

Policy Language Binding for OAuth

Dapeng Liu

Alibaba Group

Suresh Krishnan

Cisco

Jeff Lombardo

AWS

Yaron Zehavi

Raiffeisen Bank International

Yaron Sheffer

Intuit

Should we need a Policy Language Binding for OAuth?

Dapeng Liu
Alibaba Group

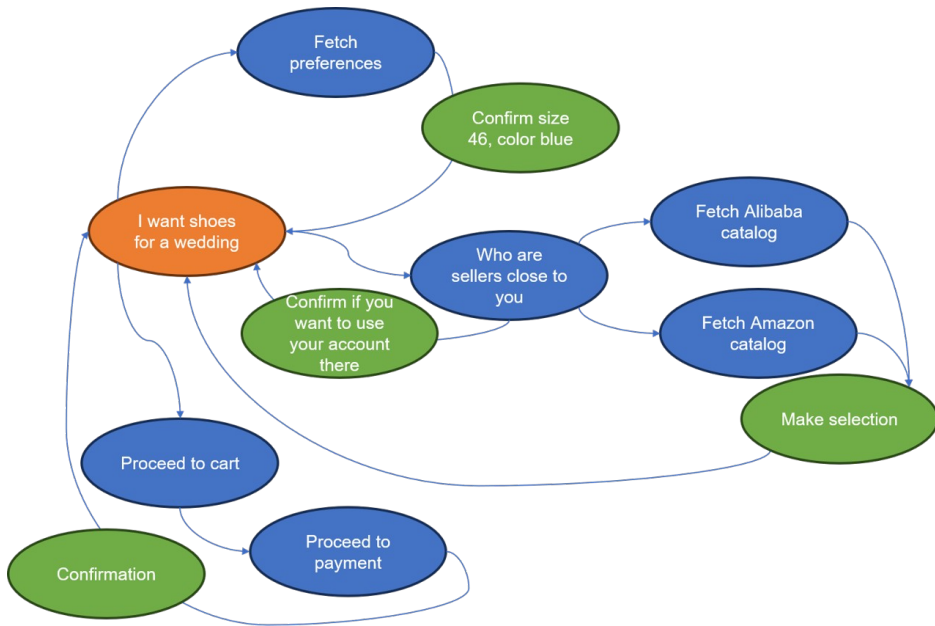
Suresh Krishnan
Cisco

Jeff Lombardo
AWS

Yaron Zehavi
Raiffeisen Bank International

Yaron Sheffer
Intuit

Problem Statement



- Workloads have evolved: from deterministic path to a probabilistic path
- Goal is decomposed in task carried over by multiple clients over multiple trust domains
- Each client must be bound to the fine grained task delegated to it so that RS can perform best access control
- Execution of a task can require negotiation between parties (Client, AS, RS)

5 success pillars for AI Agent Authorization

1

Expressing Goal

Initial objective
MUST act as a
guardrail
through
scoping
progression

2

Discovering Schema

How to
understand
scoping domain
specific schema
to support
progression

3

Formalizing Scoping Envelope

Formalizing
scoping in
requests and
issued
credentials

4

Signaling Back from RS

Reverse-guide
for
step-up scoping
when current
credential is
insufficient

5

Async Confirm

Asynchronously
obtaining
consent from
the user

Using Policy language bound in the access token

STEP 1 · REQUEST

Client constructs authorization request with policy

Client embeds a Rego policy in `authorization_details`. AS validates schema, does **not** evaluate policy.

```
authorization_details: {{
  "type": "authorization_policy",
  "policy_lang": "rego",
  "policy": "allow if input.amount
< 300 ...",
  "entrypoint": "agent_intent/allow"
}}
```

STEP 2 · TOKEN

AS issues token with policy bound

The access token carries the policy reference. This binds **specific behaviors** to this client authorization — not just a scope string.

```
// access token (JWT claims)
{
  "authorization_policy": {
    "policy_lang": "rego",
    "policy": "...",
    "entrypoint": "agent_intent/allow"
  }
}
```

STEP 3 · EVALUATE

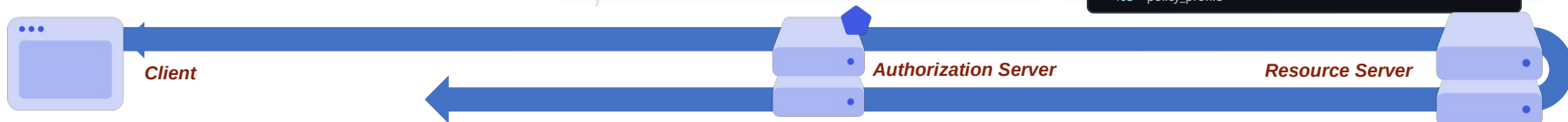
RS evaluates policy per-request

RS extracts policy from token, constructs input from runtime context, evaluates.

Behavior drift → **403 + policy_profile**.

```
// RS evaluates at runtime
// Request: add_to_cart, $250, 20:30
input = {amount:250, hour:20}
→ ALLOW

// Request: add_to_cart, $500
input = {amount:500}
→ 403 + policy_profile
```



STEP 6 · CONSTRUCT

Client builds the right authorization request

Policy follows the discovered schema. Client sends `authorization_details` with the correct profile.

```
POST /token
authorization_details={{
  "type": "authorization_policy",
  "policy_lang": "rego",
  "policy": "...",
  "entrypoint": "allow"
}}
```

STEP 5 · DISCOVER

Client reads metadata before constructing request

Client learns: "This RS accepts **rego-profile**, input schema at URI X." No custom integration needed.

```
// Agent discovers before acting
GET /.well-known/
  oauth-authorization-server

// learns: profiles=[rego,cedar]
// learns: schema_uri=...
// fetches input schema
```

STEP 4 · PUBLISH

RS/AS advertises supported policy profiles

Via `/well-known/oauth-authorization-server` and `RAR-metadata` schema.

```
"authorization_details_types_metadata": {{
  "type": "authorization_policy",
  "schema_uri": "https://as.example/
schemas/rego-profile.json",
  "profiles": ["rego", "cedar"]
}}
```

Using Policy language bound in the access token

Multiple attempts

Workgroup: OAuth Working Group
Internet-Draft: draft-ccchetti-oauth-rar-cedar-02
Published: 21 February 2024
Intended Status: Standards Track
Expires: 24 August 2024
Author: S. Cecchetti
Amazon

Cedar Profile for OAuth 2.0 Rich Authorization Requests

Abstract

This specification defines a profile of OAuth 2.0 Rich Authorization Requests in Cedar policy format within the `authorization_details` JSON object. Authorization servers and resource servers from different vendors can leverage this profile to distribute and receive relevant Cedar policy sets in an interoperable manner.

Workgroup: Web Authorization Protocol
Internet-Draft: draft-liu-oauth-rego-policy-00
Published: 9 June 2026
Intended Status: Standards Track
Expires: 11 December 2026
Authors: D. Liu, H. Zhu, S. Krishnan, A. Parecki, H. Xue
Alibaba Group, Alibaba Group, Cisco, Okta, Alibaba Group

Rego Policy Language for OAuth 2.0 Authorization

Abstract

AI agents exhibit dynamic, unpredictable behavior that cannot be fully described by traditional OAuth 2.0 scopes. This specification defines a behavioral authorization framework that enables clients, particularly AI agents, to propose Rego policy-based behavioral constraint contracts in OAuth 2.0 authorization flows using Rich Authorization Requests (RAR). It defines the `rego_policy` authorization data type for carrying behavioral constraint contracts in `authorization_details`, shifting the authorization model from static permission sets to runtime behavioral verification. It also defines a reverse-guided authorization mechanism allowing resource servers to return structured policy constraints in error responses, enabling agents to dynamically adapt their behavior and construct appropriate authorization requests.

Same outcomes

1

Policy generated client side cannot be trusted: presents policy poisoning and code injection risks
LLM are outside scope-of-trust, some language fail the default allow trap, untrusted URIs is called for Access Control poisoning

2

Policy language is a back-channel element, has a sense inside a trust domain
3rd party agents might not know how to speak it, while they already know they need to talk OAuth

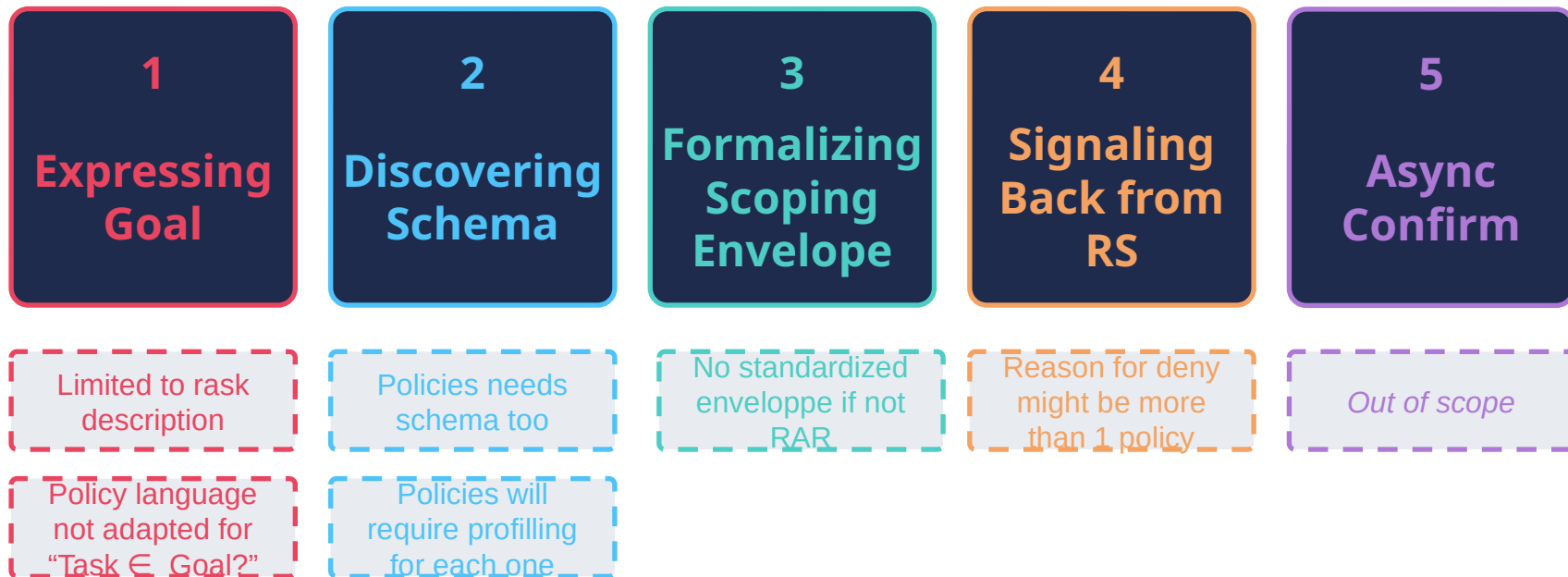
3

Client should only deliver context, not code
RAR/PAR/JAR should remain the mechanisms of choice, each proposal proves it

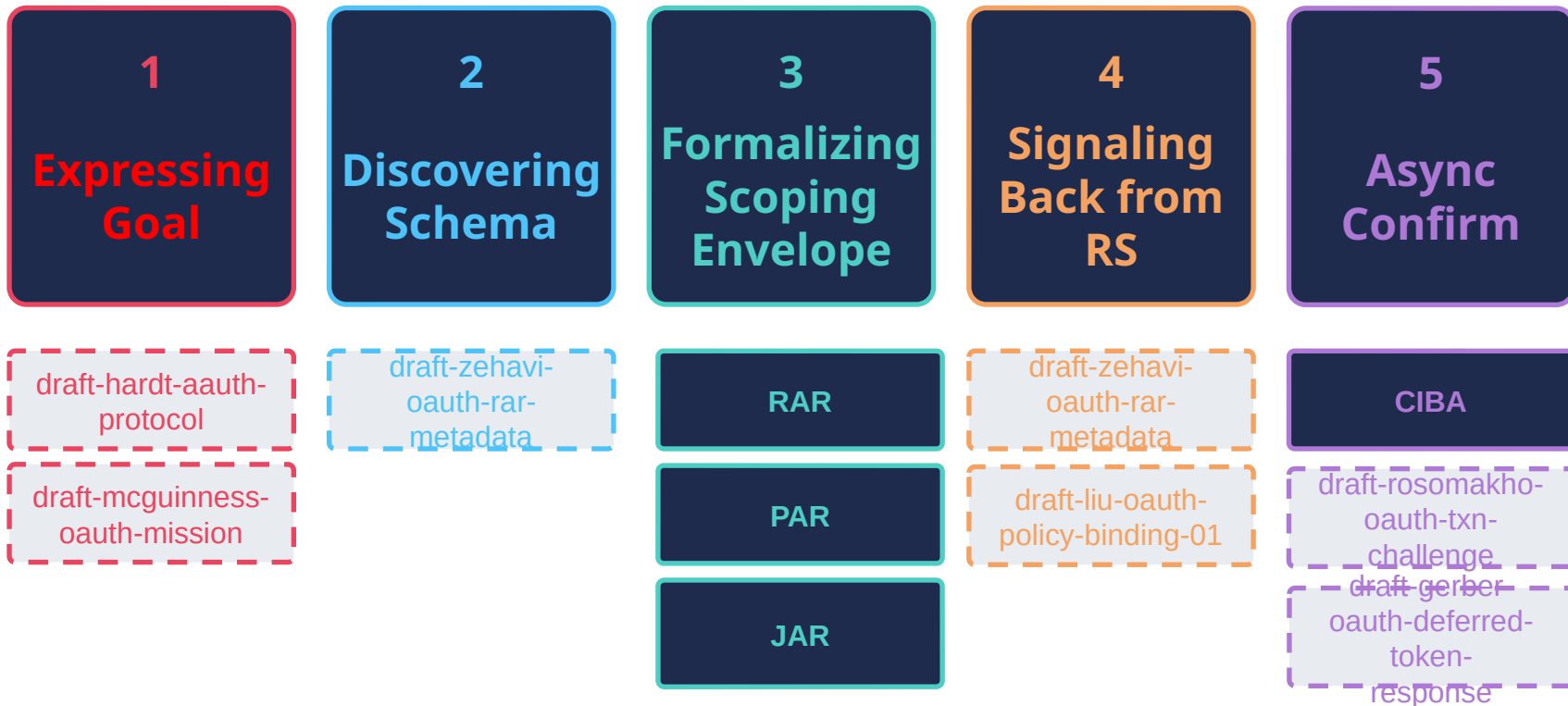
4

Tokens are not the right container for policies: conflict with PDP policies, can go stale

Policy Binding vs 5 success pillars



Existing OAuth vs 5 success pillars



Is There No Hope for Policy in OAuth?

Edge Case: DDIL Environments

Disconnected, Disrupted, Intermittent, Low-Bandwidth

- RS cannot reach PDP
- Policy in credential for local evaluation
- Potentially valid but highly constrained use case

Enriched Failure Feedback

For clients that can process policies details

- Must be limited to fragments
- Some policies could be sensitive
- Exposure could help RECON

Proposal to the WG

Form a design team to align Drafts to pillars

Objectives:

1. Enabling clients to construct fine-grained authorization requests using:
 - a. Goal state management,
 - b. Discoverable domain-specific schemas for task and authorization definition,
 - c. Task $\in$ Goal evaluation requests
2. Establishing a negotiation patterns that allows a resource server to guide clients towards required authorization

- **Out of scope**

- **Defining any domain-specific schema is out of scope**
- **Task $\in$ Goal Evaluation logic**

Thanks!