Model for distributed authorization policy sharing

IETF 124 - NMRG

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)

Problem Statement

Why this work?

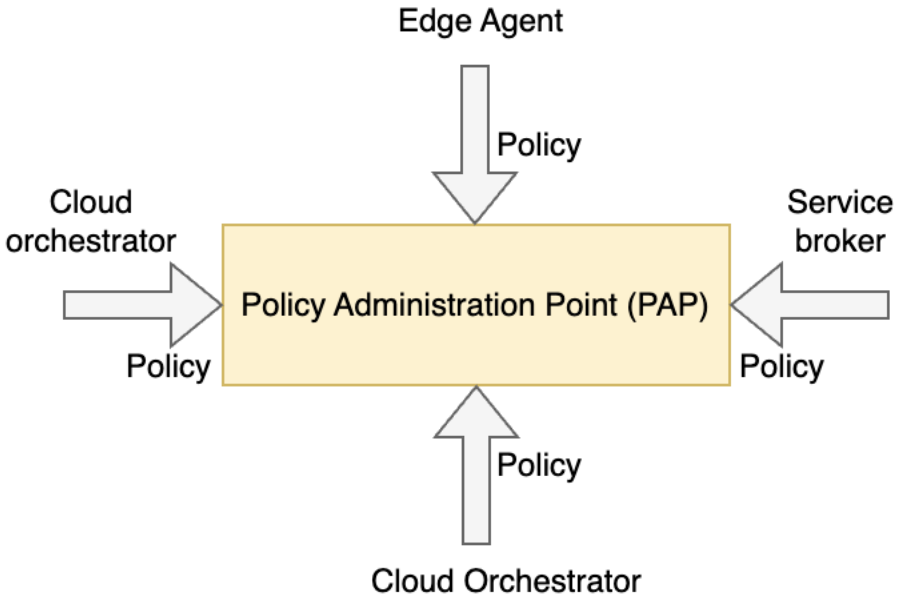
- Distributed and automated systems rely increasingly on **dynamic authorization**
- **Static policies and manual configurations** cannot keep pace with adaptive, intent-driven operations

Current Gaps

- **Fragmentation:** Each domain defines its own policy language and semantics
- **Inconsistency:** No common model for validation, updates, or lifecycle control
- **Limited interoperability:** Hard to share or audit policies across domains

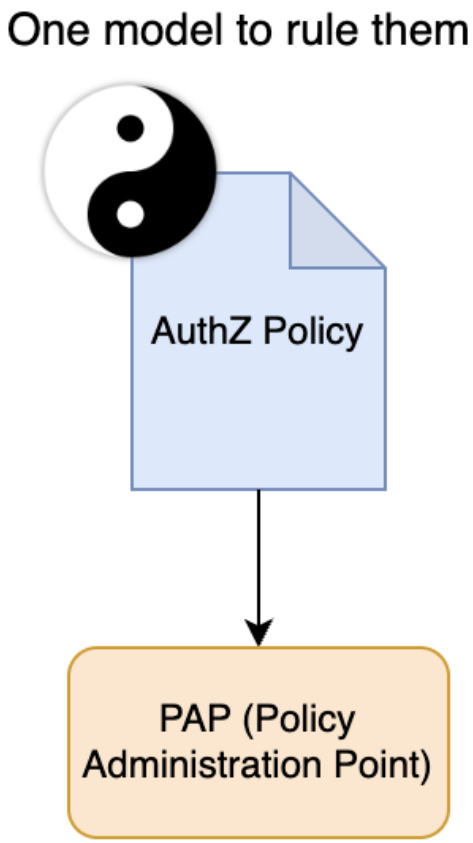
Our Goal

- Define a **unified, YANG-based framework** for expressing, updating, and sharing authorization policies — machine-readable, interoperable, and ready for automation



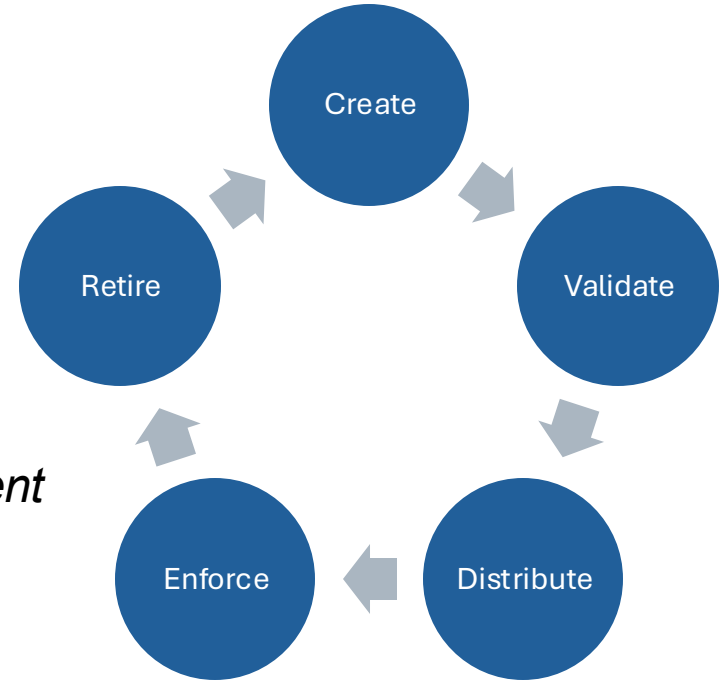
YANG-based Framework for Policy Sharing

- 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
- *Core idea:* YANG is the **source of truth**, and PaC provides the **enforcement logic**



Requirements for Distributed Policy Management

- **Granularity** → Define fine-grained authorization over *users, resources, and contextual conditions*
- **Context-awareness** → Evaluate dynamically based on *device state, network conditions, and environment*
- **Token alignment** → Ensure tokens include *required claims* for proper evaluation
- **Lifecycle control** → Support *creation, update, validation,* and *retirement* under consistent semantics
- **Interoperability** → Enable policy portability across *systems and administrative domains*



Policy-as-Code (PaC)

Treat policies as **declarative, versioned code artifacts** — enabling granular, context-aware authorization in distributed network environments

- **Dynamic adaptation:** policies can react to contextual changes
- **Fine-grained control:** express precise authorization logic beyond static rules or ACLs
- **Lifecycle assurance:** versioning and validation ensure consistent policy behavior across domains
- **Operational alignment:** integrates naturally with automated network operation workflows

Example in Rego syntax:

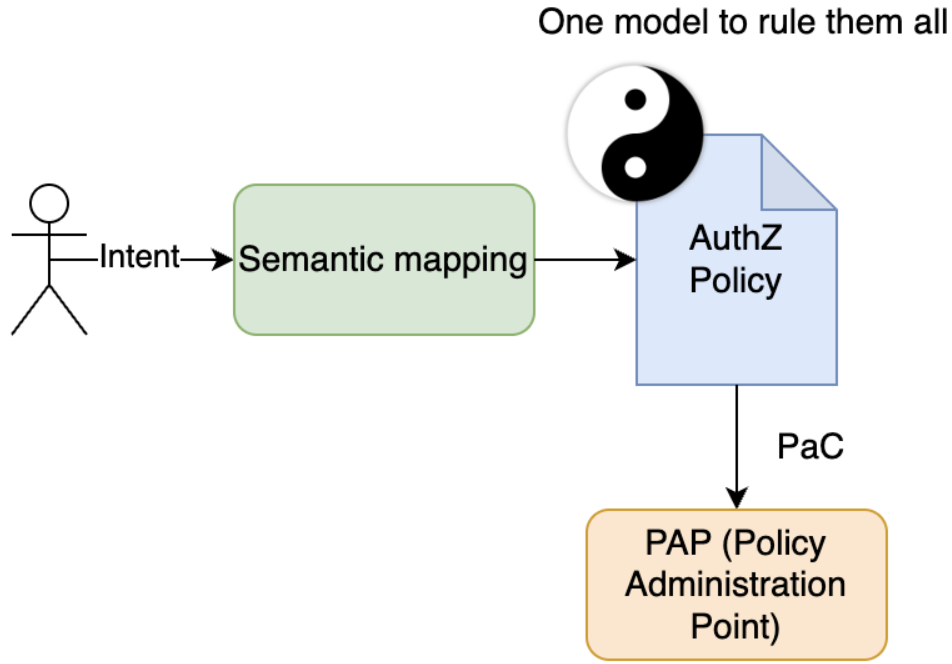
```
package example
# Allow read access if the user has the "read" role
default allow = false
allow {
    input.user.role == "read"
}
```

From Intents to Enforceable Policies

Intent-based networking (IBN) expresses *what the operator wants to achieve* —but not *how* that intent is realized or enforced

Our goal: **bridge that gap** by defining what an intent *must include* to be automatically transformed into a consistent, enforceable **PaC policy**

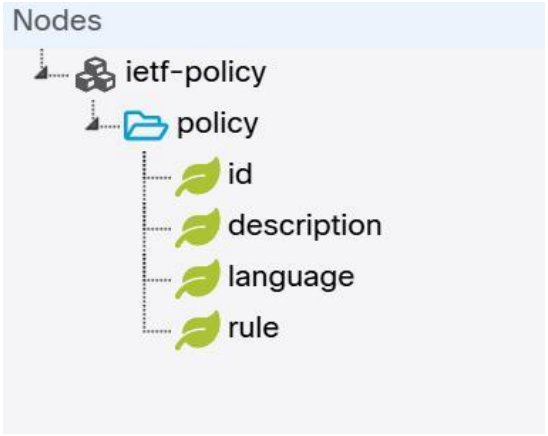
- **Define a shared semantic layer** — unify key terms: *actor*, *resource*, *action*, *context*
- **Translate intents → PaC policies**, maintaining linkage and traceability
- **Use a YANG-based model** to represent both:
  - the **intent metadata**
  - the **generated policy logic**



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

Current status

- First version of the draft *(draft-cabanillas-nmop-authz-policy-sharing-model-00)* — defining the base mechanisms for **distributed policy sharing** using **YANG + PaC**

Collaboration Points

- Feedback on **requirements** and **scope**
- Interest in **intent-based integration** (https://datatracker.ietf.org/doc/html/draft-pedro-ite-01)
- Discussion on **policy exchange** across domains