

Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 11 April 2026

A. Parecki
Okta
E. Smith
8 October 2025

OAuth Client ID Metadata Document
draft-ietf-oauth-client-id-metadata-document-00

Abstract

This specification defines a mechanism through which an OAuth client can identify itself to authorization servers, without prior dynamic client registration or other existing registration. This is through the usage of a URL as a client_id in an OAuth flow, where the URL refers to a document containing the necessary client metadata, enabling the authorization server to fetch the metadata about the client as needed.

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://drafts.oauth.net/draft-ietf-oauth-client-id-metadata-document/draft-ietf-oauth-client-id-metadata-document.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-oauth-client-id-metadata-document/>.

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/oauth-wg/draft-ietf-oauth-client-id-metadata-document>.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 April 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Conventions and Definitions | 3 |
| 3. Client Identifier | 3 |
| 4. Client Information Discovery | 4 |
| 4.1. Client Metadata | 4 |
| 4.2. Client Metadata Documents for Development Purposes | 5 |
| 4.3. Metadata Discovery Errors | 5 |
| 4.4. Metadata Caching | 5 |
| 4.5. Redirect URL Registration | 6 |
| 5. Authorization Server Metadata | 6 |
| 6. Security Considerations | 6 |
| 6.1. Relationship between redirect_uris and client_id or client_uri | 7 |
| 6.2. Client Authentication | 7 |
| 6.3. Changes in Client Keys | 8 |
| 6.4. OAuth Phishing Attacks | 8 |
| 6.5. Server Side Request Forgery (SSRF) Attacks | 8 |
| 6.6. Maximum Response Size for Client Metadata Documents | 9 |
| 6.7. Displaying Logos to End-Users | 9 |
| 6.8. Client ID Domain Trust | 9 |
| 7. IANA Considerations | 9 |
| 7.1. OAuth Authorization Server Metadata Registry | 9 |
| 8. References | 10 |
| 8.1. Normative References | 10 |
| 8.2. Informative References | 10 |

| | |
|------------------------------|----|
| Acknowledgments | 11 |
| Document History | 12 |
| Authors' Addresses | 12 |

1. Introduction

In order for an OAuth 2.0 [RFC6749] client to utilize an OAuth 2.0 authorization server, the client needs to establish a unique identifier, and needs to provide the server with metadata about the application, such as the application name, icon and redirect URIs. In cases where a client is interacting with authorization servers that it has no relationship with, manual registration is impossible.

While Dynamic Client Registration [RFC7591] can provide a method for a previously unknown client to establish itself at an authorization server and obtain a client identifier, this is not always practical in some deployments and can create additional challenges around management of the registration data and cleanup of inactive clients.

This specification describes how an OAuth 2.0 client can publish its own registration information and avoid the need for pre-registering at each authorization server.

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. Client Identifier

This specification defines the client identifier as a URL with the following restrictions. Client identifier URLs MUST have an "https" scheme, MUST contain a path component, MUST NOT contain single-dot or double-dot path segments, MUST NOT contain a fragment component and MUST NOT contain a username or password. Client identifier URLs SHOULD NOT include a query string component, and MAY contain a port.

This specification places no restrictions on what URL is used as a client identifier. A short URL is RECOMMENDED, since the URL may be displayed to the end user in the authorization interface or in management interfaces. Usage of a stable URL that does not frequently change for the client is also RECOMMENDED.

4. Client Information Discovery

One purpose of registering clients at the authorization server is so that the authorization server has additional information about the client that can be used during an OAuth flow, such as presenting information about the client to the user in an authorization consent screen, for example the client name and logo.

The authorization server **SHOULD** fetch the document indicated by the `client_id` to retrieve the client registration information.

4.1. Client Metadata

The client metadata document URL is a JSON document containing the metadata of the client. The client metadata values are the values defined in the OAuth Dynamic Client Registration Metadata OAuth Parameters registry <https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml#client-metadata> (<https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml#client-metadata>).

The client metadata document **MUST** contain a `client_id` property whose value **MUST** match the URL of the document using simple string comparison as defined in [RFC3986] Section 6.2.1.

The client metadata document **MAY** define additional properties in the response. The client metadata document **MAY** also be served with more specific content types as long as the response is JSON and conforms to `application/<AS-defined>+json`.

As there is no way to establish a shared secret to be used with client metadata documents, the following restrictions apply on the contents of the client metadata document:

- * the `token_endpoint_auth_method` property **MUST NOT** include `client_secret_post`, `client_secret_basic`, `client_secret_jwt`, or any other method based around a shared symmetric secret.
- * the `client_secret` and `client_secret_expires_at` properties **MUST NOT** be used

See Section 6.2 for more details.

Other specifications **MAY** place additional restrictions on the contents of the client metadata document accepted by authorization servers implementing their specification, for instance, preventing the registration of confidential clients by requiring the `token_endpoint_auth_method` property be set to "none".

TBD: We may want a property such as `client_id_expires_at` for indicating that the client is ephemeral and not valid after a given timestamp, especially for documents issued by a service for development purposes.

4.2. Client Metadata Documents for Development Purposes

An authorization server may have restrictions on what it accepts as valid `redirect_uris`, for instance, limiting them to the same-origin as the `client_id` or `client_uri` properties. However, if an authorization server does place additional restrictions on the accepted `redirect_uris` then it **SHOULD** provide at least one Client ID Metadata Document Service (described below) which is exempt from these restrictions.

When developing applications against an authorization server which uses this specification, developers often encounter the issue of "how do I serve a Client ID Metadata Document at a publicly accessible `https` URL whilst developing my application on my `localhost`?".

To enable developers to author applications on their machines, without exposing their machines to the public internet, the usage of Client ID Metadata Document Services by the authorization server is **RECOMMENDED**.

A Client ID Metadata Document Service is a web service through which developers can acquire a stable URL to a Client ID Metadata Document. This service **MAY** expire clients from time to time, and **MAY** require developers to provide additional information about the client being developed.

By providing at least one Client ID Metadata Document Service, an authorization server can enable developers to create applications, and still indicate to non-technical people that the client that they are about to authorize is currently under-development and may not be trustworthy or secure.

4.3. Metadata Discovery Errors

If fetching the metadata document fails, the authorization server **SHOULD** abort the authorization request.

4.4. Metadata Caching

The authorization server **MAY** cache the client metadata it discovers at the client metadata document URL.

The authorization server SHOULD respect HTTP cache headers [RFC9111] when caching client metadata, but MAY define its own upper and/or lower bounds on an acceptable cache lifetime as well.

The authorization server MUST NOT cache error responses. The authorization server also MUST NOT cache documents which are invalid or malformed.

4.5. Redirect URL Registration

According to [RFC9700], the authorization server MUST require registration of redirect URIs, and MUST ensure that the redirect URI in a request is an exact match of a registered redirect URI.

This method of client information discovery establishes a registered redirect URI with the authorization server which is used when comparing the redirect URI in an authorization request against the registered redirect URIs.

5. Authorization Server Metadata

Authorization servers that publish Authorization Server Metadata [RFC8414] MUST include the following property to signal support for client metadata documents as described in this specification.

`client_id_metadata_document_supported`: OPTIONAL. Boolean value specifying whether the authorization server supports retrieving client metadata from a `client_id` URL as described in this specification.

This enables clients to avoid sending the user to a dead end, by only redirecting the user to an authorization server that supports this specification. Otherwise, the client would redirect the user and the user would be met with an error about an invalid client as described by Section 4.1.2.1 of [RFC6749].

6. Security Considerations

In addition to the security considerations in OAuth 2.0 Core [RFC6749], and OAuth 2.0 Threat Model and Security Considerations [RFC6819], and [RFC9700] the additional considerations apply.

6.1. Relationship between redirect_uris and client_id or client_uri

An authorization server MAY impose restrictions or relationships between the redirect_uris and the client_id or client_uri properties, for example to restrict the redirect_uri to the same-origin as the Client ID Metadata Document. Without restrictions like these, there are potential trust and safety issues where the client attempts to impersonate a more well-known client or otherwise act in a way which is malicious or puts the end-user at risk.

Having no restrictions on the relationship between redirect_uris and client_id or client_uri was a common practice with [Solid-OIDC]'s Client ID Documents, so this ability is preserved for backwards compatibility between [Solid-OIDC] and this specification.

Some restrictions on redirect_uris can make developer usage of Client ID Metadata Documents difficult. The section Section 4.2 contains recommendations for enabling development usage of Client ID Metadata Documents for authorization servers that impose restrictions on the redirect_uri.

6.2. Client Authentication

Since the client establishes its own registration data at the authorization server, prior coordination of client credentials is not possible. However, clients MAY establish credentials at the authorization server by using authentication methods that use public/private key pairs, by publishing the public key in their metadata document.

For example, the client MAY include the following properties in its metadata document to establish a public key and advertise the private_key_jwt authentication method defined in [OpenID]:

```
{
  ...
  "token_endpoint_auth_method": "private_key_jwt",
  "jwks_uri": "https://client.example.com/jwks.json"
  ...
}
```

This establishes this client as a confidential client, and any communication with the authorization server MUST include client authentication of the registered type.

The particular method of how the client manages the private key is out of scope of this specification, but may include manual provisioning or methods such as Attestation Based Client

Authentication [I-D.draft-ietf-oauth-attestation-based-client-auth]. For example, the client developer could run a Client Attester Backend, using a native application's platform-specific APIs to authenticate to the backend service, where the private key corresponding to the `jwt_uri` key is managed by the backend service. This would allow a mobile app to request JWTs from the backend service that the mobile app could then use as client authentication to the authorization server.

6.3. Changes in Client Keys

If the authorization server notices that the `jwt_uri` or the contents at the `jwt_uri` have changed compared to the last time it fetched the metadata, the authorization server MAY take actions such as revoking any tokens issued to this client, or revoking the user's consent for this client. The particular actions to take are left up to the discretion of the authorization server based on its own risk assessment.

6.4. OAuth Phishing Attacks

Authorization servers SHOULD fetch the `client_id` metadata document provided in the authorization request in order to provide users with additional information about the request, such as the application name and logo. If the server does not fetch the client metadata document, then it SHOULD take additional measures to ensure the user is provided with as much information as possible about the request.

The authorization server SHOULD display the hostname of the `client_id` on the authorization interface, in addition to displaying the fetched client information if any. Displaying the hostname helps users know that they are authorizing the expected application.

If fetching the client metadata document fails for any reason, the `client_id` URL is the only piece of information the user has as an indication of which application they are authorizing.

6.5. Server Side Request Forgery (SSRF) Attacks

Authorization servers fetching the client metadata document and resolving URLs located in the metadata document should be aware of possible SSRF attacks. Authorization servers SHOULD avoid fetching any URLs using private or loopback addresses and consider network policies or other measures to prevent making requests to these addresses. Authorization servers SHOULD also be aware of the possibility that URLs might be non-http-based URI schemes which can lead to other possible SSRF attack vectors.

6.6. Maximum Response Size for Client Metadata Documents

Authorization servers SHOULD limit the response size when fetching the client metadata document, as to avoid denial of service attacks against the authorization server by consuming excessive resources (memory, disk, database). The recommended maximum response size for client metadata documents is 5 kilobytes.

6.7. Displaying Logos to End-Users

Authorization servers that wish to make use of the `logo_uri` property within client metadata document SHOULD prefetch the file at `logo_uri` and cache it for the cache duration of the client metadata document. This allows for moderation tools to verify the file contents (e.g., preventing usage of logos that look like other logos), as well as preventing the logo from being dynamically changed to confuse an end-user.

Caching of the `logo_uri` response can additionally prevent cross-domain tracking through the `logo_uri` being requested by the client, since the cached file would be served not from the remote URI but instead from a URI that the Authorization server trusts.

6.8. Client ID Domain Trust

The authorization server MAY choose to have its own heuristics and policies around the trust of domain names used as client IDs.

For example, the authorization server could require that the first 100 users to authorize a `client_id` see an additional warning screen before the OAuth consent screen. The authorization server could check attributes of the domain reputation, such as how recently the domain was registered, and put up extra warnings for new domains.

7. IANA Considerations

7.1. OAuth Authorization Server Metadata Registry

The following authorization server metadata value is defined by this specification and registered in the IANA "OAuth Authorization Server Metadata" registry established in OAuth 2.0 Authorization Server Metadata [RFC8414].

- * Metadata Name: `client_id_metadata_document_supported`:
- * Metadata Description: JSON boolean value specifying whether the authorization server supports retrieving client metadata from a `client_id` URL.

- * Change Controller: IETF
- * Specification Document: Section 5 of [draft-ietf-oauth-client-id-metadata-document-00]

8. References

8.1. Normative References

- [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/rfc/rfc2119>>.
- [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/rfc/rfc3986>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/rfc/rfc6749>>.
- [RFC6819] Lodderstedt, T., Ed., McGloin, M., and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", RFC 6819, DOI 10.17487/RFC6819, January 2013, <<https://www.rfc-editor.org/rfc/rfc6819>>.
- [RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <<https://www.rfc-editor.org/rfc/rfc7591>>.
- [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/rfc/rfc8174>>.
- [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/rfc/rfc8414>>.
- [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/rfc/rfc9700>>.

8.2. Informative References

- [I-D.draft-ietf-oauth-attestation-based-client-auth]
Looker, T., Bastian, P., and C. Bormann, "OAuth 2.0 Attestation-Based Client Authentication", Work in Progress, Internet-Draft, draft-ietf-oauth-attestation-based-client-auth-07, 15 September 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-oauth-attestation-based-client-auth-07>>.
- [IndieAuth]
Parecki, A., "IndieAuth", 12 February 2022, <<https://indieauth.spec.indieweb.org/>>.
- [OpenID] Sakimura, N., Bradley, J., Jones, M., Medeiros, B. de., and C. Mortimore, "OpenID Connect Core 1.0", 15 December 2023, <https://openid.net/specs/openid-connect-core-1_0.html>.
- [OpenID.Federation]
Hedberg, R., Jones, M.B., Solberg, A., Bradley, J., Marco, G. D., and V. Dzhuvinov, "OpenID Federation 1.0", 17 May 2024, <https://openid.net/specs/openid-federation-1_0.html>.
- [RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <<https://www.rfc-editor.org/rfc/rfc7523>>.
- [RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <<https://www.rfc-editor.org/rfc/rfc9111>>.
- [Solid-OIDC]
Coburn, A., elf Pavlik, and D. Zagidulin, "Solid-OIDC", 28 March 2022, <<https://solidproject.org/TR/2022/oidc-20220328>>.

Acknowledgments

The idea of using URIs as the `client_id` in OAuth based authorization requests is not new, and has previously been specified in varying ways by [IndieAuth], [Solid-OIDC], and [OpenID.Federation]. This specification is largely inspired by the work of Aaron Coburn, elf Pavlik, and Dmitri Zagidulin in their [Solid-OIDC] specification which defined dereferenceable Client Identifier Documents.

The authors would like to thank the following people for their contributions and reviews of this specification: Brian Campbell, Dick Hardt, Leif Johansson, Pieter Kasselmann, Bryan Newbold, Matthieu Sieben, and Filip Skokan.

Document History

(This appendix to be deleted by the RFC editor in the final specification.)

-00

* Initial draft

Authors' Addresses

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

Emelia Smith
Email: emelia@brandedcode.com
URI: <https://thisismissem.social>