

OAuth 2.0 Proxied Token Introspection

Publication Date: [2025-10-27]
Authors: Dominik František Bučík, Marcus Hardt, Jens Jensen, Ivan Kanakarakis, Christos Kanellopoulos (ed.), Tomasz Kuczyński, Nicolas Liampotis (ed.), Jan Pavlíček, Mischa Sallé
Document Code: AARC-G052
DOI: 10.5281/zenodo.10205863
Community: Architecture Area

Abstract

This specification extends the OAuth 2.0 Token Introspection method defined in RFC7662. It defines a method for an OAuth 2.0 Authorization Server (AS) that receives an introspection request for a token it did not issue, to query a different, trusted AS. This enables the AS to determine the active state of the token and to retrieve associated metadata.



Table of Contents

1. Introduction	3
1.1 Notational Conventions	3
1.2 Terminology	3
2. Proxied Token Introspection Request and Response	5
2.1 Token Validation by Resource Server	5
2.2. Token Validation by AS Performing Proxied Token Introspection	6
2.3. Token Introspection Response by the token-issuing AS	7
2.4. Token Introspection Response by AS performing Proxied Token Introspection	8
3. Implementation Considerations	10
4. Security Considerations	12
5. Privacy Considerations	13
6. References	14

1. Introduction

This specification extends the OAuth 2.0 Token Introspection method defined in [\[RFC7662\]](#). In particular, it enables the introspection endpoint of an Authorization Server (AS) that receives an introspection request for a token, not issued by itself, to initiate a new token introspection request toward a different, trusted AS to determine the validity of the set of metadata related to the token. As described in [\[RFC7662\]](#), the token metadata can include but are not limited to the following:

- whether or not the token is currently active (or if it has expired or otherwise become invalid),
- what access rights the token carries; rights are typically conveyed through OAuth 2.0 scopes or other authorization claims including resource owner memberships in roles and groups that are relevant to the resource being accessed, or through entitlements assigned to the resource owner for the targeted resource that the authorization server knows about (see also [\[RFC9068\]](#)),
- the authorization context in which the token was granted, including information such as the subject of the token and the client that the token was issued to.

The method defined in this specification termed Proxied Token Introspection addresses challenges in environments where there are multiple Authorization Servers (ASes) issuing tokens. In such environments, protected resources may have a trust relationship with a specific AS but may still receive OAuth 2.0 tokens issued by any of the ASes. This specification requires that each AS provide an introspection endpoint per [\[RFC7662\]](#) and can itself act as a client that can proxy token introspection requests and responses to another AS.

The proxied token introspection approach allows for the following:

- Resource servers can verify OAuth 2.0 tokens that were issued by ASes with which the RS is not registered; instead an AS they are registered with can proxy their token introspection requests to the AS that actually issued the token.
- If necessary, the AS performing proxied token introspection can alter, add or remove information in the token - such as authorisation statements - from the domain of the token-issuing AS to its own domain.

1.1 Notational Conventions

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 [\[RFC2119\]](#).

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

1.2 Terminology

This section defines the terminology used by this specification. This section is a normative portion of this specification, imposing requirements upon implementations.

This specification uses the terms "access token", "authorization endpoint", "authorization grant", "authorization server" ("AS"), "client", "client identifier", "protected resource", "refresh token", "resource owner", "resource server" ("RS"), and "token endpoint" defined by OAuth 2.0 [\[RFC6749\]](#), the terms "claim names" and "claim values" defined by JSON Web Token (JWT) [\[RFC7519\]](#), and the terms "token introspection" and "introspection endpoint" defined by OAuth 2.0 Token Introspection [\[RFC7662\]](#).

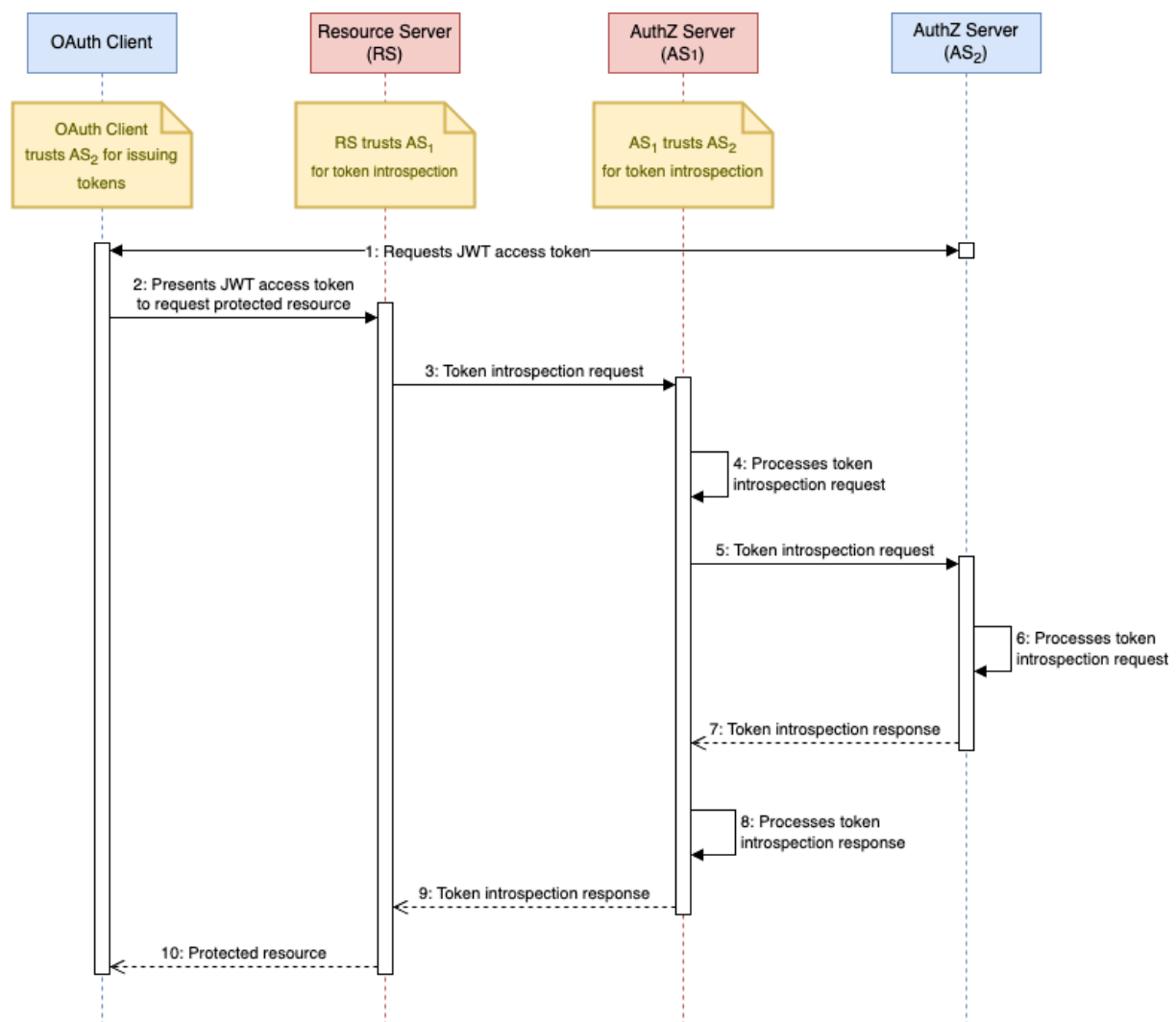
This specification defines the following term:

Proxied Token Introspection

The act performed by an AS when it does not maintain information about an OAuth 2.0 token and acts itself similarly to a resource server towards another trusted AS for introspecting that token using the network protocol defined in this document.

2. Proxied Token Introspection Request and Response

The abstract flow illustrated in Figure 1 describes the interactions among the entities involved in proxied token introspection. In Step 1, the OAuth 2.0 client requests an access token by authenticating with its trusted AS, AS₂, by presenting the authorization grant. AS₂ authenticates the client and validates the authorization grant, and if valid, issues an access token.



2.1 Token Validation by Resource Server

The OAuth 2.0 client uses the access token to request the protected resource from the resource server (RS) (see Step 2 in Fig. 1). The RS treats the token as an opaque token and calls the introspection endpoint of its trusted AS (AS₁ in Fig. 1) using an HTTP request as defined in OAuth 2.0 Token Introspection [\[RFC7662, Section 2.1\]](#) (see Step 3 in Fig. 1).

Access to the introspection endpoint MUST require some form of authorization as described in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.1](#).

The following is a non-normative example request where, for the purpose of authorization, the RS acts as an OAuth 2.0 client and uses a client identifier and client secret to authenticate itself to the introspection endpoint:

```
POST /introspect HTTP/1.1
Host: as1.example.org
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic cnM6Z3FWWFFUZmVMSHRTdnhzTg==

token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2FzMi5leGFtcGxlLm9yZy8iLCJzdWIiOiI1YmE1NTJkNjciLCJleHAiOiJlE3NDYwODMxNjYsImIhdCI6MTc0NjA3OTU2NiwiianRpIjoiaZGJlMzliZjNhM2JhNDIzOGE1MTNmNTFkNmUxNjkyZzQ1LCJjbGllbnRfaWQiOiJzNkJoZlJrcXQzIiwic2NvcGUiOiJvcGVuaWQgcHJvZmZlbnRfaWQiOiJzNkJoZlJrcXQzIiwiaWF0IjoiK9TnkoCRjYbCDAvIDMlR400SPeK8B22Pd2xKAZgxaA
```

The token in the example includes the following claims:

```
{
  "iss": "https://as2.example.org/",
  "sub": "5ba552d67",
  "exp": 1746083166,
  "iat": 1746079566,
  "jti" : "dbe39bf3a3ba4238a513f51d6e1691c4",
  "client_id": "s6BhdRkqt3",
  "scope": "openid profile email entitlements"
}
```

2.2. Token Validation by AS Performing Proxied Token Introspection

In processing the request, AS₁ MUST determine whether the token has been issued by itself (see Step 4 in Fig. 1). This specification requires signed access tokens to be presented in JWT format containing the `iss` claim for identifying the issuer of the token, as defined in JWT [\[RFC7519\]](#). The handling of opaque and/or encrypted access tokens is out of scope of this specification (see also Sections 3 & 5).

If AS₁ determines that the token has been issued by itself, then an introspection response as defined in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.2](#), MUST be returned.

If the token has not been issued by AS₁ itself, it MUST either:

2.4. Token Introspection Response by AS performing Proxied Token Introspection

The AS performing the Proxied Token Introspection (AS₁ in Fig. 1) responds to the resource server with a JSON object [\[RFC7159\]](#) in "application/json" format with the top-level members defined in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.2](#) (see Step 9 in Fig. 1). The response SHOULD include the claim values included in the response from the token-issuing AS (AS₂ in Fig. 1). AS₁ MAY adjust the claims and/or claim values included in the response considering the requirements of the resource server which made the original request (see Step 8 in Fig. 1).

The following is a non-normative example of a response for an active token:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "active": true,
  "iss": "https://as2.example.org/",
  "sub": "5ba552d67",
  "client_id": "s6BhdRkqt3",
  "scope": "openid profile email entitlements",
  "exp": 1746083166,
  "iat": 1746079566,
  "entitlements": [
    "e9e30dba-f08f-4109-8486-d5c6a331660a",
    "entitlement2"
  ]
}
```

The AS performing the Proxied Token Introspection (AS₁ in Fig. 1) MUST return an introspection response with the "active" field set to "false" if the introspection call is properly authorised but any of the following conditions apply:

- this particular token is not valid for use by the RS making the request¹; for instance, if AS₁ determines that the token is not locally issued and is of an OAuth 2.0 token type [\[OAuth-TT\]](#) which cannot be used as an OAuth 2.0 bearer token, such as a refresh token,
- AS₁ can not validate the token through any of the trusted AS (refer to [Section 2.2](#)).

The following is a non-normative example response for a token that has been revoked or is otherwise invalid:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

¹ The details of how the AS makes such a decision are outside the scope of this specification.

```
{  
  "active": false  
}
```

If the introspection call is not authorised then the AS performing the proxied token introspection MUST respond with an HTTP 401 code as described in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.3](#).

3. Implementation Considerations

The AS is an entity that implements Proxied Token Introspection as defined in this specification. The AS may be a system of one or more ASes acting as a frontend for one or more other ASes. Although in the examples of this document we present a topology where the AS trusted by the RS has a direct trust relationship with the AS that has issued the OAuth 2.0 Token, more complex topologies are also possible where between the AS of the RS and the token-issuing AS there is a chain of ASes that implement the Proxied Token Introspection as described in this document.

The means by which the RS chooses which entity to invoke are outside the scope for this specification. Note that this is analogous to the discovery of the token introspection defined in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2](#).

When forming its proxied token introspection response, the AS MUST NOT change claims related to the token itself (i.e. the `iss`, `exp`, `iat`, `nbf`, `token_type`, `client_id`, and `jti` claims value of the response²), but it MAY modify other claims or their values. Furthermore, the AS MAY respond differently to different resource servers making the same request, as described in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.2](#). For instance, the AS implementing this specification MAY limit which scopes (from a given token) are returned to each resource server. In other cases, the AS implementing this specification MAY limit, extend or modify the entitlements [\[RFC9068\]](#) from a given token depending on the protected resource making the request. Additionally, the AS MAY change the string identifier(s) representing the intended audience for the token.

The response to a properly formed and authorised query for an inactive or otherwise invalid token is not considered an error response by this specification. Instead, the AS MUST respond with an introspection response with the "active" field set to "false". That response SHOULD NOT include any additional information about the inactive token, including why the token is inactive, as described in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.2](#).

As described in OAuth 2.0 Token Introspection [\[RFC7662\], Section 4](#), the response MAY be cached by the resource server to improve performance, based on its own policy and risk profile. In the context of this specification, the AS performing Proxied Token Introspection (AS₁ in Fig. 1) MAY also cache responses from the trusted AS (AS₂ in Fig. 1) to reduce load on each of the ASes that are in the path between the AS trusted by the RS and the token-issuing AS. Such caching introduces a tradeoff between performance and the freshness of token validation information, as AS₁ may not perform real-time validation for each request, which contrasts with the expectations of [\[RFC7662\]](#) for up-to-date checks. To mitigate risks associated with cached responses, implementers SHOULD consider countermeasures, such as mechanisms to ensure transparency and freshness of cached data, though such measures are outside the scope of this specification and may be defined in specific profile extensions to AARC-G052. Implementers should evaluate whether caching is suitable for their deployment, considering the RS's reliance on fresh token state information.

² For use cases where the `iss` claim needs to be modified, a method based on Token Exchange [\[RFC8693\]](#) would be more appropriate. This is out of the scope of this specification.

The token-issuing AS MAY respond differently depending on the identity of the entity performing the token introspection request, as described in OAuth 2.0 Token Introspection [\[RFC7662\]](#), [Section 2.2](#). In addition, some AS implementations enforce audience restrictions by requiring that the requestor is included in the audience for the token. While such checks are important for securing token use, the specification of how to signal or validate audiences is out of scope of both [\[RFC7662\]](#) and this specification. Profiles building on this specification SHOULD define mechanisms for audience signalling and validation (e.g. using audience claims or resource indicators [\[RFC8707\]](#)) to ensure that tokens are used only by their intended recipients.

4. Security Considerations

The security considerations discussed in OAuth 2.0 Token Introspection [\[RFC7662\], Section 4](#) also apply to this specification. In addition to these considerations, this specification requires mechanisms (such as [\[RFC7515\]](#)) to ensure that access tokens presented in JWT format get delivered without having been tampered with. Although the token-issuing AS can use any algorithm defined in [\[RFC7518, RFC9864\]](#) for signing JWT access tokens, use of asymmetric keys is RECOMMENDED as it simplifies the process of acquiring validation information. As per [\[RFC9068\], Section 2.1](#), signed JWT access tokens MUST NOT use "none" as the signing algorithm. Furthermore, authorization servers conforming to this specification MUST include RS256 (as defined in [\[RFC7518\], Section 3.1](#)) among their supported signature algorithms. Entities performing proxied token introspection are not required to validate the signature of the token.

As per [RFC7662, Section 4](#), if the token can be used only at certain resource servers, the AS MUST determine whether or not the token can be used at the entity making the introspection call. How the client or the AS issuing the token is able to specify the correct audience(s) is out of scope of this specification. As described in [Section 3](#), the AS MAY adjust the introspection response contents based on the resource server making the request.

For cross-infrastructure use of access tokens, the protected resource treats the access token as opaque and hence there is no risk of Cross-JWT Confusion ([\[RFC8725\], Section 2.8](#)).

The following mechanisms can prevent token scanning attacks and overloading of the token introspection endpoint:

- the AS implementing this specification MUST require some form of authorisation to the introspection endpoint, as described in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.1](#),
- the AS implementing this specification SHOULD employ rate-limiting mechanisms ([\[RFC7662\], Section 4](#)), based on the identity of the RS performing the token introspection request³
- the RS MAY cache responses, as described in OAuth 2.0 Token Introspection [\[RFC7662\], Section 2.2](#).
- the AS implementing this specification MAY cache responses, as described in [Section 3](#).

³ Otherwise the AS that performs proxied token introspection is at risk of becoming overloaded due to a high volume of introspection requests from resource servers or being rate-limited by the token-issuing AS.

5. Privacy Considerations

The use of Proxied Token Introspection introduces constraints on the use of encrypted access tokens, such as JSON Web Encryption (JWE). Since ASes cannot inspect the contents of encrypted tokens (e.g. the `iss` claim) without access to decryption keys, they cannot reliably make routing or delegation decisions. Consequently, access tokens intended for use across trust boundaries—where resource servers rely on an AS different from the token-issuing AS—must remain inspectable by the AS performing Proxied Token Introspection. This requirement implies that full encryption of token contents is not feasible in such scenarios. This constraint represents a privacy tradeoff, as access tokens must remain unencrypted to support interoperability in federated environments.

The content of the JWT access token will be accessible to the resource server and eventually to the AS that performs proxied token introspection. Therefore, measures **MUST** be taken to prevent disclosure of personal data in the JWT claims to unintended parties. For instance, all sensitive data could be left out of the token and be released only in an introspection response. In general, measures **MUST** be taken to prevent disclosure of this information to unintended parties., as discussed in OAuth 2.0 Token Introspection [\[RFC7662\], Section 5](#).

6. References

- [AARC-G045] AARC Blueprint Architecture (AARC-G045);
<<https://aarc-community.org/guidelines/aarc-g045>>
- [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/info/rfc2119>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012,
<<http://www.rfc-editor.org/info/rfc6749>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015,
<<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC7518, May 2015,
<<https://www.rfc-editor.org/info/rfc7518>>.
- [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>>.
- [RFC7662] Richer, J., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015,
<<https://www.rfc-editor.org/rfc/rfc7662>>.
- [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/info/rfc8414>>.
- [RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020,
<<https://www.rfc-editor.org/info/rfc8693>>.
- [RFC8707] Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707, DOI 10.17487/RFC8707, February 2020,
<<https://www.rfc-editor.org/info/rfc8707>>.
- [RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <<https://www.rfc-editor.org/info/rfc8725>>.
- [RFC9068] Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", RFC 9068, DOI 10.17487/RFC9068, October 2021,
<<https://www.rfc-editor.org/info/rfc9068>>.
- [RFC9864] Jones, M., "Fully-Specified Algorithms for JSON Object Signing and Encryption (JOSE) and CBOR Object Signing and Encryption (COSE)",

RFC 9864, October 2025

<<https://www.rfc-editor.org/info/rfc9864>>.