

oauth
Internet-Draft
Intended status: Standards Track
Expires: 24 October 2026

M. Chen
L. Su
China Mobile
22 April 2026

Policy, Lifecycle, and Intent Extensions for OAuth Rich Authorization
Requests
draft-chen-oauth-rar-agent-extensions-01

Abstract

OAuth 2.0 Rich Authorization Requests (RAR) [RFC9396] defines a syntax for clients to request fine-grained permissions. While powerful, this mechanism lacks standardized methods for clients to express three critical contextual dimensions prevalent in modern automated systems: the high-level user intent driving the request, the required security policy under which the request should be evaluated, and the binding of the authorization's lifecycle to an external process.

This document extends the RAR framework to address these gaps. It first defines a new `authorization_details` type, `*intent_request*`, to facilitate goal-oriented authorization. Second, it introduces two new parameters for the `authorization_details` object: `*policy_context*`, to request authorization under a specific assurance level, and `*lifecycle_binding*`, to tie the validity of the authorization grant to the state of an external entity. These extensions enable authorization servers to make more intelligent, secure, and context-aware decisions, particularly in ecosystems involving autonomous agents and complex, long-running tasks.

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 24 October 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	2
1.1. Terminology	4
2. Intent-Based Authorization: The intent_request Type	4
2.1. The intent_request Authorization Details Type	4
2.2. Object Structure	4
2.3. Authorization Flow and The Role of Token Exchange	5
3. Contextual Parameters for Authorization Details	5
3.1. The "policy_context" Parameter	5
3.1.1. Structure of the "policy_context" Object	6
3.2. The "lifecycle_binding" Parameter	6
3.2.1. Structure of the "lifecycle_binding" Object	6
4. Example Authorization Request	7
5. Authorization Server Metadata	8
6. Error Response	8
7. Security Considerations	8
8. IANA Considerations	9
8.1. OAuth Authorization Server Metadata	9
8.2. OAuth Extensions Error Registry	9
9. Acknowledgements	10
10. References	10
10.1. Normative References	10
10.2. Informative References	10
Authors' Addresses	10

1. Introduction

OAuth 2.0 Rich Authorization Requests [RFC9396] significantly enhances the expressive power of authorization requests beyond simple string-based scopes. This enables clients, such as sophisticated AI agents, to articulate their needs with greater precision.

The OAuth 2.0 Rich Authorization Requests (RAR) [RFC9396] framework provides a vital, standardized syntax for clients to request structured, fine-grained permissions. This moves beyond the limitations of simple string-based scopes. However, as automated systems and AI-driven agents become more prevalent, their authorization requirements introduce new complexities that the current RAR standard does not explicitly address. This document identifies three such challenges:

***Goal-Oriented Authorization*:** Complex clients, such as coordinator agents, often operate based on a high-level user goal (e.g., "book a trip") rather than a pre-defined set of API permissions. The initial authorization request needs a way to seek consent for the overall intent, which is subsequently decomposed into specific actions. This creates an "intent-to-permission gap" in the protocol.

***Policy Assurance Mismatch*:** A client might request a high-risk action (e.g., a payment), but the authorization server (AS) might evaluate this request against a default, low-assurance security policy (e.g., one not requiring multi-factor authentication). This ambiguity can be exploited in "policy downgrade" attacks. The protocol lacks a mechanism for a client to assert the required security posture for a given request.

***Over-privileged Token Lifetime*:** For long-running, automated tasks, access tokens with a fixed expiration time (exp claim) violate the principle of least privilege. If the task finishes early or is cancelled, the token remains valid, creating an unnecessary window of risk. The authorization grant's validity should ideally be coupled to the task's actual lifecycle.

To address these challenges, this document extends RAR in two primary ways:

It introduces a new `authorization_details` type, `intent_request`, to allow clients to request authorization for a high-level goal.

It introduces two new parameters, `policy_context` and `lifecycle_binding`, which can be included within any `authorization_details` object to provide essential context about security requirements and validity constraints.

This specification addresses these gaps by extending the RAR framework to include explicit policy context and lifecycle binding information within the authorization request itself.

1.1. Terminology

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].

This document uses the terms "Authorization Server" (AS), "Client", "Resource Server" (RS), and "Access Token" as defined in The OAuth 2.0 Authorization Framework [RFC6749]. The term "authorization_details" is used as defined in Rich Authorization Requests [RFC9396].

2. Intent-Based Authorization: The intent_request Type

2.1. The intent_request Authorization Details Type

This specification defines a new value, `*intent_request*`, for the type member of the `authorization_details` object. This type allows a client to seek authorization for a high-level user goal, which the AS can then use as a basis for issuing more specific, fine-grained authorizations at a later stage.

2.2. Object Structure

When the type member has a value of `intent_request`, the object has the following members:

- * `intent` (string, REQUIRED): A human-readable, textual description of the user's goal. This value is intended for display on user consent screens and for auditing purposes.
- * `constraints` (object, OPTIONAL): A JSON object containing structured, machine-readable parameters that define the boundaries of the intent. The structure and semantics of these parameters are specific to the intent and are to be defined by the ecosystem or the AS.

for example

```
{
  "type": "intent_request",
  "intent": "Plan and book my business trip to the IETF meeting in Paris.",
  "constraints": {
    "max_budget": "2000",
    "currency": "EUR",
    "travel_class": "economy"
  }
}
```

2.3. Authorization Flow and The Role of Token Exchange

The `intent_request` flow is typically a two-stage process:

***Initial Intent Authorization*:** The client makes an authorization request including an `authorization_details` object of type `intent_request`. The AS authenticates the user and presents the intent and a summary of the constraints for consent. If the user approves, the AS issues an authorization grant (and subsequently, an access token) that represents the approved intent. This initial "intent token" MAY NOT be audience-restricted to any specific resource server and may have limited usability on its own.

***Just-in-Time Permission Issuance*:** As the client decomposes the intent into concrete sub-tasks, it uses the "intent token" to request task-specific access tokens. This is accomplished via the Token Exchange grant type [RFC8693]. The client presents the "intent token" as the `subject_token` and includes a new `authorization_details` parameter in the request, specifying the fine-grained permissions needed for the sub-task. The AS is responsible for validating that the requested permissions are a legitimate and safe decomposition of the originally consented intent.

3. Contextual Parameters for Authorization Details

This specification defines two new JSON object members, `policy_context` and `lifecycle_binding`, that can be included in any `authorization_details` object, including `intent_request`.

3.1. The "policy_context" Parameter

This specification defines a new JSON object member, "policy_context", for the "authorization_details" object. This member allows a client to request that the authorization decision be evaluated under a specific set of policy constraints and assurance levels.

3.1.1. Structure of the "policy_context" Object

The "policy_context" member, A JSON object containing members that provide context for the policy decision.

- * assurance_level (string): An identifier for a desired policy assurance level (e.g., "financial_grade_v1"), which MUST be a value supported and advertised by the AS.
- * compliance_frameworks (array of strings): Identifiers for specific compliance frameworks (e.g., "gdpr") that the operation must adhere to.

```
{
  "assurance_level": "financial_grade_v1",
  "compliance_frameworks": ["pci-dss", "gdpr", "iso27001"]
}
```

AS Processing: When present, the AS MUST validate the requested context against its supported policies. It SHOULD present the implications of the policy (e.g., "This requires multi-factor authentication") to the user during consent. If granted, the AS SHOULD include the validated policy_context in the authorization_details claim of the resulting JWT access token.

3.2. The "lifecycle_binding" Parameter

This specification defines another new JSON object member, "lifecycle_binding", for the "authorization_details" object. This member ties the authorization's validity to the lifecycle of an external entity.

3.2.1. Structure of the "lifecycle_binding" Object

The "lifecycle_binding" member, A JSON object defining the binding mechanism.

```
{
  "type": "task_status_webhook",
  "task_id": "job-d4a3b2-8c7e-4f5a-9b1c-2d3e4f5a6b7c",
  "termination_states": ["COMPLETED", "FAILED", "CANCELLED"]
}
```

The "type" member indicates the mechanism by which the AS can monitor the external entity's state. The "task_status_webhook" type indicates that the external task provider will notify the AS of terminal state changes via a pre-configured webhook. The object MUST also contain:

- * "task_id" (string): A unique identifier for the task.
- * "termination_states" (array of strings): A list of states that, once the task enters, will trigger the immediate revocation of the associated authorization.

AS Processing: The AS MUST be capable of monitoring the entity's state via the specified mechanism. It MUST record the link between the task_id and the issued authorization grant. Upon notification of a terminal state, the AS MUST immediately revoke the grant and all associated tokens.

Resource Server Responsibilities: An RS receiving a token with a lifecycle_binding claim MUST NOT rely solely on the exp claim. It MUST use a real-time validation method, such as token introspection [RFC7662] or a revocation feed, to ensure the token has not been revoked due to a lifecycle event.

4. Example Authorization Request

The following example demonstrates a coordinator agent requesting authorization for a user's travel planning intent, utilizing all three extensions.

```
"authorization_details": [  
  {  
    "type": "intent_request",  
    "intent": "Plan and book my business trip to the IETF meeting in Paris.",  
    "constraints": {  
      "max_budget": "2000",  
      "currency": "EUR"  
    },  
    "policy_context": {  
      "assurance_level": "financial_transaction_v2",  
      "compliance_frameworks": ["gdpr"]  
    },  
    "lifecycle_binding": {  
      "type": "task_status_webhook",  
      "task_id": "trip-planning-job-a1b2c3",  
      "termination_states": ["COMPLETED", "CANCELLED_BY_USER"]  
    }  
  }  
]
```

- * intent_request: The client asks for approval of the high-level goal.

- * `policy_context`: It asserts that all subsequent actions derived from this intent (like payments) must be processed under a high-assurance financial policy.
- * `lifecycle_binding`: It ensures that once the planning task is complete or cancelled, all associated permissions are automatically revoked.

5. Authorization Server Metadata

This specification adds the following parameters to the OAuth Authorization Server Metadata [RFC8414].

1.`policy_assurance_levels_supported`: OPTIONAL. A JSON array of objects, where each object represents a supported assurance level. Each object has the following members:

- * `"level"` (string): A structured identifier for the assurance level (e.g., `"nist_ial2"`). REQUIRED.
- * `"description"` (string): A human-readable description. OPTIONAL.
- * `"uri"` (string): A URI pointing to the detailed definition of the level. OPTIONAL.

2.`policy_compliance_frameworks_supported`: OPTIONAL. A JSON array of strings containing the compliance framework identifiers supported by the AS.

6. Error Response

When a request is denied due to a failure in validating the `"policy_context"`, the AS returns an error response with the following error code:

Error Code: `*policy_requirement_not_met*` Description: The requested `"policy_context"` is not supported or cannot be satisfied. The AS SHOULD include an `"error_description"` parameter to provide developers with more details about the failure.

7. Security Considerations

Intent-to-Permission Escalation: The `intent_request` type shifts significant responsibility to the AS's policy engine. A primary threat is a client using a broadly-phrased intent to acquire excessive or unrelated permissions via subsequent token exchanges. The AS MUST implement a strict and carefully designed policy engine to govern this decomposition. The constraints provided in the

initial request MUST be treated as strict, non-negotiable boundaries.

***Preventing Policy Downgrade Attacks*:** The "policy_context" member allows the AS to enforce that high-risk actions are always requested with a corresponding high-assurance policy context. The AS MUST reject requests where this mapping is violated, thus preventing a client from obtaining a privileged token under weak security pretenses.

***Ensuring Timely Revocation*:** The "lifecycle_binding" member significantly reduces the risk window of compromised tokens for long-running tasks. The effectiveness of this mechanism is entirely dependent on the reliability of the state notification and revocation propagation systems. Implementations MUST include robust error handling and fallbacks. The token's "exp" claim serves as an essential fallback mechanism.

***User Consent and Transparency*:** The structured nature of these new members allows the AS to present a far more intelligible and accurate consent screen to the user, leading to more meaningful and informed authorization decisions.

***Reliability of External State Monitoring*:** The AS must trust the source of the lifecycle events (e.g., the task provider's webhook). Mechanisms such as mutual TLS authentication and request signing SHOULD be used to secure the communication channel for state updates.

8. IANA Considerations

8.1. OAuth Authorization Server Metadata

This specification requests the registration of the following values in the "OAuth Authorization Server Metadata" registry:

- * policy_assurance_levels_supported
- * policy_compliance_frameworks_supported

8.2. OAuth Extensions Error Registry

This specification requests the registration of the following value in the "OAuth Extensions Error Registry":

- * Error Code: policy_requirement_not_met

9. Acknowledgements

This document based on RFC9396

10. References

10.1. Normative References

- [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>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/rfc/rfc6749>>.
- [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>>.
- [RFC9396] Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, <<https://www.rfc-editor.org/rfc/rfc9396>>.
- [RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <<https://www.rfc-editor.org/rfc/rfc8414>>.

10.2. Informative References

- [RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <<https://www.rfc-editor.org/rfc/rfc7662>>.

Authors' Addresses

Meiling Chen
China Mobile
BeiJing
China
Email: chenmeiling@chinamobile.com

Li Su
China Mobile
BeiJing
China
Email: suli@chinamobile.com