OAuth 2.0 Proxied Token Introspection

Publication Date [2023-11-13]

Authors: Dominik František Bučík, Marcus Hardt, Jens Jensen, Ivan Kanakarakis, Christos Kanellopoulos, Nicolas Liampotis (ed.), Mischa Sallé

Document Code: AARC-G052
DOI: *10.5281/zenodo.10205863*

Community: Architecture Area

Abstract

This specification extends the OAuth 2.0 Token Introspection (RFC7662) method to allow conveying meta-information about a token from an Authorization Server (AS) to the protected resource even when there is no direct trust relationship between the protected resource and the token issuer. The method defined in this specification, termed "proxied" token introspection, requires access tokens to be presented in JWT format containing the `iss` claim for identifying the issuer of the token. Proxied token introspection assumes that the AS which is trusted by the protected resource has established a trust relationship with the AS which has issued the token that needs to be validated.

Table of Contents

1. Introduction

This specification builds on the OAuth 2.0[1] Token Introspection [RFC7662] method to allow a resource server to query an authorization server (AS) it trusts for determining the set of metadata for a token regardless of whether or not that token has been issued by the given authorization server. 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 does not require protected resources in one organisation to trust the authorization servers of every other organisation eligible for issuing tokens. However, the specification requires the presence of an AS Proxy, providing an introspection endpoint for the protected resource. The AS Proxy is trusted by the protected resource and has established a trust relationship with the AS which issued the token that needs to be validated. Similar to [RFC7662], proxied token introspection allows a protected resource to query the token metadata regardless of whether or not this information is carried in the token itself.

It should be emphasised that this specification does not preclude offline validation (e.g. by the RS or the AS proxy) of access tokens.

The remote validation approach allows for the following:

- resource servers are not required to establish direct trust relationships with all the AS that issue tokens,
- tokens are not required to include all the metadata within the token itself (which in some cases may include sensitive information such as the resource owner memberships in groups),
- support for tokens with minimal JWT structure; the only requirement is to include information for determining the token issuer,
- support for revocation of access tokens (without token introspection, there is no way to invalidate a token until it expires),
- low complexity for protected resources since the token validation process is delegated to their AS.

[1] The upcoming OAuth 2.1 specification includes a reference to Token Introspection (RFC7662) so this specification should be compatible with OAuth 2.1.

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 terms:

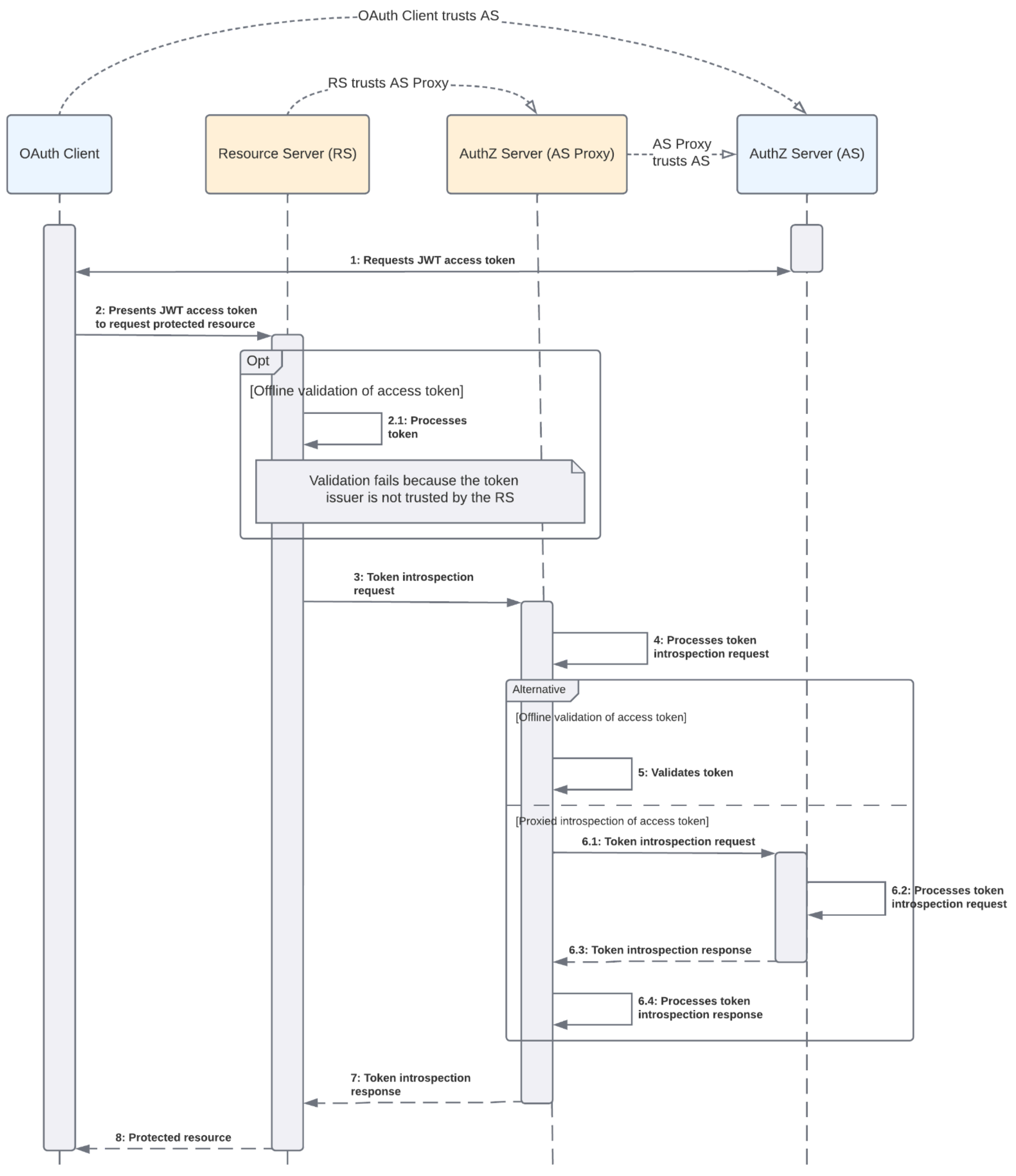
AS Proxy
: The entity implementing Proxied Token Introspection; it is trusted by the resource server inquiring about the current state of an OAuth 2.0 token. The AS Proxy may be the same entity as the Authorization Server or a separate entity.

Offline validation of JWT access tokens
: The act of validating access tokens by validating the signature of the token as defined in [RFC7519], Section 7.2 and parsing the claims within the structured token itself.

Proxied Token Introspection
: The act of inquiring about the current state of an OAuth 2.0 token through use of 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 client requests an access token by authenticating with the AS and presenting the authorization grant. The AS authenticates the client and validates the authorization grant, and if valid, issues an access token.


OAuth Client trusts AS
RS trusts AS Proxy
OAuth Client
Resource Server (RS)
AuthZ Server (AS Proxy)
AS Proxy trusts AS
AuthZ Server (AS)
1: Requests JWT access token
2: Presents JWT access token to request protected resource
Opt
[Offline validation of access token]
2.1: Processes token
Validation fails because the token issuer is not trusted by the RS
3: Token introspection request
4: Processes token introspection request
Alternative
[Offline validation of access token]
5: Validates token
[Proxied introspection of access token]
6.1: Token introspection request
6.2: Processes token introspection request
6.3: Token introspection response
6.4: Processes token introspection response
7: Token introspection response
8: Protected resource

2.1 Token Validation by Resource Server

The OAuth client uses the access token to request the protected resource from the resource server (RS) (see Step 2 in Fig. 1). The RS MAY attempt to perform offline token validation (see optional Step 2.1 in Fig. 1). If the RS cannot determine[2] the issuer of the token or determines that the token has not been issued by a trusted issuer (which is the case illustrated in Fig 1) then the RS MUST treat the token as an opaque token and call the introspection endpoint of the AS Proxy. In Fig 1 we assume that the RS trusts a single AS. Refer to Section 3 for other deployment scenarios

The RS calls the introspection endpoint of the AS Proxy 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 resource server uses a client identifier and client secret to authenticate itself to the introspection endpoint:

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

      token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwc
      zovL2FzLmV4YW1wbGUub3JnLyIsInN1YiI6IjViYTU1MmQ2NyIsImF1ZCI6W
      yJodHRwczovL3JzLmV4YW1wbGUub3JnLyIsImh0dHBzOi8vYXMtcHJveHkub
      3JnLyJdLCJleHAiOjE2Mzk1Mjg5MTIsImlhdCI6MTYxODM1NDA5MCwianRpI
      joiZGJlMzliZjNhM2JhNDIzOGE1MTNmNTFkNmUxNjkxYzQiLCJjbGllbnRfa
      WQiOiJzNkJoZFJrcXQzIiwic2NvcGUiOiJvcGVuaWQgcHJvZmlsZSBlbWFpb
      CBlbnRpdGxlbWVudHMifQ.qbBOqzZQlCVzJvaF-j1Gr9oGVl_VgOrSbm0Kmx
      P59ew
```

The token in the example includes the following claims:

```
{
    "iss": "https://as.example.org/",
    "sub": "5ba552d67",
    "aud": ["https://rs.example.org/", "https://as-proxy.org/"],
    "exp": 1639528912,
    "iat": 1618354090,
    "jti" : "dbe39bf3a3ba4238a513f51d6e1691c4",
```

[2] For example, when the access tokens are encrypted and can only be decrypted by the AS Proxy.

```
    "client_id": "s6BhdRkqt3",
    "scope": "openid profile email entitlements"
}
```

2.2 Token Validation by AS Proxy

In processing the request (see Step 4 in Fig. 1), the AS Proxy MUST determine whether the token has been locally issued. This specification requires access tokens to be presented in JWT format containing the `iss` claim for identifying the issuer of the token, as defined in JWT [RFC7519]. Signing mechanisms (such as [RFC7515]) are required to ensure that access tokens get delivered without having been tampered with, as described in Section 4. If access tokens are encrypted the AS Proxy MUST be able to decrypt them.

If the AS Proxy determines that the token has been locally issued 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 locally issued then the AS Proxy can fulfil the token introspection request using various methods. These methods may include calling the introspection endpoint of the trusted issuer (provided the trusted issuer supports token introspection - see Step 6.1 in Fig. 1) or using other means (e.g. offline validation of access token - see Step 5 in Fig. 1). The implementer has the discretion to select and use one or more of these methods, and the specific order in which they are invoked by the AS Proxy to fulfil the token introspection request is an implementation decision.

The means by which the AS Proxy discovers the location of the introspection endpoint are outside the scope of this specification; for instance, the AS Proxy may use the OAuth 2.0 Authorization Server Metadata [RFC8414], Section 2 to discover the location of the introspection endpoint. The AS Proxy calls the introspection endpoint of the trusted issuer using an HTTP request as defined in OAuth 2.0 Token Introspection [RFC7662], Section 2.1. The introspection endpoint of the trusted issuer MUST support client authentication as described in OAuth 2.0 [RFC6749] to authorise introspection requests from AS Proxy entities. The means by which the AS Proxy discovers the client authentication methods supported by the introspection endpoint of the trusted issuer (AS) are outside the scope of this specification; for instance, the AS Proxy may use the OAuth 2.0 Authorization Server Metadata [RFC8414], Section 2 to discover the supported client authentication methods.

The following is a non-normative example request where the AS Proxy uses a client identifier and client secret to authenticate itself to the introspection endpoint of the token-issuing AS:

```
POST /introspect HTTP/1.1
     Host: as.example.org
     Accept: application/json
     Content-Type: application/x-www-form-urlencoded
     Authorization: Basic YXMtcHJveHk6QWRaaGVLcTQ3S20ycmJaTA==
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwc
zovL2FzLmV4YW1wbGUub3JnLyIsInN1YiI6IjViYTU1MmQ2NyIsImF1ZCI6W
yJodHRwczovL3JzLmV4YW1wbGUub3JnLyIsImh0dHBzOi8vYXMtcHJveHkub
3JnLyJdLCJleHAiOjE2Mzk1Mjg5MTIsImlhdCI6MTYxODM1NDA5MCwianRpI
joiZGJlMzliZjNhM2JhNDIzOGE1MTNmNTFkNmUxNjkxYzQiLCJjbGllbnRfa
WQiOiJzNkJoZFJrcXQzIiwic2NvcGUiOiJvcGVuaWQgcHJvZmlsZSBlbWFpb
CBlbnRpdGxlbWVudHMifQ.qbBOqzZQlCVzJvaF-j1Gr9oGVl_VgOrSbm0Kmx
P59ew
```

2.3 Token Introspection Response by AS

The token-issuing AS responds to the AS Proxy (see Step 6.3 in Fig. 1) 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.

2.4 Proxied Token Introspection Response by AS Proxy

The AS Proxy responds to the resource server (Step 7 in Fig. 1) 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. The response SHOULD include the claim values included in the response from the token-issuing AS (or parsed during the offline validation of the token). The AS Proxy 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 6.4 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://as.example.org/",
  "sub": "5ba552d67",
  "client_id": "s6BhdRkqt3",
  "scope": "openid profile email entitlements",
  "aud": "https://rs.example.org/",
  "exp": 1639528912,
  "iat": 1618354090,
  "entitlements": [
    "e9e30dba-f08f-4109-8486-d5c6a331660a",
    "entitlement2"
  ]
}
```

The AS Proxy 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 request[3]; for instance if the AS Proxy 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
- the AS Proxy can not validate the token through any of the supported methods (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

{
  "active": false
}
```

[3] The details of how the AS Proxy makes such a decision are outside the scope of this specification.

3. Implementation Considerations

The “AS proxy” is an entity that implements proxied token introspection and is trusted by the RS. It may not be an AS, it may be a system acting as a frontend for one or more ASs, or it may be one of the set of AS(s) trusted by the RS. 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 Proxy MUST NOT change the `iss` claim value of the response[4], but it MAY modify other claims or their values. Furthermore, the AS Proxy 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 Proxy MAY limit which scopes (from a given token) are returned to each resource server. In other cases, the AS Proxy MAY limit, extend or modify the entitlements [RFC9068] from a given token depending on the protected resource making the request. The AS Proxy MAY change the string identifier(s) representing the intended audience for the token (see example response in Section 2.4). The AS Proxy SHOULD NOT expect to identify itself as the intended audience for the token.

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 Proxy MUST respond with an introspection response with the `"active"` field set to `"false"`. That response SHOULD NOT include any additional information about an inactive token, including why the token is inactive, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.2.

If the introspection call is not authorised then the AS Proxy MUST respond with an HTTP 401 code as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.3.

As described in OAuth 2.0 Token Introspection [RFC7662], Section 4, the response MAY be cached by the resource server and/or the AS Proxy to improve performance and reduce load on the introspection endpoint of the AS Proxy and/or the token-issuing AS, but at the cost of freshness of the information used by the protected resource to make authorization decisions.

When the token-issuing AS is informed[5] of the resource server that will process the access token, it MAY also include the AS Proxy in the intended audience values for that token (in addition to the resource server).

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 implementations enforce audience restrictions on

[4] 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.
[5] The token-issuing AS may be informed of the resource service that will process the access token based on the `request` parameter defined in Resource Indicators for OAuth 2.0 [RFC8707] or the `audience` parameter defined in OAuth 2.0 Token Exchange [RFC8693].

the requestor by requiring that the requestor is included in the audience for the token. However, such mechanisms are out of scope of both RFC7662 and this specification.

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 any algorithm can be used for signing JWT access tokens in the case of [RFC7515], use of asymmetric cryptography 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.

As per RFC7519, each principal intended to process the JWT MUST identify itself with a value in the audience claim. 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 Proxy 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 Proxy MUST require some form of authorization to the introspection endpoint, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.1,
- the AS Proxy SHOULD employ rate-limiting mechanisms, based on the identity of the RS performing the token introspection request[6]
- the RS and/or the AS Proxy MAY cache responses, as described in OAuth 2.0 Token Introspection [RFC7662], Section 2.2.

[6] Otherwise the AS Proxy is at risk of being rate-limited by the token issuing AS.

5. Privacy Considerations

The content of the JWT access token will be accessible to the resource server and eventually to the AS Proxy. Therefore, measures MUST be taken to prevent disclosure of personal data in the JWT claims to unintended parties. Similar considerations apply to the claims included in the token introspection response, 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>.
[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>.

Annex A

This section focuses on achieving interoperability between two domains that use different methods for token validation and introspection:

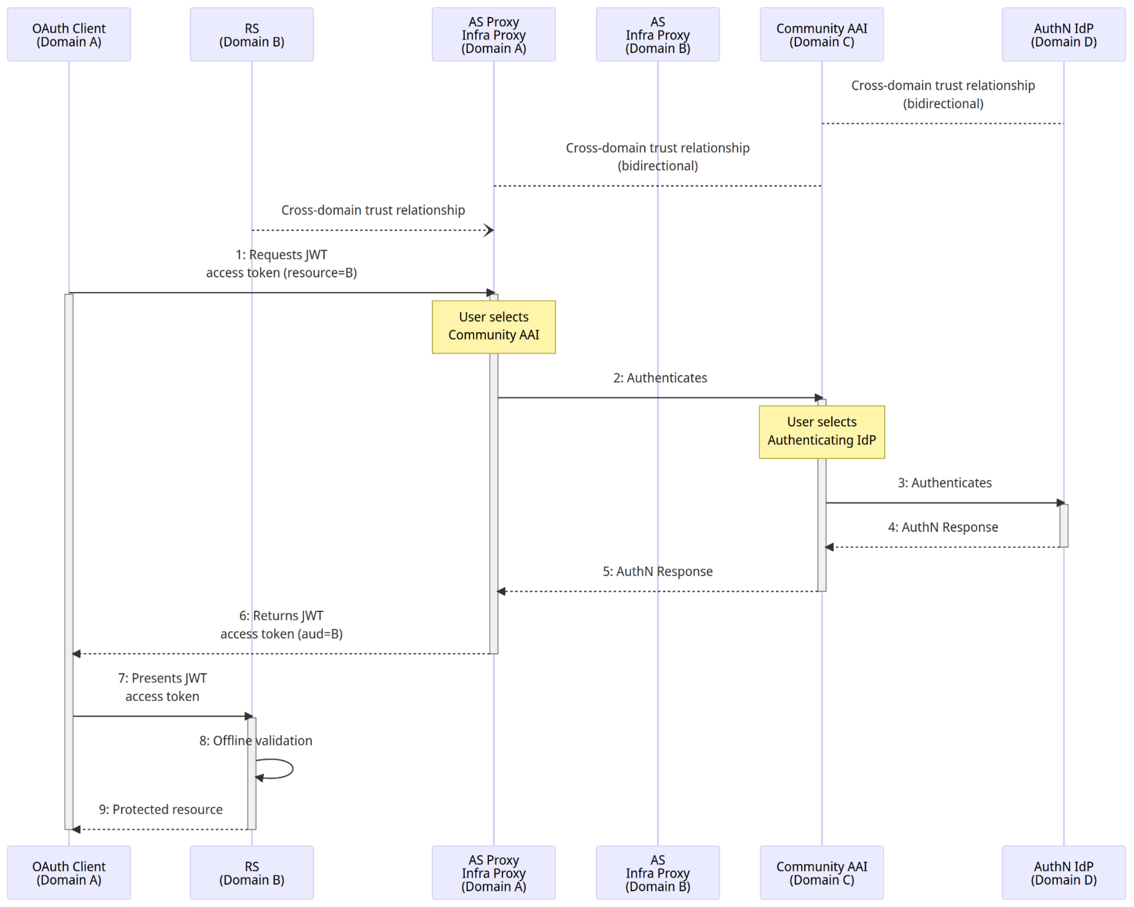
- Domain A relies on proxied token introspection as defined in AARC-G052
- Domain B uses offline token validation.

Additionally, there is a third domain, namely Domain C, which plays a crucial role in the interoperability flows. Domain C represents the Community AAI (AARC-G045) that enables the user to obtain a token based on their community-managed information. This information encompasses information such as groups and roles, which are commonly employed for authorisation purposes. Lastly, Domain D includes the user's Authenticating Identity Provider, enabling user authentication through their Community AAI.

A.1 Token Originating from Domain A

A.1.1 Offline token validation performed by RS

An OAuth client in Domain A requests a JWT access token for a resource in Domain B from its Authentication Server (AS), which is also in Domain A. The user selects their Community AAI in Domain C, which further authenticates the user through their Authentication Identity Provider (AuthN IdP) in Domain D. After successful authentication, a JWT access token is issued and returned to the client. The client then presents this Domain A token to a Resource Server (RS) in Domain B, which performs offline validation and grants access to the protected resource. The diagram illustrates the message flow and trust relationships among the participants in this cross-domain token validation scenario.



OAuth Client (Domain A)
RS (Domain B)
AS Proxy Infra Proxy (Domain A)
AS Infra Proxy (Domain B)
Community AAI (Domain C)
AuthN IdP (Domain D)
Cross-domain trust relationship (bidirectional)
Cross-domain trust relationship (bidirectional)
Cross-domain trust relationship
1: Requests JWT access token (resource=B)
User selects Community AAI
2: Authenticates
User selects Authenticating IdP
3: Authenticates
4: AuthN Response
5: AuthN Response
6: Returns JWT access token (aud=B)
7: Presents JWT access token
8: Offline validation
9: Protected resource

Workflow A.1.1: Offline token validation performed by RS (Domain B). Infra Proxy in Domain A can act as an AS Proxy, however this capability is not relevant in this flow.

Prerequisites:

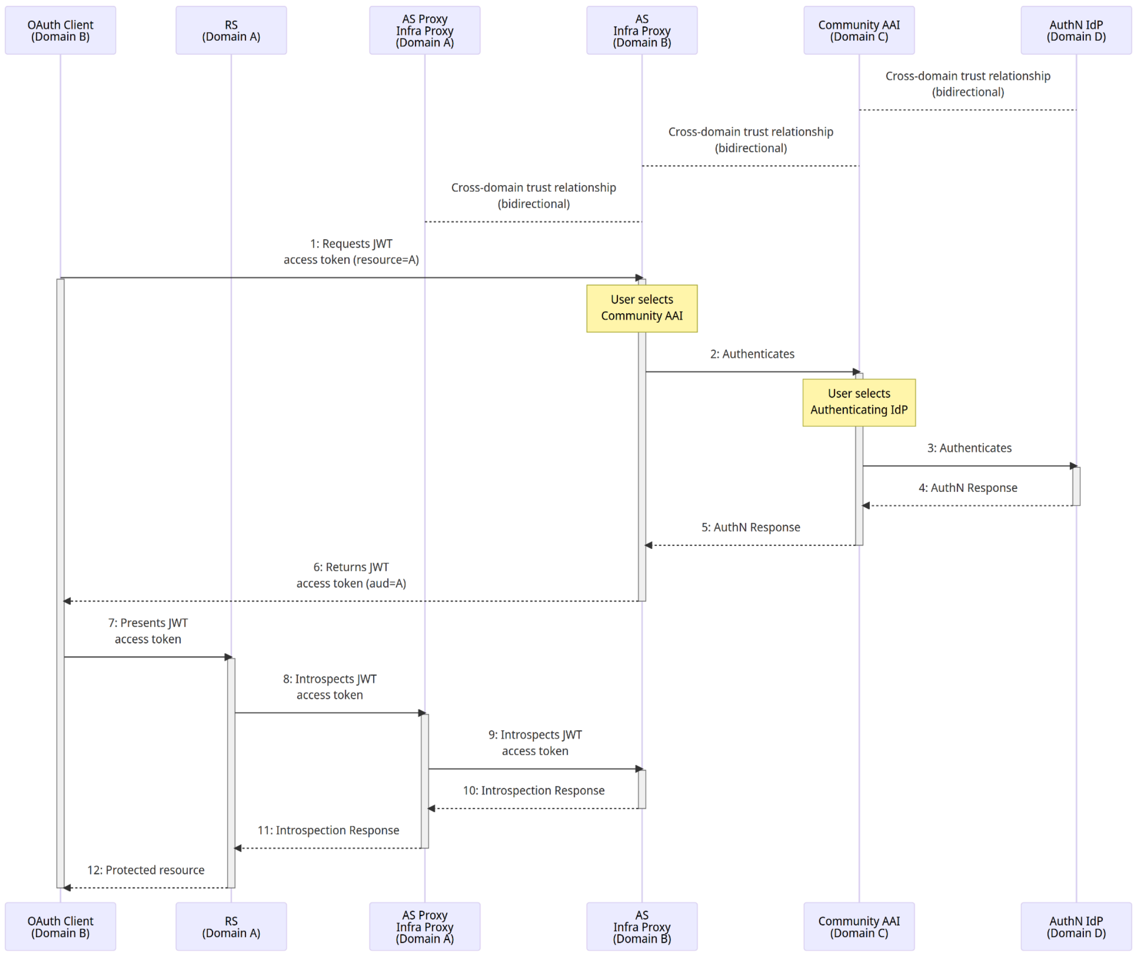
1. Community AAI in Domain C trusts the user's Authenticating Identity Provider (AuthN IdP) (bidirectional trust relationship)
2. AS in Domain A trusts the Community AAI in Domain C (bidirectional trust relationship)
3. RS in Domain B trusts the AS in Domain A
4. RS in Domain B can interpret the contents of tokens originating from Domain A
5. Tokens originating from Domain A contain the authorisation claims

A.2 Token Originating from Domain B

A.2.1 Token introspection invoked by RS, with proxied token introspection performed by AS

An OAuth client in Domain B requests a JWT access token for a resource in Domain A from its Authorization Server (AS), which is also in Domain B. The user selects their Community AAI in Domain C, which subsequently authenticates the user through their Authenticating Identity Provider (AuthN IdP) in Domain D. After successful authentication, a JWT access token is issued and returned to the client. The client then presents this Domain B token to a Resource Server (RS) in Domain A. Before granting access to the protected resource, the

RS introspects the token through its AS, which is also in Domain A. The AS in Domain A identifies the AS in Domain B as the token issuer and is then able to proxy the introspection request as per AARC-G052. The diagram illustrates the message flow and trust relationships among these participants in this cross-domain token validation scenario.


OAuth Client (Domain B)
RS (Domain A)
AS Proxy Infra Proxy (Domain A)
AS Infra Proxy (Domain B)
Community AAI (Domain C)
AuthN IdP (Domain D)
Cross-domain trust relationship (bidirectional)
Cross-domain trust relationship (bidirectional)
Cross-domain trust relationship (bidirectional)
1: Requests JWT access token (resource=A)
User selects Community AAI
2: Authenticates
User selects Authenticating IdP
3: Authenticates
4: AuthN Response
5: AuthN Response
6: Returns JWT access token (aud=A)
7: Presents JWT access token
8: Introspects JWT access token
9: Introspects JWT access token
10: Introspection Response
11: Introspection Response
12: Protected resource

Workflow A.2.1: Token introspection (RFC7662) invoked by RS (Domain A), with proxied token introspection (AARC-G052) performed by AS (Domain A)

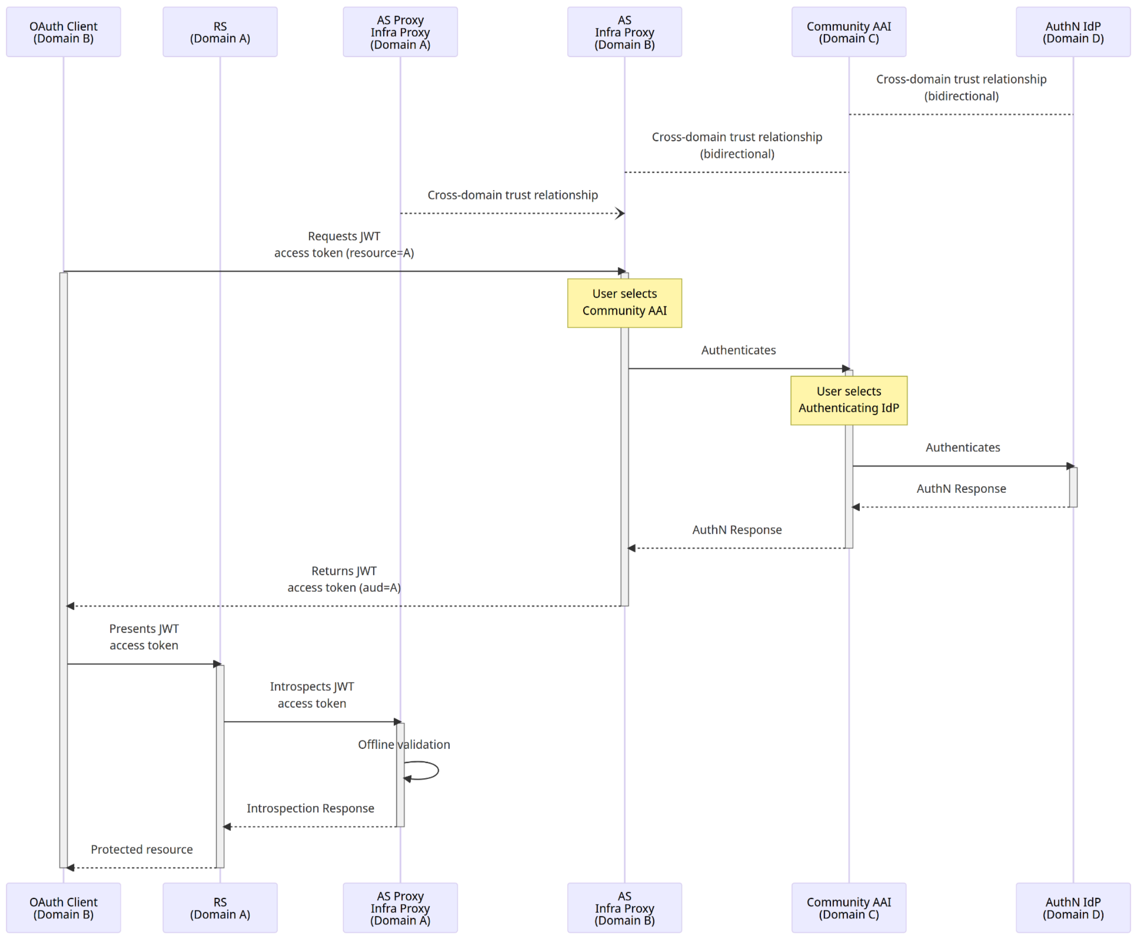
Prerequisites:

1. Community AAI in Domain C trusts the user’s Authenticating Identity Provider (AuthN IdP) (bidirectional trust relationship)
2. AS in Domain B trusts the Community AAI in Domain C (bidirectional trust relationship)
3. AS in Domain A trusts the AS in Domain B (bidirectional trust relationship)
4. AS in Domain A can interpret the contents of the token introspection response from Domain B

A.2.2 Token introspection (RFC7662) invoked by RS, with offline token validation performed by AS

An OAuth client in Domain B requests a JWT access token for a resource in Domain A from its Authorization Server (AS) in Domain B. The user selects their Community AAI in Domain C, which subsequently authenticates the user through their Authenticating Identity Provider

(AuthN IdP) in Domain D. After successful authentication, a JWT access token is issued and returned to the client. The client then presents this Domain B token to the Resource Server (RS) in Domain A. The RS submits an introspection request to its AS in Domain A, which in turn uses offline validation to serve the request. The diagram illustrates the message flow and trust relationships among the participants in this cross-domain token validation scenario.


OAuth Client (Domain B)
RS (Domain A)
AS Proxy Infra Proxy (Domain A)
AS Infra Proxy (Domain B)
Community AAI (Domain C)
AuthN IdP (Domain D)
Cross-domain trust relationship (bidirectional)
Cross-domain trust relationship (bidirectional)
Cross-domain trust relationship
Requests JWT access token (resource=A)
User selects Community AAI
Authenticates
User selects Authenticating IdP
Authenticates
AuthN Response
AuthN Response
Returns JWT access token (aud=A)
Presents JWT access token
Introspects JWT access token
Offline validation
Introspection Response
Protected resource

Workflow A2.2: Token introspection (RFC7662) invoked by RS (Domain A), with offline token validation performed by AS (Domain A)

Prerequisites:

1. Community AAI in Domain C trusts the user's Authenticating Identity Provider (AuthN IdP) (bidirectional trust relationship)
2. AS in Domain B trusts the Community AAI in Domain C (bidirectional trust relationship)
3. AS Proxy in Domain A trusts the AS in Domain B
4. AS Proxy in Domain A can interpret the contents of tokens originating from Domain B
5. Tokens originating from Domain B contain the authorisation claims

A.3 Summary

The table below provides an overview of the different token validation approaches: offline token validation, token introspection with proxied introspection, and token introspection with offline validation. These approaches differ in terms of trust scalability, support for token revocation, callout requirements, and necessary modifications to OAuth client and AS libraries.

Approach	**Advantages**	**Disadvantages**
Offline token validation performed by RS (see A.1.1)	● Does not require callout to token issuer ● Works with standard client and AS libraries	● Trust scalability: Each RS needs to trust all token issuers ● Tokens MUST contain the authorisation claims ● Does not support token revocation
Token introspection (RFC7662) invoked by RS, with proxied token introspection performed by AS (see A.2.1)	● Trust scalability: Only the AS Proxy needs to trust tokens issuers ● Supports token revocation	● Requires callout from RS to AS Proxy and from AS Proxy to token issuer ● Requires modifications to AS libraries
Token introspection (RFC7662) invoked by RS, with offline token validation performed by AS (see A2.2)	● Trust scalability: Only the AS Proxy needs to trust tokens issuers	● Requires callout from RS to AS Proxy ● Tokens MUST contain the authorisation claims ● Does not support token revocation ● Requires modifications to AS libraries