Model for distributed authorization policy sharing

IETF 124 - NMOP

draft-cabanillas-nmop-authz-policy-sharing-model-00

Lucía Cabanillas Rodríguez(lucia.cabanillasrodriguez@telefonica.com)

Diego López(diego.r.lopez@telefonica.com)

Ana Méndez Pérez(ana.mendezperez@telefonica.com)

Why a Model for Distributed Authorization Policy Sharing?

- Network and service environments, even agents, are **distributed and automated** (multi-domain, multi-cloud, etc.)

- **Authorization policies** are needed everywhere — but today they are:
  - Fragmented across network operations
  - Difficult to validate, version, or share
  - Inconsistent in lifecycle management

So many systems, so many policies… none speaking the same language

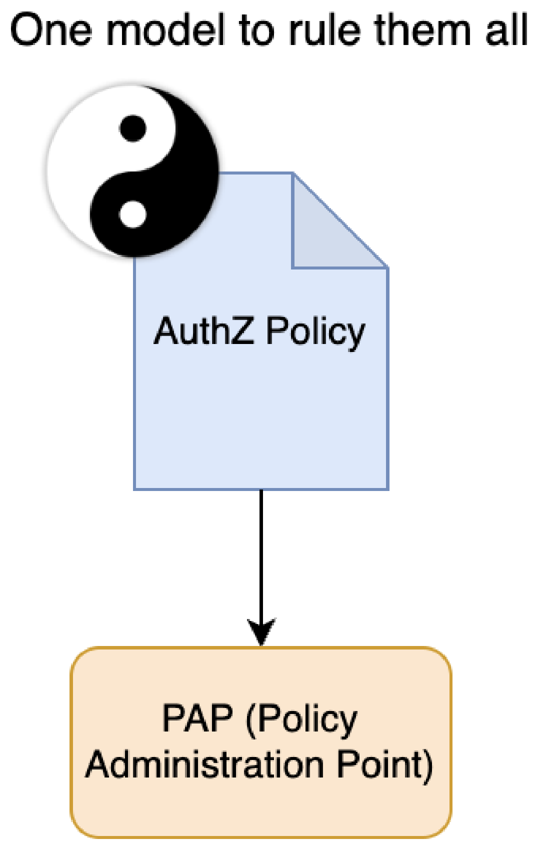
- **Manual or static mechanisms** do not scale for automation

- We need **a common model** for expressing, sharing, and enforcing policies across heterogeneous systems

- *Goal:* Enable **interoperable, machine-readable, lifecycle-managed** policies

YANG-based Framework for Policy Sharing

- Define **a canonical representation** of authorization policies using **YANG**
- Enable **Policy-as-Code (PaC)**: policies as structured, executable artifacts
- Cover the **entire policy lifecycle**:
  - Creation, validation, versioning
  - Distribution, decommissioning
- Establish **alignment with tokens** used at runtime
- *Core idea:* YANG is the **source of truth**, and PaC provides the **enforcement logic**



Architecture Overview

Policy Author

- Defines the policy in **YANG format**, including metadata, language, and logic
- Ensures clarity and consistency of policy intent

Policy Administration Point (PAP)

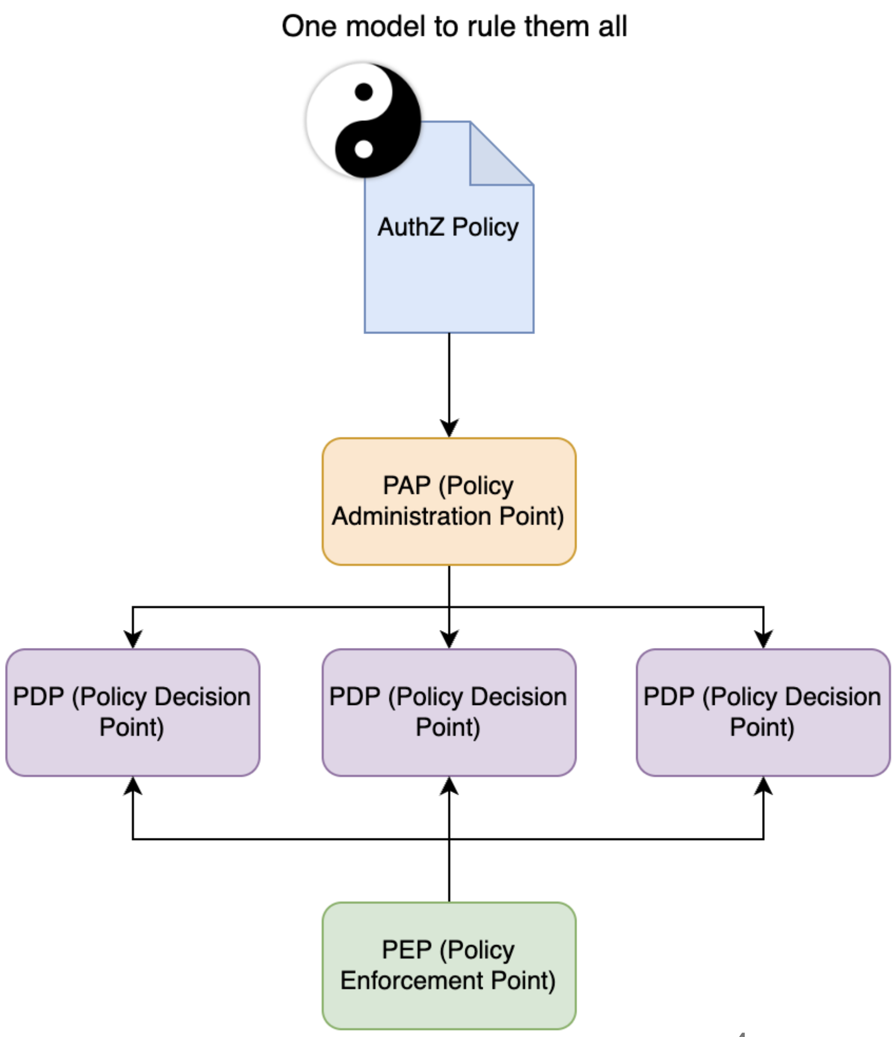
- **Validates** the policy schema and **verifies provenance** (e.g., via COSE signatures)
- **Extracts** the embedded *Policy-as-Code (PaC)* logic
- **Distributes** validated and versioned policies to PDPs

Policy Decision Point (PDP)

- **Evaluates** the PaC dynamically using contextual inputs and tokens
- Produces **authorization decisions**

Policy Enforcement Point (PEP)

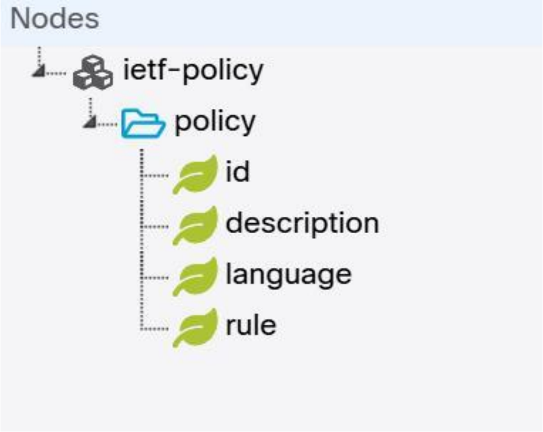
- **Applies** the decisions from the PDP in real time



Declarative Policy Encoded in YANG

Each policy definition includes:

- **Metadata** — identifier, version, and policy language
- **Declarative logic** — rule content expressed in *Rego*, *Cedar*, or *ALFA*
- **Provenance information (optional)** — COSE signatures or traceable origin data https://datatracker.ietf.org/doc/draft-ietf-opsawg-yang-provenance/




```
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\"
            }";
    }
```

Enables **schema-based validation**, **automated transformation**, and **lifecycle consistency**

Next Steps and Open Points

- First draft (version 00) — open for feedback

- Next steps:
  - Refine model structure and YANG modules
  - Explore integration with existing provenance mechanisms
  - Work on a Reference Implementation of the architecture to be demonstrated at a next Hackathon

- Questions for the WG:
  - Are the identified lifecycle and interoperability requirements relevant?