Web Authorization Protocol
Internet-Draft
Updates: 9728 (if approved)
Intended status: Standards Track
Expires: 28 August 2026

K. McGuinness
Independent
A. Parecki
Okta

24 February 2026

Update to OAuth 2.0 Protected Resource Metadata Resource Identifier Validation
draft-mcguinness-oauth-rfc9728bis-01

Abstract

RFC 9728 defines OAuth 2.0 Protected Resource Metadata, enabling clients to dynamically discover the authorization requirements of protected resources. Section 3.3 of RFC 9728 requires that when protected resource metadata is obtained via a WWW-Authenticate challenge, the resource value in the metadata MUST exactly match the URL the client used to access the protected resource, limiting the applicability of the specification.

This document updates the resource validation rule in Section 3.3 of RFC 9728 to permit the resource value to be any URI that shares the same TLS origin (scheme, host, and port) as the requested URL and whose path is a prefix of the request URL path. This enables a wider range of use cases for the WWW-Authenticate response. All other aspects of RFC 9728 remain unchanged.

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-rfc9728bis/draft-mcguinness-oauth-rfc9728bis.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-mcguinness-oauth-rfc9728bis/.

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-rfc9728bis.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document updates one specific aspect of OAuth 2.0 Protected Resource Metadata [RFC9728]: the resource validation rule defined in Section 3.3.

Section 3.3 of [RFC9728] requires that when a client retrieves protected resource metadata from a URL obtained via a WWW-Authenticate challenge, the resource value in the metadata response MUST be identical to the URL the client used to make the request that triggered the challenge. This exact-match rule is intended to prevent a malicious resource from directing clients to metadata that impersonates a different resource.

Consider a resource server at https://api.example.com that exposes the following protected resources:

* https://api.example.com/accounts

* https://api.example.com/transactions

* https://api.example.com/profile

All three protected resources are served by the same authorization server (https://as.example.com) and accept tokens with the same audience (https://api.example.com).

Under the current Section 3.3 rule in [RFC9728], when a client calls https://api.example.com/transactions without a token, the resource server responds:

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

The client retrieves the metadata, which per Section 3.3 of [RFC9728] MUST contain a resource value identical to the request URL:

```
{
  "resource": "https://api.example.com/transactions",
  "authorization_servers": ["https://as.example.com"],
  ...
}
```

The client then requests a token using https://api.example.com/transactions as the resource parameter per [RFC8707]. The authorization server may issue a token whose audience is https://api.example.com (covering all protected resources on the server), but the client has no standardized way to know this.

When the client subsequently needs to access https://api.example.com/accounts, it faces one of the following suboptimal outcomes:

1. It repeats the entire discovery and token-request flow for the new protected resource, wasting a round trip to the authorization server.

2. It speculatively reuses the existing token, which may work but is not grounded in any protocol signal from the resource server.

3. The authorization server must understand and enumerate every per-URL resource identifier that maps to a given audience, increasing configuration complexity.

To avoid these outcomes, this document relaxes the exact-match requirement for resource values obtained via WWW-Authenticate discovery to a same-origin, path-prefix match. See Section 4 for the normative specification. Appendix A provides a non-normative example illustrating how the updated rule applies to APIs with fine-grained resource URLs.

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. Terminology

This document uses the following terms as defined in the referenced specifications:

resource identifier: An HTTPS URI that identifies a protected resource or set of protected resources, as defined in Section 1.2 of [RFC9728] and [RFC8707].

TLS origin: The combination of scheme, host, and port derived from a URI, as defined in Section 4.3.2 of [RFC9110] for the https scheme. Two URIs share the same TLS origin if and only if their scheme, host, and port (after applying default port rules) are identical.

path prefix: A path that matches the beginning of another path on a segment boundary. See Section 4.3 for the precise definition.

request URL: The URL the client used to make the request that triggered the WWW-Authenticate challenge.

4. Update to RFC 9728 Section 3.3

This section contains the normative change to [RFC9728]. It updates only the resource validation rule in Section 3.3 of [RFC9728] that applies when metadata is retrieved via a WWW-Authenticate challenge. All other requirements in Section 3.3 and the rest of [RFC9728] remain in effect.

4.1. Original Rule (Replaced)

Section 3.3 of [RFC9728] states:

> If the protected resource metadata was retrieved from a URL returned by the protected resource via the WWW-Authenticate resource_metadata parameter, then the resource value returned MUST be identical to the URL that the client used to make the request to the resource server. If these values are not identical, the data contained in the response MUST NOT be used.

This document replaces the above requirement with the updated rule in Section 4.2.

4.2. Updated Validation Rule

When a client retrieves protected resource metadata from a URL obtained via the resource_metadata parameter in a WWW-Authenticate challenge (as defined in Section 5 of [RFC9728]), the client MUST verify ALL of the following conditions using the resource value from the metadata response and the URL the client used to make the request that triggered the challenge:

1. The scheme of the resource value MUST be https.

2. The host of the resource value MUST be identical (using case-insensitive comparison) to the host of the request URL.

3. The port of the resource value MUST be identical to the port of the request URL. If either URL omits the port, the default port for the https scheme (443) MUST be used for comparison.

4. The path of the resource value MUST be a prefix of the path of the request URL, as defined in Section 4.3.

If any of these conditions are not met, the client MUST NOT use the metadata, consistent with the security requirements of [RFC9728].

Note that when the resource value is identical to the request URL, all four conditions are trivially satisfied. This means the updated rule is fully backwards compatible with the original exact-match rule in [RFC9728]: any metadata that was valid under the original rule remains valid under this update.

This update only applies to metadata retrieved via a WWW-Authenticate challenge. The validation rule for metadata retrieved directly from a well-known URI is NOT changed by this document; the resource value MUST still be identical to the resource identifier from which the well-known URI was derived, as specified in Section 3.3 of [RFC9728].

4.3. Path Prefix Matching

This section defines the path prefix matching algorithm referenced by condition 4 of the updated validation rule in Section 4.2.

The path of the resource value is a prefix of the path of the request URL if any of the following conditions hold:

1. The paths are identical (exact match).

2. The resource path ends with / and the request URL path starts with the resource path. For example, a resource path of /api/v1/ is a prefix of a request URL path of /api/v1/accounts.

3. The resource path does not end with /, and the request URL path is the resource path followed by / and optionally additional segments. For example, a resource path of /api/v1 is a prefix of request URL paths /api/v1/ and /api/v1/accounts.

Matching MUST occur on segment boundaries. A resource path MUST NOT be treated as a prefix if the match would split a path segment. For example, a resource path of /api/v1 MUST NOT match a request URL path of /api/v10 or /api/v1admin, because v10 and v1admin are distinct path segments from v1.

Paths MUST be compared in their URI-normalized form per [RFC3986]. Percent-encoded characters MUST be decoded before comparison. An empty or absent path MUST be treated as / for comparison purposes.

The following table illustrates the matching behavior:

resource path	Request URL path	Match?
/	/	Yes
/	/accounts	Yes
/	/api/v1/accounts	Yes
/api	/api/v1/accounts	Yes
/api/	/api/v1/accounts	Yes
/api/v1	/api/v1/accounts	Yes
/api/v1	/api/v1/	Yes
/api/v1/	/api/v1/accounts	Yes
/api/v1	/api/v1	Yes
/api/v1	/api/v10	*No*
/api/v1	/api/v1admin	*No*
/api/v1	/api/v2/accounts	*No*
/api/v1	/other	*No*
/accounts	/transactions	*No*

Table 1

5. Examples

This section is non-normative. It illustrates how the updated validation rule in Section 4 applies in practice.

5.1. Host-Level Resource Identifier

A resource server at https://api.example.com advertises a host-level resource identifier covering all of its protected resources.

When a client requests https://api.example.com/transactions without a token:

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

The metadata response contains:

```
{
  "resource": "https://api.example.com/",
  "authorization_servers": ["https://as.example.com"],
  "scopes_supported": ["accounts.read", "transactions.read",
    "profile.read"]
  ...
}
```

The client validates that https://api.example.com/ shares the same TLS origin as https://api.example.com/transactions and that the path / is a prefix of /transactions.

5.2. Path-Level Resource Identifier

A resource server at https://platform.example.com exposes two independent sets of protected resources under different path prefixes.

When a client requests https://platform.example.com/api/v1/transactions without a token:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer resource_metadata="https://platform.example.com/.well-known/oaut
h-protected-resource/api/v1"
```

The metadata response contains:

```
{
  "resource": "https://platform.example.com/api/v1",
  "authorization_servers": ["https://as.example.com"],
  "scopes_supported": ["transactions.read", "accounts.read"]
  ...
}
```

The client validates that https://platform.example.com/api/v1 shares the same TLS origin as https://platform.example.com/api/v1/transactions and that the path /api/v1 is a prefix of /api/v1/transactions on a segment boundary.

A subsequent request to https://platform.example.com/api/v2/reports would NOT match the resource https://platform.example.com/api/v1 because /api/v1 is not a prefix of /api/v2/reports. The client would need to perform a separate discovery for the /api/v2 protected resources.

6. Client Token Caching Guidance

This section is non-normative.

A client MAY maintain a token cache keyed by the tuple (authorization_server, resource), where authorization_server is the issuer identifier from which the token was obtained and resource is the resource value from the protected resource metadata. A cached token MAY be reused for requests to any URL whose path falls under the same resource path prefix on the same TLS origin, provided the token was obtained from the same authorization server, has not expired, and carries sufficient scopes.

A client MAY also optimistically reuse a cached token when accessing a URL under the same resource path prefix before performing metadata discovery for that URL. If the resource server rejects the token (e.g., with a 401 response), the client falls back to the standard discovery flow. The client should not assume that all protected resources under a given path prefix accept the same token.

7. Security Considerations

The security considerations of [RFC9728] (Section 7) continue to apply in full. This section describes how the updated validation rule in Section 4 interacts with those considerations.

7.1. Preservation of Origin Security Boundary

Section 7.3 of [RFC9728] describes impersonation attacks where an adversary publishes metadata claiming to represent a legitimate resource. The updated validation rule in Section 4.2 maintains the fundamental security property that prevents these attacks: the TLS origin check (conditions 1-3) ensures that the resource value is authoritative for the same server that issued the challenge. An attacker who controls https://evil.example.com cannot cause a client to accept metadata with a resource value of https://api.example.com/, because the origins differ.

7.2. Path-Level Isolation

Some deployments host multiple independent services on the same origin, distinguished only by path (e.g., a shared hosting environment where https://shared.example.com/serviceA and https://shared.example.com/serviceB are operated by different tenants). The path-prefix matching rule in this document supports these deployments: each service can advertise its own path-scoped resource identifier (e.g., https://shared.example.com/serviceA) that covers only protected resources under that path prefix.

Resource servers in multi-tenant environments SHOULD use path-specific resource identifiers that maintain the necessary isolation between tenants. A resource identifier of https://shared.example.com/serviceA will not match requests to https://shared.example.com/serviceB/resource because /serviceA is not a path prefix of /serviceB/resource. This ensures that tokens obtained for one service cannot be inadvertently reused for another service on the same origin.

7.3. Token Scope and Audience

Relaxing the resource matching rule does not change the authorization server’s responsibility to enforce audience restrictions on issued tokens. The authorization server remains the authority on what resource(s) a token is valid for. A client’s use of a resource identifier (whether host-level or path-scoped) as the resource parameter in a token request is a signal to the authorization server, which MAY issue a token with a narrower or broader audience at its discretion.

7.4. Sender-Constrained Tokens

Deployments using sender-constrained tokens (e.g., DPoP [RFC9449]) are compatible with the relaxed matching rule defined in this document. The sender constraint binds the token to the client, not to a specific protected resource, so token reuse across protected resources on the same origin does not weaken the sender-constraint security property.

7.5. Metadata Substitution Within an Origin

The updated validation rule in Section 4.2 permits a protected resource to direct clients to metadata with a resource value that differs from the URL of the protected resource (but shares the same origin and satisfies the path-prefix condition). This means that a compromised protected resource on an origin could direct clients to metadata controlled by another protected resource on the same origin. This risk is inherent to same-origin trust and is no different from the existing web security model, where all content on the same origin shares a trust boundary. This risk also exists under the original [RFC9728] rule, since a compromised protected resource can already return arbitrary WWW-Authenticate challenges.

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[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>.

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

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/info/rfc9449>.

[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>.

[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

[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>.

[I-D.mcguinness-oauth-resource-token-resp] McGuinness, K. and J. Hanson, "OAuth 2.0 Resource Parameter in Access Token Response", Work in Progress, Internet-Draft, draft-mcguinness-oauth-resource-token-resp-01, 21 December 2025, <https://datatracker.ietf.org/doc/html/draft-mcguinness-oauth-resource-token-resp-01>.

Appendix A. Resource Discovery for Fine-Grained APIs

This non-normative example illustrates a common deployment pattern where the original exact-match validation rule in Section 3.3 of [RFC9728] creates a barrier to adoption, and shows how the updated rule in Section 4 resolves it.

A.1. Exact-Match Constraint

Consider a resource server that exposes individual user resources at unique URLs of the form https://example.com/users/{id}.

A client that does not yet hold an access token makes a GET request to the resource. The resource server responds with 401 Unauthorized and a WWW-Authenticate challenge identifying the Protected Resource Metadata document:

```
GET /users/100 HTTP/1.1
Host: example.com

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

Following the discovery procedure in Section 3 of [RFC9728], the client retrieves the metadata document:

```
GET /.well-known/oauth-protected-resource HTTP/1.1
Host: example.com

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

{
  "resource": "https://example.com/users/",
  "authorization_servers": ["https://as.example.com"],
  ...
}
```

Under the original Section 3.3 rule of [RFC9728], the resource value MUST be identical to the request URL https://example.com/users/100. Because the metadata instaed contains an invalid value of https://example.com/users/, the client MUST NOT use the metadata.

To satisfy the original exact-match requirement, a resource server has two options.

Option 1: A distinct metadata document per resource URL.

Each resource URL requires its own Protected Resource Metadata document, for example:

```
https://example.com/users/100/.well-known/...
https://example.com/users/101/.well-known/...
https://example.com/users/102/.well-known/...
...
```

For APIs where the set of resource URLs is not fixed in advance, this creates an unbounded number of metadata documents to provision, serve, and keep consistent. In practice this results in increased storage overhead, cache fragmentation, and operational complexity that scales linearly with the number of resources.

Option 2: Out-of-band knowledge of a representative resource URL.

To work around the exact-match constraint, a client could first make a GET request to a "parent" resource (e.g., https://example.com/users) in order to trigger a 401 Unauthorized response whose associated metadata contains a resource value of https://example.com/users. The client would perform discovery against that URL, obtain an access token, and then use the access token to access https://example.com/users/100.

```
GET /users HTTP/1.1
Host: example.com

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

This approach requires out-of-band knowledge of which URL to request first, which is not conveyed by the protocol. It also introduces an additional round trip before the client can access the originally desired resource, increasing latency and complexity for every new client interaction.

A.2. Resolution with the Updated Rule

The updated validation rule in Section 4.2 permits the resource value to be any URI that shares the same TLS origin as the request URL and whose path is a prefix of the request URL path. This allows a single metadata document to cover a collection of related resources, enabling straightforward discovery with no additional round trips or out-of-band coordination.

A client that does not yet hold an access token makes a GET request to the resource. The resource server responds with 401 Unauthorized and a WWW-Authenticate challenge:

```
GET /users/100 HTTP/1.1
Host: example.com

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

The client retrieves the metadata document:

```
GET /.well-known/oauth-protected-resource HTTP/1.1
Host: example.com

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

{
  "resource": "https://example.com/users/",
  "authorization_servers": ["https://as.example.com"],
  ...
}
```

The client validates that https://example.com/users/ shares the same TLS origin as https://example.com/users/100 and that the path /users/ is a prefix of /users/100 on a segment boundary. Both conditions are satisfied, so the client accepts the metadata.

The client uses https://example.com/users/ as the resource indicator (per [RFC8707]) when requesting an access token from the authorization server. The resulting access token can be reused for subsequent requests to https://example.com/users/101, https://example.com/users/102, and so on without repeating the discovery or token-request flow.

Acknowledgments

The authors would like to thank the members of the OAuth Working Group and MCP Authorization Working Group for their feedback and discussion on the challenges of dynamic resource discovery and cross-resource token reuse.

Document History

-01

* Resolved merge error for publication

-00

* Initial version.

Authors' Addresses

Karl McGuinness
Independent
Email: public@karlmcguinness.com

Aaron Parecki
Okta
Email: aaron.parecki@okta.com