Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 24 January 2026

K. McGuinness
Independent
J. Hanson
Keycard Labs

23 July 2025

OAuth 2.0 Resource Parameter in Access Token Response
draft-mcguinness-oauth-resource-token-resp-00

Abstract

This specification defines a new parameter, resource, to be returned in OAuth 2.0 access token responses. It enables clients to confirm the intended protected resource (resource server) for the issued token. This mitigates ambiguity and certain classes of security vulnerabilities such as resource mix-up attacks, particularly in systems that use the Resource Indicators for OAuth 2.0 specification [RFC8707].

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-oauth-resource-token-resp/draft-mcguinness-oauth-resource-token-resp.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-mcguinness-oauth-resource-token-resp/.

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-oauth-resource-token-resp.

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 24 January 2026.

Copyright Notice

Table of Contents

1. Introduction

OAuth 2.0 defines a framework in which clients request access tokens from authorization servers and present them to resource servers. In deployments where multiple resources (or APIs) are involved, the Resource Indicators for OAuth 2.0 [RFC8707] specification introduced a resource request parameter that allows clients to indicate the protected resource for which the token is intended.

However, [RFC8707] does not require the authorization server to return any confirmation of the resource to which the access token applies (audience). When an authorization request includes one or more resource parameters, the authorization server can exhibit a range of behaviors depending on its capabilities and policy configuration.

An authorization server MAY:

* Ignore the resource parameter (e.g., if it does not support [RFC8707]) and audience-restrict the issued access token to a default resource or set of resources.

* Accept and honor all requested resource values, audience-restricting the issued access token to the entire set of requested resources.

* Accept a subset of the requested resource values, audience-restricting the token accordingly.

* Override the requested resource values and issue a token audience-restricted to an authorization-server-defined set of resources, based on policy or client registration.

* Reject one or more requested resource values and return an OAuth 2.0 error response with the error code invalid_target as defined in [RFC8707].

This leads to ambiguity in the client's interpretation of the token's audience, potentially resulting in *resource mix-up attacks* or incorrect token usage such as:

1. A client requests an access token for Resource A.

2. The authorization server issues a token for Resource B (intentionally or due to configuration).

3. The client unknowingly sends the token to Resource A, which may mistakenly accept it or return a misleading error.

4. The client misuses a token for a different audience, causing a confidentiality or access control breach.

This document introduces a new parameter, resource, to be returned in the access token response, so the client can validate that the issued token corresponds to the intended resource.

2. Conventions and Terminology

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.

2.1. Terminology

The terms "client", "authorization server", "resource server’, "access token", "protected resource", "authorization request", "access token request", "access token response" is defined by the OAuth 2.0 Authorization Framework specification [RFC6749].

The term "resource" is defined by the Resource Indicators for OAuth 2.0 specification [RFC8707].

The term "StringOrURI" is defined by the JWT specification [RFC7519].

3. Resource Parameter in Token Response

3.1. Syntax

Authorization servers that support this specification SHOULD include the resource parameter in successful access token responses, as defined in Section 5.1 of [RFC6749] for a valid token request.

The value of the resource parameter MUST be an array of case-sensitive strings, each containing a StringOrURI value that identifies the protected resource for which the token is valid. In the special case when the token is targeted to a single resource, the resource value MAY be a single case-sensitive string containing a StringOrURI value.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token": "2YotnFZFEjr1zCsicMWpAA",
  "token_type": "Bearer",
  "expires_in": 3600,
  "resource": "https://api.example.com/"
}
```

3.2. Semantics

* If the client included one or more resource parameters in the request per [RFC8707], the resource value in the response MUST reflect the accepted or selected resource(s).

* If the authorization server selected a default resource, it SHOULD return that selected resource in the resource parameter.

* If the requested resource is not valid for the client, user, or authorization server, then the authorization server SHOULD return an invalid_target OAuth error response code according to [RFC8707]

* If the token is not bound to a specific resource or the concept does not apply, the resource parameter SHOULD be omitted.

4. Client Processing

Clients that support this extension:

* SHOULD compare the returned resource URIs against the originally requested resource URI(s), if applicable.

* MUST treat mismatches as errors, unless the client is explicitly designed to handle token audience negotiation.

* MUST NOT use the token with a resource other than the one identified in the response.

4.1. Examples

4.1.1. Single Protected Resource

4.1.1.1. Authorization Request

Client makes an authorization request for a protected resource using the URL as the resource indicator

```
   GET /authorize?response_type=code
     &client_id=client123
     &redirect_uri=https%3A%2F%2Fclient.example%2Fcallback
     &scope=resource%3Aread
     &state=abc123
     &resource=https%3A%2F%2Fresource.example.com%2F
     &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
     &code_challenge_method=S256
   HTTP/1.1
   Host: authorization-server.example.com
```

4.1.1.2. Redirect

User successfully authenticates and delegates access to the client for the requested resource and scopes

```
HTTP/1.1 302 Found
Location: https://client.example/callback?code=SplxlOBeZQQYbYS6WxSbIA&state=abc123
```

4.1.1.3. Token Request

```
   POST /token HTTP/1.1
   Host: authorization-server.example.com
   Content-Type: application/x-www-form-urlencoded

   grant_type=authorization_code&
   code=SplxlOBeZQQYbYS6WxSbIA&
   redirect_uri=https%3A%2F%2Fclient.example%2Fcallback&
   client_id=client123&
   code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

4.1.1.4. Token Response

Resource is confirmed and unambiguous.

```
   HTTP/1.1 200 OK
   Content-Type: application/json
   Cache-Control: no-store

   {
     "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
     "token_type": "Bearer",
     "expires_in": 3600,
     "scope": "resource:read",
     "resource": "https://resource.example.com/"
   }
```

4.1.2. Multiple Protected Resources

4.1.2.1. Authorization Request

Client makes an authorization request for multiple protected resources using the URLs as the resource indicators

```
GET /authorize?response_type=code
  &client_id=client123
  &redirect_uri=https%3A%2F%2Fclient.example%2Fcallback
  &scope=resource%3Aread
  &state=abc123
  &resource=https%3A%2F%2FresourceA.example.com%2F
  &resource=https%3A%2F%2FresourceB.example.com%2F
  &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
  &code_challenge_method=S256
HTTP/1.1
Host: authorization-server.example.com
```

4.1.2.2. Redirect

User successfully authenticates and delegates access to the client for the requested resource and scopes

```
HTTP/1.1 302 Found
Location: https://client.example/callback?code=SplxlOBeZQQYbYS6WxSbIA&state=abc123
```

4.1.2.3. Token Request

Client exchanges the authorization code for an access token

```
POST /token HTTP/1.1
Host: authorization-server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=SplxlOBeZQQYbYS6WxSbIA&
redirect_uri=https%3A%2F%2Fclient.example%2Fcallback&
client_id=client123&
code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

4.1.2.4. Token Response

Both resources are confirmed and unambiguous.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "resource:read",
  "resource": [
    "https://resourceA.example.com/",
    "https://resourceB.example.com/"
  ]
}
```

4.1.3. Default Resource

4.1.3.1. Authorization Request

Client makes an authorization request without a resource indicator

```
GET /authorize?response_type=code
  &client_id=client123
  &redirect_uri=https%3A%2F%2Fclient.example%2Fcallback
  &scope=resource%3Aread
  &state=abc123
  &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
  &code_challenge_method=S256
HTTP/1.1
Host: authorization-server.example.com
```

4.1.3.2. Redirect

User successfully authenticates and delegates access to the client for the requested scopes

```
HTTP/1.1 302 Found
Location: https://client.example/callback?code=SplxlOBeZQQYbYS6WxSbIA&state=abc123
```

4.1.3.3. Token Request

```
   POST /token HTTP/1.1
   Host: authorization-server.example.com
   Content-Type: application/x-www-form-urlencoded

   grant_type=authorization_code&
   code=SplxlOBeZQQYbYS6WxSbIA&
   redirect_uri=https%3A%2F%2Fclient.example%2Fcallback&
   client_id=client123&
   code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

4.1.3.4. Token Response

Default resource is confirmed and unambiguous.

```
   HTTP/1.1 200 OK
   Content-Type: application/json
   Cache-Control: no-store

   {
     "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
     "token_type": "Bearer",
     "expires_in": 3600,
     "scope": "resource:read",
     "resource": "https://resource.example.com/"
   }
```

4.1.4. Invalid Resource

4.1.4.1. Authorization Request

```
   GET /authorize?response_type=code
     &client_id=client123
     &redirect_uri=https%3A%2F%2Fclient.example%2Fcallback
     &scope=resource%3Aread
     &state=invalid123
     &resource=https%3A%2F%2Fevil.example.net%2F
     &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
     &code_challenge_method=S256
   HTTP/1.1
   Host: authorization-server.example.com
```

4.1.4.2. Error Redirect

The server rejected the requested resource value (e.g authorization or policy violation, resource is not valid, etc).

```
  HTTP/1.1 302 Found
  Location: https://client.example/callback?error=invalid_target&error_description=Resour
ce%20not%20allowed&state=invalid123
```

5. Security Considerations

The lack of confirmation about the audience of an access token introduces a security risk in OAuth deployments, particularly when:

* A client uses multiple authorization servers and resource servers

* A client dynamically discovers an authorization server and attempts to obtain an access token at runtime via a HTTP authorization challenge with OAuth 2.0 Protected Resource Metadata [RFC9728]

* An attacker attempts a *mix-up attack* where a token intended for one resource is used at another;

* The authorization server ignores or overrides the requested resource without informing the client.

This specification addresses such issues by explicitly returning the resource URI in the token response, similar in spirit to the iss parameter defined in [RFC9207].

Clients are advised to:

* Validate the resource parameter when present;

* Avoid use of access tokens with unverified or unintended resources;

* Treat absence of the resource parameter as a potential ambiguity if the client requires strict audience binding.

6. Privacy Considerations

Returning the resource value may reveal some information about the protected resource. If the value is sensitive, the authorization server SHOULD assess whether the resource name can be safely disclosed to the client.

7. IANA Considerations

This document registers the following value in the OAuth Parameters registry established by [RFC6749].

7.1. OAuth Access Token Response Parameters Registry

Name	Description	Specification
resource	Resource to which the access token applies	This document

Table 1

8. 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>.

[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>.

[RFC9728] Jones, M.B., Hunt, P., and A. Parecki, "OAuth 2.0 Protected Resource Metadata", RFC 9728, DOI 10.17487/RFC9728, April 2025, <https://www.rfc-editor.org/info/rfc9728>.

[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>.

[RFC9207] Meyer zu Selhausen, K. and D. Fett, "OAuth 2.0 Authorization Server Issuer Identification", RFC 9207, DOI 10.17487/RFC9207, March 2022, <https://www.rfc-editor.org/info/rfc9207>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[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/info/rfc7519>.

[RFC9700] Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "Best Current Practice for OAuth 2.0 Security", BCP 240, RFC 9700, DOI 10.17487/RFC9700, January 2025, <https://www.rfc-editor.org/info/rfc9700>.

[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>.

Appendix A. Additional Examples

A.1. Requesting a token for a dynamically discovered protected resource

The following example details the need for the resource parameter when a client dynamically discovers an authorization server and obtains an access token using [RFC9728] and [RFC8414]

Client attempts to access a protected a resource without a valid access token

```
GET /resource
Host: api.example.com
Accept: application/json
```

Client is challenged for authentication

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer resource_metadata=
  "https://api.example.com/.well-known/oauth-protected-resource"
```

Client fetches the resource’s OAuth 2.0 Protected Resource Metadata per [RFC9728] to dynamically discover an authorization server that can issue an access token for the resource.

```
   GET /.well-known/oauth-protected-resource
   Host: api.example.com
   Accept: application/json

   HTTP/1.1 200 Ok
   Content-Type: application/json

   {
      "resource":
        "https://api.example.com/resource",
      "authorization_servers":
        [ "https://authorization-server.example.com/" ],
      "bearer_methods_supported":
        ["header", "body"],
      "scopes_supported":
        ["resource.read", "resource.write"],
      "resource_documentation":
        "https://api.example.com/resource_documentation.html"
    }
```

Client discovers the Authorization Server configuration per [RFC8414]

```
GET /.well-known/oauth-authorization-server
Host: authorization-server.example.com
Accept: application/json

HTTP/1.1 200 Ok
Content-Type: application/json

{
  "issuer": "https://authorization-server.example.com/",
  "authorization_endpoint": "https://authorization-server.example.com/oauth2/authorize",
  "token_endpoint": "https://authorization-server.saas.com/oauth2/token",
  "jwks_uri": "https://authorization-server.example.com/oauth2/keys",
  "scopes_supported": [
    "resource.read", "resource.write"
  ],
  "response_types_supported": [
    "code"
  ],
  "grant_types_supported": [
    "authorization_code", "refresh_token"
  ],
  ...
}
```

Client makes an authorization request for the resource

```
   GET /oauth2/authorize?response_type=code
     &client_id=client123
     &redirect_uri=https%3A%2F%2Fclient.example%2Fcallback
     &scope=resource%3Aread
     &state=abc123
     &resource=https%3A%2F%api.example.com%2Fresource
     &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
     &code_challenge_method=S256
   HTTP/1.1
   Host: authorization-server.example.com

   User successfully authenticates and delegates access to the client
   for the requested resource and scopes

HTTP/1.1 302 Found
Location: https://client.example/callback?code=SplxlOBeZQQYbYS6WxSbIA&state=abc123

   Client exchanges the authorization code for an access token

   POST /oauth2/token HTTP/1.1
   Host: authorization-server.example.com
   Content-Type: application/x-www-form-urlencoded

   grant_type=authorization_code&
   code=SplxlOBeZQQYbYS6WxSbIA&
   redirect_uri=https%3A%2F%2Fclient.example%2Fcallback&
   client_id=client123&
   code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk

   Client obtains an access token for the resource

   HTTP/1.1 200 OK
   Content-Type: application/json
   Cache-Control: no-store

   {
     "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
     "token_type": "Bearer",
     "expires_in": 3600,
     "scope": "resource:read",
     "resource": "https://api.example.com/resource"
   }

   Client verifies the requested a token explicitly bound to the
   discovered resource.
```

Acknowledgments

This proposal builds on prior work in OAuth 2.0 extensibility and security analysis, particularly [RFC8707], [RFC9700], and [RFC9207].

Document History

-00

* Initial revision

Authors' Addresses

Karl McGuinness
Independent
Email: public@karlmcguinness.com

Jared Hanson
Keycard Labs
Email: jared@keycard.ai
URI: https://keycard.ai