Web Authorization Protocol
Internet-Draft
Intended status: Informational
Expires: 2 January 2026

N. Watson
Google, LLC

1 July 2025

Rich OAuth Error Responses
draft-watson-oauth-rich-error-response-00

Abstract

Define an error-handling protocol extension for the OAuth 2.0 token endpoint that allows the authorization server or resource server to specify an extra parameter on error responses that should be passed through to follow-up authorization requests.

About This Document

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

The latest revision of this draft can be found at https://njwatson32.github.io/oauth-error/draft-watson-oauth-rich-error-response.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-watson-oauth-rich-error-response/.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (mailto:oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/. Subscribe at https://www.ietf.org/mailman/listinfo/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/njwatson32/oauth-error.

Status of This Memo

This Internet-Draft will expire on 2 January 2026.

Table of Contents

1. Introduction

OAuth 2.0 [RFC6749] and [RFC6750] define several different error codes that authorization servers and resource servers may return. On token endpoint calls, most most runtime errors are lumped into invalid_grant, with the rest of the errors indicating a coding or configuration error by the client. Similarly, invalid_token covers most runtime errors on resource server calls.

Given the changing security, privacy, and regulatory landscapes since OAuth 2.0 was released, many authorization servers have new types of error conditions, which may be too complex to adequately describe in a JSON response.

This specification extends OAuth to allow for authorization servers and resource servers to return an extra parameter which can be passed back to the authorization server when the user is present in order to provide a richer error experience which may even allow for recovery by the user.

2. Motivating Use Cases

Some errors which may require a richer experience include:

* User account types which have reduced capabilities (e.g. underage or enterprise accounts)

* Applications which can be disabled or restricted

* Custom authentication strength requirements not covered by an existing spec

* TODO more?

In the common case, errors requiring user attention are likely to appear at authorization time, when the user is present to see and/or address them. However, in circumstances where data used for access control is changed out of band, these errors could result in failures at refresh token exchange or access token usage. In this situation, the error codes defined by the original OAuth 2.0 spec are insufficiently expressive to address these different error scenarios. The error_description parameter also cannot cover errors that:

* Need to display richer detail (e.g. explain to user why account is blocked)

* Contain sensitive data (e.g. account has been flagged as underage)

Because the user is not always be present when the client receives an error from the authorization or resource server, there needs to be some way to preserve error state until the user is present again. Simply returning invalid_grant so that the client reattempts the authorization flow has a few downsides:

* Some clients may not automatically restart authorization and instead render a button to do so. In this case the user has no explanation for why they’re suddenly signed out of the client.

* It may result in unnecessary work for the user if they need to complete some preliminary auth steps in order to reach the error state (e.g. sign in or account selection for authorization servers that support multiple login).

* invalid_grant is semantically the wrong error code for many error conditions.

3. 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.

4. Error Responses

4.1. Token endpoint

When returning an error response from the token endpoint, the authorization server MAY return an additional parameter with any error code.

error_state
: Indicates that the client SHOULD initiate a new authorization grant flow and add this parameter as the error_state parameter there. The client MUST NOT modify or introspect the error_state parameter.

In this case, error_state is a private contract wholly within the authorization server, so its format requires no specification (though see considerations below).

4.2. Bearer authentication scheme

This specification defines the following WWW-Authenticate auth-param value, which may be presented alongside any error code.

error_body
: A boolean indicating that the client SHOULD initiate a new authorization grant flow and add the response body as the error_state parameter there. The client MUST NOT modify or introspect the error response body.

If access tokens are validated by the authorization server, the error response body is a private contract wholly within the authorization server and requires no specification (though see considerations below).

If access tokens are validated directly on the resource server (e.g. using an authorization server public key), then the authorization server will need to understand the error_state generated by the resource server. It is RECOMMENDED to represent error_state as a JWT [RFC7519], and use JSON Web Encryption [RFC7516] to encrypt it.

4.2.1. Resource server error_state JWT

If using a JWT to represent error_state, the standard JWT claims MUST be set as follows:

* typ: "error+jwt"
* iss: Endpoint URL of the resource server endpoint returning the error
* aud: Authorization server issuer as defined by its metadata endpoint [RFC8414]
* sub: Identifier of the user whose access token triggered the error
* iat: Issue time
* nbf: Not before
* exp: Expiration time, but see considerations below

The JWT will contain other custom claims that can be understood by the authorization server to display the error, but due to the diverse nature of potential errors, it’s not possible to enumerate them in a spec. Authorization servers MUST ignore unknown claims.

4.3. error_state considerations

It is RECOMMENDED that error_state be opaque to the client, though clients MUST NOT introspect the parameter regardless. If error_state contains any user data, it MUST be encrypted. It is RECOMMENDED that error_state contains a timestamp and the authorization server defines a reasonable timeframe in which it must be used. Given that there may be some human-introduced delay before error_state is sent to the authorization endpoint and the expected lack of sensitivity of the error_state parameter, it is RECOMMENDED that the expiration timestamp be at least 1 day.

The authorization server and resource server MAY use error_state to represent both recoverable and non-recoverable errors. Examples of recoverable errors could include step-up auth or re-acknowledgment of updated terms of service. Examples of non-recoverable errors could include account or API disables where the authorization server needs to communicate some information to the user.

It is NOT RECOMMENDED to distinguish between recoverable and non-recoverable errors to the client, as it potentially leaks sensitive information to the client. (Signals that should be shared for security and abuse reasons should use the protocols defined in [OpenID SSF].)

4.4. New error codes

This specification additionally defines the access_denied error code for the token endpoint and resource servers, as an error code that may be more appropriate in some cases where error_state is returned.

access_denied
: Indicates that the request could not be completed, despite being well-formed and properly authenticated. This error code MUST be returned as an HTTP 403.

5. Authorization Grant Request

If the client is not running in a context where it can initiate an authorization grant flow (e.g. a workflow running on some cloud VM), it MAY transmit the error_state parameter to the system that can.

Authorization servers that support error_state MUST support at least one of the following to avoid sending long encrypted parameters in the URL:

* HTTP POST requests to the authorization grant flow

* Pushed authorization requests [RFC9126]

The authorization server MUST ignore an expired error_state parameter (though it is entirely possible that the same error will be re-encountered during a fresh authorization grant flow).

6. Security Considerations

The resource server could in theory return an entire authorization URL that the client should follow, but doing so could allow a compromised resource server to direct the user to a malicious authorization server that would phish or steal user credentials. Returning just the error_state parameter prevents this as clients are responsible for targeting the correct authorization endpoint.

TODO Risk of attaching error_state to URL for different user.

TODO Guidance on background-running clients that need to notify the user. Don’t want to train users to click on authz links.

7. Privacy Considerations

Encrypting the error_state parameter allows the authorization and resource servers to embed additional user data into the parameter that should not be visible to clients. For example, unencrypted, a server couldn’t reveal in the error that access was denied because the user is underage. However, encrypted (and salted), there’s no problem plumbing an under_age flag to the authorization server.

8. IANA Considerations

8.1. OAuth Parameters Registration

This specification registers the following OAuth parameter definitions in the IANA OAuth Parameters registry.

8.1.1. Registry Contents

* Name: error_state
  - Parameter Usage Location: authorization request, token response
  - Change Controller: IETF
  - Reference: This document

9. 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>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/rfc/rfc6750>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/rfc/rfc7516>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[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>.

[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>.

[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>.

Acknowledgments

TODO acknowledge.

Author’s Address

Nicholas Watson
Google, LLC
Email: nwatson@google.com