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.";
                }
            }
            description
                "Specifies the language used to express the policy.";
        }
        leaf rule {
            type string;
            description
                "Example:    package example
                # Allow read access if the user has the ’read’ role
                default allow = false
                allow {
                    input.user.role ==\"read\"
                }";
        }
    }
}
```

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 relies on a set of functional components that cooperate to define, validate, distribute, and enforce authorization policies across systems and administrative domains. In this framework, YANG serves as the canonical container for policy definitions, providing a structured and verifiable representation that includes both metadata and the declarative policy logic (Policy-as-Code , PaC).

The Policy Administration Point (PAP) is the central manager: it extracts the PaC from the YANG, validates it, and distributes it to one or more Policy Decision Points (PDPs).

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 (identifier, version, language) and the actual declarative rule (PaC).

Policy Administration Point (PAP): The PAP manages the full lifecycle of policies. It receives the YANG policy, validates its schema and provenance (e.g., using COSE signatures as described in [I-D.ietf-opsawg-yang-provenance]), and extracts the embedded PaC rule. The PAP can also transform or adapt the PaC for the target PDPs if needed, but the original logic remains intact. Finally, the PAP distributes the validated and executable PaC to the relevant PDPs.

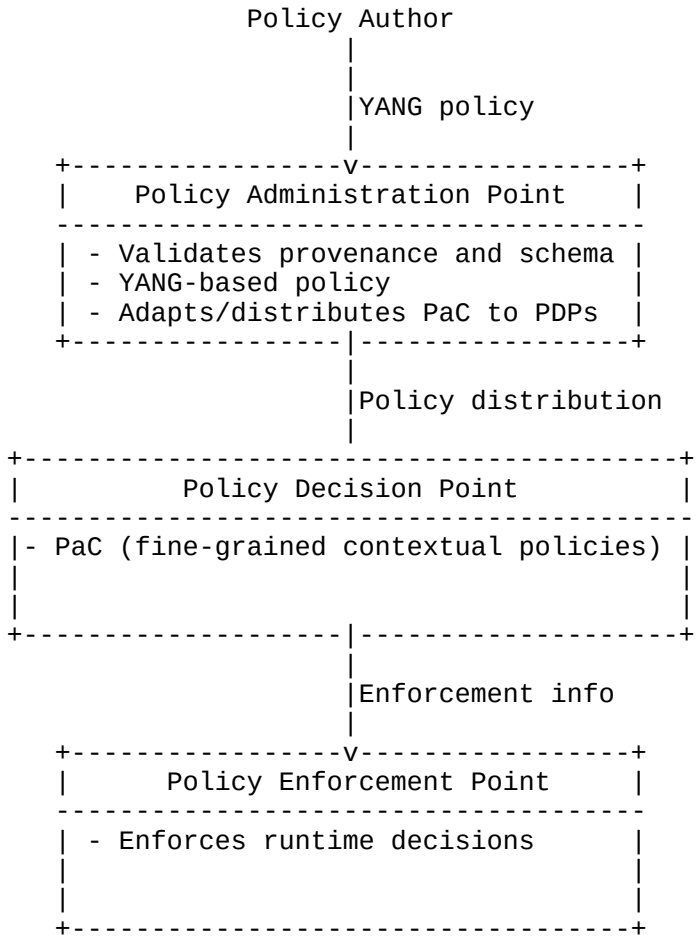
Policy Decision Point (PDP): The PDP receives the executable PaC from the PAP and performs runtime evaluation. Decisions are made based on contextual attributes, claims, and tokens (e.g., WIMSE tokens), producing authorization outcomes such as Permit, Deny, or Obligations.

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 following describes the operational flow of policies across the functional components, highlighting how YANG-based policy definitions and PaC are handled.

'''



'''

7. Security Considerations

Ensuring the integrity, authenticity, and provenance of policy data is critical to prevent unauthorized modification or injection of malicious logic. 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 bind signatures to YANG elements, enabling verifiable provenance and ensuring policy integrity.

When such provenance mechanisms are applied to policy definitions, each policy instance can include a verifiable signature or evidence chain linking it to its authoritative source.

8. IANA Considerations

This document has no IANA actions.

9. 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-01, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-yang-provenance-01>.

[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>.

[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>.

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
Telefonica
Email: lucia.cabanillasrodriguez@telefonica.com

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

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