TBD
Internet-Draft
Intended status: Standards Track
Expires: 20 November 2026

J. Kahrer
Curity

19 May 2026

OAuth Client Challenge Protocol
draft-kahrer-oauth-client-challenge-protocol-00

Abstract

This document extends the OAuth 2.0 token endpoint error response (RFC 6749) with a new error code that indicates to the client that it must provide additional input for the authorization server to authorize it and accept its request.

This mechanism enables just-in-time authorization flows in which the authorization server dynamically challenges the client during a request and the client tries to satisfy it. For example, the authorization server can ask the client mid-flow to provide an assertion, a verifiable presentation, or other proof-of-possession material.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-kahrer-oauth-client-challenge-protocol/.

Source for this draft and an issue tracker can be found at https://github.com/curityio/ietf-draft-oauth-client-challenge-protocol.

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/.

Copyright Notice

Table of Contents

1. Introduction

The OAuth 2.0 Authorization Framework [RFC6749] assumes that a single grant signaled through the grant_type parameter is sufficient for the authorization server to authorize the client. It does not define how the authorization server signals to the client to provide additional input for it to make a decision - like an additional grant from the Resource Owner or an attestation artifact to prove the client's provenance.

Real-world deployments increasingly require the authorization server to apply dynamic, contextual authorization policies — for example:

* Demanding a freshly signed client attestation when risk signals indicate an elevated threat level.

* Requiring the client to prove its mandate before a high-value token is issued.

This document introduces a way for the authorization server to start a challenge session where it defines requirements that it expects from the client. It extends the OAuth 2.0 Authorization Framework by introducing the following parameters:

1. A new error code insufficient_client_authorization for token error responses that the authorization server returns when it cannot proceed without additional client-supplied material.

2. A companion response parameter authorization_requirement — a typed JSON object that describes what the authorization server requires.

3. Processing rules for both parties, including the requirement to return unauthorized_client when subsequently provided input fails validation.

1.1. Motivation and Use Cases

1.1.1. Just-in-Time Authorization

Whether and in which form an authorization server may return an access token to a client depends not only on the presented grant and requested access but also on the capabilities of the client, e.g., whether it is a public client or a confidential client, whether it is a first-party client or a third-party client. Just-in-time authorization means that the authorization server can, dynamically

and on demand, evaluate necessary data from different sources to understand the context of a request and use that information to accept or deny a token request from a client.

In a just-in-time flow, the authorization server defers its final authorization decision, challenges the client for supplemental proof material, and only then either grants or denies the request.

1.1.2. Client Authentication Step-Up

An authorization server may accept client authentication for low-assurance tokens but require an attestation for tokens granting elevated privileges (see ([I-D.ietf-oauth-attestation-based-client-auth])). The insufficient_client_authorization mechanism allows the authorization server to escalate the authentication requirement without the client needing to speculatively include high-assurance credentials on every request. It allows, for example, public clients to increase their trust profile.

1.1.3. Continuos Authorization

The security state of a system can change at any time. Systems may communicate via signals about certain security-relevant events that they observed for other systems to adopt. In such an environment the authorization server may receive signals that constitute the need for additional input for authorization to e.g., mitigate attacks and prevent misuse.

1.2. Comparison with OAuth 2.0 First-Party Applications

OAuth 2.0 First-Party Applications [I-D.ietf-oauth-first-party-apps] defines an API for user authentication where the authorization server challenges the OAuth 2.0 client to provide data from the user. The proposed API is similar to the mechanism defined in this document. However, there is a subtle difference: OAuth 2.0 First-Party Applications defines a new error code for the client to provide more data from the end-user (Resource Owner). Its main purpose is to enable clients to control the user experience. For that it makes two important assumptions:

* The client can interact with an end-user.

* The client is trusted to handle sensitive data like the end-user’s credentials, i.e., the client is a first-party application.

The extension in this document is different because it assumes that the client can satisfy the challenge from the Authorization Requirement itself. It is applicable for both first- and third-party use cases where the authorization server challenges the client to provide additional input for the authorization grant. The client does not have to handle end-user credentials. What's more, it does not require an additional endpoint but extends the token response.

2. Conventions and Definitions

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] when, and only when, they appear in all capitals, as shown here.

3. Terminology

This document uses the terms "access token", "authorization server", "client", "client authentication", "token response", "token request" and "token endpoint" as defined by the OAuth 2.0 Authorization Framework [RFC6749], unless otherwise specified by this document.

In addition, the document uses the following terms:

Insufficient Client Authorization Response: A token error response from the OAuth 2.0 token endpoint that indicates to the client that the authorization server requires additional input for it to authorize the client.

Authorization Requirement: A typed JSON object returned by the authorization server in an Insufficient Client Authorization Response that specifies the additional material the client must supply.

Challenge Session: A string managed by the authorization server that serves as the nonce in the challenge-response pattern. It associates an Insufficient Client Authorization Response with the subsequent request that satisfies it.

4. Insufficient Client Authorization Response

This document registers the error code insufficient_client_authorization for use in OAuth 2.0 token endpoint error responses as defined in Section 5.2 of [RFC6749].

The following content applies to the Insufficient Client Authorization Response.

error: REQUIRED. The error parameter MUST be insufficient_client_authorization.

authorization_requirement: REQUIRED. The authorization_requirement parameter is a typed JSON object as defined in Section 5.

challenge_session: REQUIRED. An opaque identifier generated by the authorization server that binds this authorization challenge to the follow-up request.

The client MUST include this value in the subsequent request to the authorization server if it receives one along with the insufficient_client_authorization error response.

expires_in: OPTIONAL. A JSON number that defines the number of seconds from the time of the Insufficient Client Authorization Response until the challenge_session expires.

The client MUST NOT submit a response after this time.

The authorization server MUST comply with Section 5.2 of [RFC6749]. The authorization server SHOULD respond with HTTP status code 403 (Forbidden). It MAY include other parameters in the response. The client MUST ignore any parameters it does not understand.

If the client does not understand or cannot satisfy the Authorization Requirement, it MUST treat the Insufficient Client Authorization Response as if the authorization server returned an unauthorized_client error.

The Insufficient Client Authorization Response indicates the following:

1. The authorization server has determined that it cannot yet authorize the client or issue an access token.

2. The condition is resolvable: the authorization server knows what additional input would allow it to proceed.

3. The authorization server wishes to challenge the client to supply that input.

The following represents a non-normative example of an Insufficient Client Authorization Response.

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Cache-Control: no-store

{
  "error": "insufficient_client_authorization",
  "authorization_requirement": {
    "type": "verifiable_presentation",
    "challenge_session": "7f3d9e2a-4c1b-4f8e-b5a0-1e6c8d2f0a9b",
    "presentation_definition": { ... }
  }
}
```

5. The authorization_requirement Object

The authorization_requirement parameter holds a JSON object that indicates what type of input the authorization server requires for the client to satisfy the insufficient client authorization.

The following member is defined for all authorization_requirement types:

type: REQUIRED. An absolute URI or a string identifying the authorization requirement type. The value determines the semantics of other members in the object.

6. Providing Authorization Requirement

Each profile of this document that specifies a type of Authorization Requirement also MUST define how the client can fulfill the challenge and provide the required input to the authorization server.

If the client does not understand the type of the authorization_requirement of an Insufficient Client Authorization Response or if it cannot satisfy the requirement, the client MUST treat the Insufficient Client Authorization Response as if the authorization server returned an unauthorized_client error as defined in Section 5.2 in [RFC6749], see also Section 4.

The client MUST include the challenge_session parameter with the same value as it received in the Insufficient Client Authorization Response in its subsequent request and attempt to resolve the challenge.

Some extensions to OAuth 2.0, notably Pushed Authorization Requests [RFC9126], make use of the token endpoint response outside a token endpoint request. A profile that defines an Authorization Requirement type SHOULD define mechanisms to fulfill the requirements that are applicable to authorization and token requests alike.

If the authorization server deems the supplied input from the client in response to an Authorization Requirement challenge as invalid and if there is no other way for the client to resolve the insufficient_authorization error, the authorization server MUST respond with an unauthorized_client error.

7. Client Metadata

Clients that communicate their client metadata, for example via dynamic client registration [RFC7591] or client ID metadata document [I-D.ietf-oauth-client-id-metadata-document] can signal support for this specification. To do so, they MUST include the following property in their client metadata:

insufficient_client_authorization_supported: OPTIONAL. JSON boolean value specifying whether the client supports this specification.

This enables authorization servers to apply this specification without breaking integrations.

8. Security Considerations

8.1. Sender-Constrained Tokens

The authorization server SHOULD issue sender-constrained tokens as a result of a successfully resolved client authorization challenge. In this way, the authorization server can mitigate the risk of token theft and replay. The idea is that the authorization server binds the token to a public key that the client controls. For the token to be valid, the client MUST provide a proof-of-possession that satisfies that key binding. The exact methods on how the authorization server binds the token to the client’s key and how the client provides a proof-of-possession are out of scope of this document. Demonstrating Proof-of-Possession (DPoP) [RFC9449] and certificate-bound access tokens [RFC8705] are two examples, and there may be others.

8.2. Challenge Session

The challenge_session parameter associates a Insufficient Client Authorization Response with follow-up resolution attempts. To mitigate session hijacking and replay, the authorization server SHOULD bind the challenge_session to the device requesting tokens, for example via DPoP. Similar to sender-constrained tokens, the binding prevents other devices from replaying a captured challenge_session and thus prevents other devices from taking over sessions (session hijacking).

9. IANA Considerations

9.1. OAuth Parameters Registration

This document defines the following values for the IANA "OAuth Parameters" registry of [IANA.oauth-parameters] established by [RFC6749].

Parameter Name: challenge_session

Parameter Usage Location: token response, token request

Change Controller: IETF

Specification Document: Section 4 of [draft-kahrer-oauth-client-challenge-protocol-00]

9.2. OAuth Error Codes

This document defines the following values for the IANA "OAuth Extensions Error" registry of [IANA.oauth-parameters].

Error Name: insufficient_client_authorization

Error Usage Location: token endpoint

Related Protocol Extension: n/a

Change Controller: IETF

Specification Document: Section 4 of [draft-kahrer-oauth-client-challenge-protocol-00]

9.3. OAuth Client Metadata Registration

This document defines the following values for the IANA "OAuth Dynamic Client Registration Metadata" registry of [IANA.oauth-parameters] established by [RFC7591].

Client Metadata Name: insufficient_client_authorization_supported

Client Metadata Description: JSON boolean value that specifies whether the OAuth client supports insufficient client authorization error in token error responses.

Change Controller: IESG

Specification Document: Section 7 of [draft-kahrer-oauth-client-challenge-protocol-00]

10. References

10.1. Normative References

[IANA.oauth-parameters]
IANA, "OAuth Parameters", <https://www.iana.org/assignments/oauth-parameters>.

[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>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <https://www.rfc-editor.org/rfc/rfc7591>.

[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>.

10.2. Informative References

[I-D.ietf-oauth-attestation-based-client-auth]
Looker, T., Bastian, P., and C. Bormann, "OAuth 2.0 Attestation-Based Client Authentication", Work in

Progress, Internet-Draft, draft-ietf-oauth-attestation-based-client-auth-08, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-attestation-based-client-auth-08>.

[I-D.ietf-oauth-client-id-metadata-document]
Parecki, A. and E. Smith, "OAuth Client ID Metadata Document", Work in Progress, Internet-Draft, draft-ietf-oauth-client-id-metadata-document-01, 1 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-client-id-metadata-document-01>.

[I-D.ietf-oauth-first-party-apps]
Parecki, A., Fletcher, G., and P. Kasselman, "OAuth 2.0 for First-Party Applications", Work in Progress, Internet-Draft, draft-ietf-oauth-first-party-apps-03, 27 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-first-party-apps-03>.

[Mastercard.VI]
Group, V. I. W., "Verifiable Intent (VI) — Specification Overview", February 2026, <https://verifiableintent.dev/spec/>.

[OpenID.Native-SSO]
Fletcher, G., "OpenID Connect Native SSO for Mobile Apps", November 2022, <https://openid.net/specs/openid-connect-native-sso-1_0.html>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <https://www.rfc-editor.org/rfc/rfc8705>.

[RFC9126] Lodderstedt, T., Campbell, B., Sakimura, N., Tonge, D., and F. Skokan, "OAuth 2.0 Pushed Authorization Requests", RFC 9126, DOI 10.17487/RFC9126, September 2021, <https://www.rfc-editor.org/rfc/rfc9126>.

[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>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

Appendix A. Example with Authorization Details

The following is an example that illustrates a token request using authorization_details as defined in [RFC9396]. The authorization details indicate that the client aims to operate in an open banking ecosystem that has certain requirements.

```
POST /token HTTP/1.1
Host: authorization-server.example
Content-Type: application/x-www-form-urlencoded
OAuth-Client-Attestation: eyJ0eXAiOiJvYX...
DPoP: eyJ0eXAiOiJkcG9...

grant_type=client_credentials&
resource=https://api.openbanking.example&
authorization_details=%5B%7B%22type%22%3A%22payment_initiation%22%2C%22actions%22%3A%5B%2
2initiate%22%2C%22status%22%2C%22cancel%22%5D%2C%22locations%22%3A%5B%22https%3A%2F%2Fapi
.openbanking.example%2Fpayments%22%5D%2C%22instructedAmount%22%3A%7B%22currency%22%3A%22E
UR%22%2C%22amount%22%3A%22123.50%22%7D%2C%22creditorName%22%3A%22Merchant%20A%22%2C%22cre
ditorAccount%22%3A%7B%22iban%22%3A%22DE02100100109307118603%22%7D%2C%22remittanceInformat
ionUnstructured%22%3A%22Ref%20Number%20Merchant%22%7D%5D
```

While the client authentication already provides some information about the provenance of the running software, the client also needs to prove that it complies with the requirements. Therefore, the authorization server challenges the client and responds with a Insufficient Client Authorization Response. The authorization server requests from the client the presentation of a verifiable credentials, e.g., a verifiable intent [Mastercard.VI].

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Cache-Control: no-store

{
  "error": "insufficient_client_authorization",
  "authorization_requirement": {
    "type": "verifiable_presentation",
    "challenge_session": "7f3d9e2a-4c1b-4f8e-b5a0-1e6c8d2f0a9b",
    "presentation_definition": { ... }
  }
}
```

Acknowledgments

The author wants to thank the following people for their feedback, input and contributions to this document:

Jacob Ideskog, Micha Trojanowski

Document History

-00

* Initial draft

Author’s Address

Judith Kahrer
Curity
Email: judith.kahrer@curity.io