Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 18 August 2026

K. McGuinness
Independent
A. Parecki
Okta

14 February 2026

OAuth 2.0 Token Exchange Target Service Discovery
draft-mcguinness-token-xchg-target-svc-disco-01

Abstract

This specification defines a method for OAuth 2.0 clients to discover the set of available target services (audiences, resources, and scopes) for a given subject token when performing OAuth 2.0 Token Exchange. The discovery endpoint accepts any subject token type registered in the OAuth Token Type Registry and returns values that are valid inputs to subsequent Token Exchange requests, supporting advanced use cases such as identity chaining and cross-domain delegation.

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://mcguinness.github.io/draft-mcguinness-token-xchg-target-svc-disco/draft-mcguinness-token-xchg-target-svc-disco.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-mcguinness-token-xchg-target-svc-disco/.

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/mcguinness/draft-mcguinness-token-xchg-target-svc-disco.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 18 August 2026.

Copyright Notice

Table of Contents

1. Introduction

OAuth 2.0 Token Exchange [RFC8693] enables a client to present a token issued in one security domain to an Authorization Server and securely trade it for a new token issued for another domain. The exchanged token is minted with the target server's token requirements, including its own audience, resource indicators, and scopes, so it becomes valid and enforceable for the desired downstream service. This enables controlled cross-domain access, removes the confused-deputy problem by ensuring tokens are explicitly targeted to the correct service, and avoids requiring direct trust between the original token issuer and the target service.

Authorization Servers may be capable of issuing tokens to multiple services for a given subject token and client, but the client must already know which values it may request. Today, this knowledge is typically provided through static configuration, proprietary APIs, or informal documentation, leading to brittle integrations and unnecessary Token Exchange failures, particularly when subjects are authorized to access only a subset of available services.

This specification defines the OAuth 2.0 Token Exchange Target Service Discovery Endpoint, a standardized mechanism that enables clients to dynamically discover the set of available Token Exchange targets (audiences, resources, and scopes) for a given subject token. The authorization server evaluates both the subject token and the client's permissions and returns only the values the client is authorized to request.

This extension is especially valuable in scenarios requiring identity chaining and cross-domain authorization:

* Progressive token exchange across service boundaries, such as multi-tier microservices architectures and delegated flows where tokens must be exchanged with narrower or transformed permissions at each hop. In these scenarios, discovery is critical because the set of available downstream services and their required permissions vary based on the subject token's context and the client's authorization. The permission narrowing at each service hop means that static configuration cannot accommodate these per-subject and per-client variations, leading to integration failures when clients attempt token exchanges for services the subject is not authorized to access.

* Cross-boundary deployments where downstream services exist in separate administrative, organizational, or cloud domains requiring strict control over token issuance and acceptance. Unlike progressive token exchange which focuses on permission transformation within a single domain, this scenario addresses the challenge of discovering available target services across distinct trust boundaries where authorization policies are independently managed. Discovery is essential here because authorization policies and available target services change dynamically across these boundaries, and static configuration cannot reflect real-time policy decisions or account for varying permissions across different administrative domains.

* Single Sign-On (SSO) to API flows, such as those enabled by the Identity Assertion Authorization Grant (ID-JAG) [I-D.oauth-identity-assertion-authz-grant], where a client needs to seamlessly connect to cross-domain resources and act on behalf of the user to access APIs

This specification provides the following benefits:

* *Dynamic Discovery*: Eliminates static configuration requirements by enabling clients to discover available target services at runtime, reducing integration failures and improving developer experience.

* *Standardization*: Provides a standardized discovery mechanism, replacing proprietary APIs and improving interoperability across OAuth 2.0 implementations.

* *Real-Time Authorization*: Returns target services based on real-time policy evaluation, client permissions, and subject token context, ensuring accurate and up-to-date authorization

information. This enables per-subject and per-client results, which is essential when the set of available target services varies by user or client. Static configurations cannot accommodate these per-subject or per-client variations.

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. Token Exchange Target Service Discovery Endpoint

This specification defines a new endpoint for OAuth 2.0 Authorization Servers that enables clients to discover the set of available target services (audiences, resources, and scopes) for a given subject token when performing OAuth 2.0 Token Exchange [RFC8693].

The endpoint is identified by the Authorization Server metadata parameter token_exchange_target_service_discovery_endpoint, as defined in Section 5.

3.1. Endpoint Request

The client makes a request to the token exchange target service discovery endpoint by sending an HTTP POST request to the endpoint URL. The client MUST use TLS as specified in Section 1.6 of [RFC6749].

The endpoint URL MUST be obtained from the Authorization Server’s metadata document [RFC8414] using the token_exchange_target_service_discovery_endpoint parameter. The endpoint URL is an absolute URL.

The client sends the parameters using the application/x-www-form-urlencoded format per Appendix B of [RFC6749]. Character encoding MUST be UTF-8 as specified in Appendix B of [RFC6749]. If a parameter is included more than once in the request, the authorization server MUST return an error response with the error code invalid_request as described in Section 3.3.

3.1.1. Request Parameters

The following parameters are used in the request:

subject_token REQUIRED. The subject token used as input to

discovery. The token type is indicated by the subject_token_type parameter.

subject_token_type REQUIRED. A string value containing a URI, as described in Section 3 of [RFC8693], that indicates the type of the subject_token parameter. This identifier MUST be a valid URI, and SHOULD be registered in the "OAuth URI Registry" as established by [RFC6755].

The client MAY include additional parameters as defined by extensions and the authorization server MUST ignore unknown parameters.

Client authentication MAY be required by the authorization server. The means of client authentication are defined by the authorization server and MAY include any method supported by the authorization server, including those defined in Section 2.3 of [RFC6749] and extensions. If client authentication is required by the authorization server but not provided in the request, the authorization server MUST return an error response with the error code invalid_client as described in Section 3.3.

3.1.2. Subject Token Processing

The authorization server MUST process the subject_token parameter according to the following rules:

1. The authorization server MUST validate that the subject_token and subject_token_type parameters are not empty strings. If either parameter is an empty string, the authorization server MUST return an error response with the error code invalid_request as described in Section 3.3.

2. The authorization server MUST validate that the subject_token_type parameter is a valid URI. If the format is invalid (e.g., not a valid absolute URI or URN), the authorization server MUST return an error response with the error code invalid_request as described in Section 3.3.

3. The authorization server MUST determine whether it supports the indicated token type. This determination is an implementation decision and MAY be based on the authorization server’s capabilities, the subject_token value, the authenticated client, or any combination thereof. If the subject_token_type is not supported by the authorization server for the given context, the authorization server MUST return an error response with the error code unsupported_token_type as described in Section 3.3.

4. The authorization server MUST validate the subject_token according to the rules for the indicated token type. The validation process MUST verify:

   * The token is properly formatted for the indicated token type
   * The token's signature (if applicable) is valid and can be verified using the appropriate cryptographic keys
   * The token was issued by a trusted issuer
   * The token has not expired
   * The token has not been revoked
   * The token is associated with the authenticated client, if client authentication is required

5. If the subject_token is invalid for any reason (e.g., malformed, expired, revoked, or does not match the subject_token_type), the authorization server MUST return an error response with the error code invalid_request as described in Section 3.3.

6. The authorization server MUST evaluate the subject_token in conjunction with the authenticated client's permissions to determine which target services are available for discovery. The specific authorization policy evaluation mechanism is implementation-specific and MAY be based on scopes, claims, resource-based access control, or other authorization models. When constructing the response, the authorization server MUST omit any target service objects or properties that would contain empty strings.

3.1.3. Request Example

The following is an example of a discovery request:

```
POST /target-discovery HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

subject_token=SlAV32hkKG...ACCESSTOKEN...
&subject_token_type=urn:ietf:params:oauth:token-type:access_token
```

3.2. Endpoint Response

The authorization server validates the request and returns a response with the discovery results. The Content-Type header of the response MUST be set to application/json.

The authorization server MAY include HTTP cache validators (such as ETag or Last-Modified headers) and expiration times (such as Cache-Control or Expires headers) in the response to enable conditional requests by the client, as specified in [RFC7234].

3.2.1. Successful Response

If the request is valid and authorized, the authorization server returns a JSON object containing a supported_targets property with an array of available token exchange targets. Each element in the array represents a target service that the client is authorized to request in a subsequent token exchange operation.

Each target service object contains the following properties:

audience REQUIRED. A string value containing an absolute URI indicating an available audience value for token exchange, as defined in Section 2.1 of [RFC8693]. Empty strings are not supported and the response MUST contain an URI. If the audience value is a URI but not a URL (e.g., a URN) and does not provide a location, the authorization server SHOULD return a resource property that contains a location.

resource OPTIONAL. A string value containing an absolute URI, or an array of string values each containing a URI, indicating available resource indicator values, as defined in Section 2 of [RFC8707]. Empty strings are not supported. If present as a string, the string MUST contain a non-empty URI. If present as an array, the array MUST contain at least one non-empty URI string and MUST NOT be empty. If no resources are available for a target service, this property MUST be omitted from the response rather than including an empty string, empty array, or null value.

scope OPTIONAL. A string value containing a space-delimited list of OAuth 2.0 scope values, as defined in Section 3.3 of [RFC6749], that are available for this target service. Each individual scope value in the list MUST conform to the scope syntax defined in Section 3.3 of [RFC6749]. Empty strings are not supported. If the property is present, the string MUST contain at least one non-empty scope value. If no scopes are available for a target service, this property MUST be omitted from the response rather than including an empty string. The authorization server

determines which scopes to return based on its authorization policy evaluation, which is implementation-specific. The scopes returned SHOULD be those that would be authorized in a subsequent token exchange request per Section 2.1 of [RFC8693].

supported_token_types OPTIONAL. An array of strings indicating the token types that may be requested for this target service in a subsequent token exchange operation. Each string MUST be a valid absolute URI. Empty strings are not supported; array elements MUST contain non-empty URI strings. If the array would be empty or contain only empty strings, this property MUST be omitted from the response. If omitted, the client may use any token type supported by the authorization server.

Extensions to this specification MAY define additional properties for the response object or for target service objects. Clients MUST ignore any properties they do not understand.

Multiple target service objects for the same audience MAY be returned with different resource(s) and scopes. The combination of audience and resource (the entire set of resources, when present) MUST be unique within the supported_targets array. That is, no two objects in the supported_targets array may have both the same audience value and the same set of resource values (when comparing arrays, the order of elements does not matter, but the complete set must match).

If no target services are available for the given subject token and client, the authorization server returns a JSON object with an empty supported_targets array: {"supported_targets": []}.

3.2.2. Response Example

The following is an example of a successful discovery response:

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

{
  "supported_targets": [
    {
      "audience": "https://api.example.com",
      "resource": ["https://api.example.com/orders", "https://api.example.com/inventory"]
,
      "scope": "orders.read inventory.read",
      "supported_token_types": [
        "urn:ietf:params:oauth:token-type:access_token"
      ]
    },
    {
      "audience": "https://billing.provider.example",
      "scope": "customer.read customer.write",
      "supported_token_types": [
        "urn:ietf:params:oauth:token-type:jwt-bearer"
      ]
    },
    {
      "audience": "https://backend-audit-service.example.com",
      "scope": "audit.read audit.write",
      "supported_token_types": [
        "urn:ietf:params:oauth:token-type:access_token"
      ]
    }
  ]
}
```

3.3. Error Response

If the request failed, the authorization server returns an error response as defined in Section 5.2 of [RFC6749]. The response MUST use the application/json media type, and the Content-Type header MUST be set to application/json. In addition to the error codes specified in Section 5.2 of [RFC6749], the following error codes may be returned:

unsupported_token_type The authorization server does not support the subject token type indicated by the subject_token_type parameter.

3.3.1. Error Response Example

The following is an example of an error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "error": "invalid_request",
  "error_description": "The subject token is invalid or expired"
}
```

4. Example

This example demonstrates a complete cross-domain identity chaining workflow described in [I-D.ietf-oauth-identity-chaining] with the addition of the token exchange target service discovery endpoint.

4.1. Step 1: Discover Target Services

The client begins with a subject access token issued by Domain A and calls the target service discovery endpoint to learn which target services are available.

4.1.1. Discovery Request

```
POST https://as.domainA.example/target-discovery HTTP/1.1
Host: as.domainA.example
Content-Type: application/x-www-form-urlencoded

client_id=client-A
&client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
&client_assertion=eyJhbGciOi...
&subject_token=SlAV32hkKG...ACCESSTOKEN...
&subject_token_type=urn:ietf:params:oauth:token-type:access_token
```

4.1.2. Discovery Response

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

{
  "supported_targets": [
    {
      "audience": "https://api.domainB.example",
      "resource": ["https://api.domainB.example/orders", "https://api.domainB.example/inv
entory"],
      "scope": "orders.read inventory.read",
      "supported_token_types": [
        "urn:ietf:params:oauth:token-type:jwt-bearer"
      ]
    }
  ]
}
```

From this response, the client learns that it may request a token exchange for the audience https://api.domainB.example with the resources https://api.domainB.example/orders and https://api.domainB.example/inventory and the scopes orders.read and inventory.read. The client also learns that JWT bearer tokens are supported for this target service.

4.2. Step 2: Discover Token Types (Optional)

If the discovery response does not include supported_token_types, or if the client needs to verify token type support, the client may query the Authorization Server Metadata of Domain B to determine which token types are supported for the target service.

4.2.1. Metadata Request

```
GET https://as.domainB.example/.well-known/oauth-authorization-server HTTP/1.1
Host: as.domainB.example
```

4.2.2. Metadata Response

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

{
  "issuer": "https://as.domainB.example",
  "token_endpoint": "https://as.domainB.example/token",
  "requested_token_types_supported": [
    "urn:ietf:params:oauth:token-type:jwt-bearer"
  ]
}
```

This confirms that Domain B supports JWT bearer tokens as a requested token type.

4.3. Step 3: Perform Token Exchange

The client now performs a token exchange with Domain A’s token endpoint, requesting a JWT for Domain B using the values discovered in the previous steps.

4.3.1. Token Exchange Request

```
POST https://as.domainA.example/token HTTP/1.1
Host: as.domainA.example
Content-Type: application/x-www-form-urlencoded

grant_type=urn:ietf:params:oauth:grant-type:token-exchange
&subject_token=SlAV32hkKG...ACCESSTOKEN...
&subject_token_type=urn:ietf:params:oauth:token-type:access_token
&requested_token_type=urn:ietf:params:oauth:token-type:jwt-bearer
&audience=https://api.domainB.example
&resource=https://api.domainB.example/orders
&resource=https://api.domainB.example/inventory
&scope=orders.read inventory.read
```

4.3.2. Token Exchange Response

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

{
  "access_token": "eyJraWQiOi...DOMAINB.JWT...",
  "issued_token_type": "urn:ietf:params:oauth:token-type:jwt-bearer",
  "token_type": "N_A",
  "expires_in": 3600,
  "scope": "orders.read inventory.read"
}
```

The client now holds a Domain-B-scoped JWT token that can be used to access the target service, derived from Domain A’s access token through the token exchange process.

5. Authorization Server Metadata

This specification defines the following authorization server metadata [RFC8414] parameter to enable clients to discover the token exchange target service discovery endpoint:

token_exchange_target_service_discovery_endpoint URL of the token

exchange target service discovery endpoint. This URL MUST use the https scheme. The authorization server SHOULD publish this metadata value.

The following is a non-normative example of the metadata:

```
{
  "issuer": "https://as.example.com",
  "token_exchange_target_service_discovery_endpoint": "https://as.example.com/target-disc
overy"
}
```

6. Security Considerations

6.1. Subject Token Validation

The authorization server MUST validate the subject token provided in the request. The validation process MUST verify that:

* The subject token is valid and not expired
* The subject token type matches the subject_token_type parameter
* The subject token is associated with the authenticated client (if client authentication is required)
* The subject token has not been revoked

If any validation fails, the authorization server MUST return an invalid_request error as described in Section 3.3.

6.2. Client Authentication

The authorization server SHOULD require client authentication for the discovery endpoint to prevent unauthorized access to authorization information. The authorization server MUST support at least one of the client authentication methods defined in Section 2.3 of [RFC6749]. If client authentication is required but not provided, the authorization server MUST return an error response with the error code invalid_client as described in Section 3.3.

6.3. Authorization Policy Enforcement

The authorization server MUST evaluate both the subject token and the client’s permissions when determining which target services to return. The server MUST only return target services that the client is authorized to request in a subsequent token exchange operation. The specific authorization policy evaluation mechanism is implementation-specific and MAY be based on scopes, claims, resource-

based access control, attribute-based access control, or other authorization models supported by the authorization server.

6.4. Information Disclosure

The discovery endpoint reveals information about which target services are available for a given subject token and client. This information could be used by an attacker to enumerate authorization relationships. To mitigate this risk:

* The authorization server SHOULD require client authentication
* The authorization server SHOULD apply rate limiting to prevent enumeration attacks
* The authorization server MAY return different results based on the authenticated client to limit information disclosure
* The authorization server SHOULD log access to the discovery endpoint for security monitoring

6.5. Token Confidentiality

The subject token is transmitted in the request. The authorization server MUST require the use of TLS as specified in Section 1.6 of [RFC6749] to protect the token in transit.

6.6. Error Handling

The authorization server MUST NOT provide detailed error messages that could aid an attacker in understanding the authorization server’s internal state or policies. Error responses SHOULD be generic and not reveal specific information about why a request failed beyond what is necessary for the client to correct the request.

7. Privacy Considerations

The discovery endpoint returns information about authorization relationships between subjects, clients, and target services. This information may be considered privacy-sensitive, as it reveals:

* Which target services a subject is authorized to access
* The scope of permissions available for each target service
* The existence of authorization relationships

To protect privacy:

* The authorization server SHOULD only return information that the authenticated client is authorized to know

* The authorization server SHOULD apply the principle of least privilege when determining which target services to return

* The authorization server SHOULD log access to the discovery endpoint in accordance with applicable privacy regulations

* The authorization server MAY provide mechanisms for subjects to control or limit the information returned by the discovery endpoint

8. IANA Considerations

8.1. OAuth Authorization Server Metadata Registry

This specification registers the following value in the IANA "OAuth Authorization Server Metadata" registry established by [RFC8414].

Metadata Name: token_exchange_target_service_discovery_endpoint

Metadata Description: URL of the token exchange target service discovery endpoint

Change Controller: IESG

Specification Document(s): [[this document]]

9. References

9.1. Normative References

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://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>.

[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>.

[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>.

[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>.

[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/info/rfc8174>.

9.2. Informative References

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, DOI 10.17487/RFC6755, October 2012, <https://www.rfc-editor.org/info/rfc6755>.

[I-D.ietf-oauth-identity-chaining] Schwenkschuster, A., Kasselman, P., Burgin, K., Jenkins, M. J., and B. Campbell, "OAuth Identity and Authorization Chaining Across Domains", Work in Progress, Internet-Draft, draft-ietf-oauth-identity-chaining-08, 9 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-chaining-08>.

[I-D.oauth-identity-assertion-authz-grant] Parecki, A., McGuinness, K., and B. Campbell, "OAuth 2.0 Identity Assertion JWT Authorization Grant", 2025, <https://datatracker.ietf.org/doc/draft-ietf-oauth-identity-assertion-authz-grant/>.

Acknowledgments

The authors would like to thank the following individuals who contributed ideas, feedback, and wording that helped shape this specification: Max Gerber

Document History

-01

* Changed discovery response to an object for extensibility with supported_targets param

* Updated references

-00

* Initial revision

Authors' Addresses

Karl McGuinness
Independent
Email: public@karlmcguinness.com

Aaron Parecki
Okta
Email: aaron@parecki.com
URI: https://aaronparecki.com