oauth
Internet-Draft
Intended status: Standards Track
Expires: 8 August 2026

M. Chen
L. Su
China Mobile

4 February 2026

Policy and Lifecycle Extensions for OAuth Rich Authorization Requests
draft-chen-oauth-rar-agent-extensions-00

Abstract

OAuth 2.0 Rich Authorization Requests (RAR), as defined in RFC 9396, provides a mechanism for clients to request fine-grained, structured authorization details. However, in emerging ecosystems of collaborating AI agents and long-running automated tasks, two critical authorization dimensions remain implicit: the required security assurance level of the authorization policy and the strict binding of authorization lifetime to a task’s lifecycle.

This document extends the "authorization_details" object of RAR by introducing two new members: "policy_context" and "lifecycle_binding". The "policy_context" member allows a client to explicitly request that the authorization be evaluated and granted under a specific policy assurance level, mitigating policy downgrade attacks and enhancing user consent. The "lifecycle_binding" member enables the validity of an authorization grant to be tied directly to the state of an external entity, such as an automated task, ensuring permissions are automatically and immediately revoked upon task completion. These extensions provide a standardized way to achieve more secure, transparent, and context-aware authorization in complex, automated environments.

Status of This Memo

Table of Contents

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.

1.1. The Need for Advanced Authorization in AI Agent Ecosystems

Modern distributed systems are increasingly composed of autonomous or semi-autonomous AI agents that collaborate to fulfill complex user intents. For example, a user’s request to "plan a business trip to Paris" might involve a primary "coordinator" agent that delegates sub-tasks to specialized agents for flight booking, hotel reservation, and expense reporting.

In such scenarios, traditional authorization models face two major challenges:

*Permission Abuse and Policy Downgrade: An agent might request a high-privilege action (e.g., booking a flight) but the authorization server (AS) might grant it under a default, low- assurance security policy. A malicious or compromised client could exploit this ambiguity to bypass stricter security controls like multi-factor authentication or fraud detection that should be associated with high-risk operations. The authorization request lacks a standard mechanism for the client to explicitly request and for the user to consent to a specific, appropriate level of policy assurance.

*Excessive Permission Lifetime: An agent is granted permission to manage a long-running task (e.g., a data processing job). Traditional access tokens rely on a fixed expiration time ("exp" claim). If the task completes or fails prematurely, the token remains valid for the remainder of its lifetime, creating a significant window of opportunity for misuse if the token is compromised. The principle of least privilege dictates that the permission’s lifetime should be strictly bound to the task’s active lifecycle.

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

1.2. 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. The "policy_context" Member

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 subject to a specific set of policy constraints and assurance levels.

2.1. Structure of the "policy_context" Object

The "policy_context" member, if present, MUST be a JSON object. Its members provide context for the policy decision.

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

2.2. Pre-defined Policy Context Parameters

2.2.1. The "assurance_level" Parameter

A string that identifies the desired policy assurance level. This identifier MUST be from a list of values pre-defined and published by the authorization server. It serves as the primary input for the AS to select the appropriate set of evaluation rules (e.g., requiring MFA, performing risk analysis).This identifier MUST be one of the values advertised by the Authorization Server in its "policy_assurance_levels_supported" metadata entry (see Section 5).

2.2.2. The "compliance_frameworks" Parameter

An optional array of strings identifying specific compliance frameworks that the requested operation must adhere to. These string identifiers MUST be values from the "OAuth Policy Compliance Frameworks" registry established by this specification (see Section 10.3.2).

2.3. Authorization Server Processing

When an AS receives an "authorization_details" object containing a "policy_context" member, it MUST:

1. Validate that the value of "assurance_level" is known and supported.

2. Use the "policy_context" to select and execute the corresponding authorization policies.

3. Perform a cross-validation to ensure the requested policy context is appropriate for the requested "type" and "actions". For instance, an AS policy may require that a "payment_transaction" type MUST request a "financial_grade_v1" or higher assurance level.

4. Clearly communicate the semantic meaning of the requested "policy_context" to the end-user during the consent process.

2.4. Issuance in Access Tokens

If the request is approved, the AS SHOULD include the validated "policy_context" object within the "authorization_details" claim of the issued JWT access token. This enables the resource server to perform its own local policy enforcement based on this context.

3. The "lifecycle_binding" Member

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.

3.1. Structure of the "lifecycle_binding" Object

The "lifecycle_binding" member, if present, MUST be a JSON object.

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

3.2. Lifecycle Binding Types

The "type" member indicates the mechanism by which the AS can monitor the external entity’s state.

3.2.1. 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:

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

3.3. Authorization Server Processing

When an AS processes a request with a "lifecycle_binding" member, it MUST:

1. Verify its capability to monitor the specified task.

2. Create an internal record linking the "task_id" to the access token to be issued (e.g., via its "jti").

3. Upon receiving notification that the task has entered a terminal state, the AS MUST immediately revoke the associated token and propagate this revocation information.

3.4. Resource Server Responsibilities

Resource servers receiving tokens that contain a "lifecycle_binding" claim SHOULD NOT rely solely on the "exp" claim for validation. They MUST employ a mechanism to check for real-time revocation, such as subscribing to a revocation feed from the AS or using token introspection [RFC7662].

4. Example Authorization Request

The following is a non-normative example of an "authorization_details" object utilizing both new members for a request to execute a sensitive data processing task.

[{ "type": "patient_data_analysis_job", "actions": ["execute", "monitor_progress"], "locations": ["https://api.compute.com/jobs"], "policy_context": { "assurance_level": "hipaa_phi_access", "compliance_frameworks": ["hipaa", "gdpr"] }, "lifecycle_binding": { "type": "task_status_webhook", "task_id": "analysis-job-1138", "termination_states": ["COMPLETED", "FAILED_VALIDATION"] } }]

5. Authorization Server Metadata

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

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.

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

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.

The "policy_context" and "lifecycle_binding" members may contain information about the user’s intended actions and associated tasks. This information must be handled with care by the AS and RS, subject to the system’s overall privacy policy.

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