Internet Engineering Task Force (IETF)
Request for Comments: 8414
Category: Standards Track
ISSN: 2070-1721

M. Jones
Microsoft
N. Sakimura
NRI
J. Bradley
Yubico

June 2018

OAuth 2.0 Authorization Server Metadata

Abstract

This specification defines a metadata format that an OAuth 2.0 client can use to obtain the information needed to interact with an OAuth 2.0 authorization server, including its endpoint locations and authorization server capabilities.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8414.

Table of Contents

1. Introduction

This specification generalizes the metadata format defined by "OpenID Connect Discovery 1.0" [OpenID.Discovery] in a way that is compatible with OpenID Connect Discovery while being applicable to a wider set of OAuth 2.0 use cases. This is intentionally parallel to the way that "OAuth 2.0 Dynamic Client Registration Protocol" [RFC7591] generalized the dynamic client registration mechanisms defined by "OpenID Connect Dynamic Client Registration 1.0" [OpenID.Registration] in a way that is compatible with it.

The metadata for an authorization server is retrieved from a well-known location as a JSON [RFC8259] document, which declares its endpoint locations and authorization server capabilities. This process is described in Section 3.

This metadata can be communicated either in a self-asserted fashion by the server origin via HTTPS or as a set of signed metadata values represented as claims in a JSON Web Token (JWT) [JWT]. In the JWT case, the issuer is vouching for the validity of the data about the authorization server. This is analogous to the role that the Software Statement plays in OAuth Dynamic Client Registration [RFC7591].

The means by which the client chooses an authorization server is out of scope. In some cases, its issuer identifier may be manually configured into the client. In other cases, it may be dynamically discovered, for instance, through the use of WebFinger [RFC7033], as described in Section 2 of "OpenID Connect Discovery 1.0" [OpenID.Discovery].

1.1. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

All uses of JSON Web Signature (JWS) [JWS] and JSON Web Encryption (JWE) [JWE] data structures in this specification utilize the JWS Compact Serialization or the JWE Compact Serialization; the JWS JSON Serialization and the JWE JSON Serialization are not used.

1.2. Terminology

This specification uses the terms "Access Token", "Authorization Code", "Authorization Endpoint", "Authorization Grant", "Authorization Server", "Client", "Client Authentication", "Client Identifier", "Client Secret", "Grant Type", "Protected Resource", "Redirection URI", "Refresh Token", "Resource Owner", "Resource Server", "Response Type", and "Token Endpoint" defined by OAuth 2.0 [RFC6749]; the terms "Claim Name", "Claim Value", and "JSON Web Token (JWT)" defined by JSON Web Token (JWT) [JWT]; and the term "Response Mode" defined by "OAuth 2.0 Multiple Response Type Encoding Practices" [OAuth.Responses].

2. Authorization Server Metadata

Authorization servers can have metadata describing their configuration. The following authorization server metadata values are used by this specification and are registered in the IANA "OAuth Authorization Server Metadata" registry established in Section 7.1:

issuer
: REQUIRED. The authorization server's issuer identifier, which is a URL that uses the "https" scheme and has no query or fragment components. Authorization server metadata is published at a location that is ".well-known" according to RFC 5785 [RFC5785] derived from this issuer identifier, as described in Section 3. The issuer identifier is used to prevent authorization server mix-up attacks, as described in "OAuth 2.0 Mix-Up Mitigation" [MIX-UP].

authorization_endpoint
: URL of the authorization server's authorization endpoint [RFC6749]. This is REQUIRED unless no grant types are supported that use the authorization endpoint.

token_endpoint
: URL of the authorization server's token endpoint [RFC6749]. This is REQUIRED unless only the implicit grant type is supported.

jwks_uri
: OPTIONAL. URL of the authorization server's JWK Set [JWK] document. The referenced document contains the signing key(s) the client uses to validate signatures from the authorization server. This URL MUST use the "https" scheme. The JWK Set MAY also contain the server's encryption key or keys, which are used by clients to encrypt requests to the server. When both signing and encryption keys are made available, a "use" (public key use) parameter value is REQUIRED for all keys in the referenced JWK Set to indicate each key's intended usage.

registration_endpoint
: OPTIONAL. URL of the authorization server's OAuth 2.0 Dynamic Client Registration endpoint [RFC7591].

scopes_supported
: RECOMMENDED. JSON array containing a list of the OAuth 2.0 [RFC6749] "scope" values that this authorization server supports. Servers MAY choose not to advertise some supported scope values even when this parameter is used.

response_types_supported
: REQUIRED. JSON array containing a list of the OAuth 2.0 "response_type" values that this authorization server supports. The array values used are the same as those used with the "response_types" parameter defined by "OAuth 2.0 Dynamic Client Registration Protocol" [RFC7591].

response_modes_supported
: OPTIONAL. JSON array containing a list of the OAuth 2.0 "response_mode" values that this authorization server supports, as specified in "OAuth 2.0 Multiple Response Type Encoding Practices" [OAuth.Responses]. If omitted, the default is "["query", "fragment"]". The response mode value "form_post" is also defined in "OAuth 2.0 Form Post Response Mode" [OAuth.Post].

grant_types_supported
: OPTIONAL. JSON array containing a list of the OAuth 2.0 grant type values that this authorization server supports. The array values used are the same as those used with the "grant_types" parameter defined by "OAuth 2.0 Dynamic Client Registration Protocol" [RFC7591]. If omitted, the default value is "["authorization_code", "implicit"]".

token_endpoint_auth_methods_supported
: OPTIONAL. JSON array containing a list of client authentication methods supported by this token endpoint. Client authentication method values are used in the "token_endpoint_auth_method" parameter defined in Section 2 of [RFC7591]. If omitted, the default is "client_secret_basic" -- the HTTP Basic Authentication Scheme specified in Section 2.3.1 of OAuth 2.0 [RFC6749].

token_endpoint_auth_signing_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWS signing algorithms ("alg" values) supported by the token endpoint for the signature on the JWT [JWT] used to authenticate the client at the token endpoint for the "private_key_jwt" and "client_secret_jwt" authentication methods. This metadata entry MUST be present if either of these authentication methods are specified in the "token_endpoint_auth_methods_supported" entry. No default algorithms are implied if this entry is omitted. Servers SHOULD support "RS256". The value "none" MUST NOT be used.

service_documentation
: OPTIONAL. URL of a page containing human-readable information that developers might want or need to know when using the authorization server. In particular, if the authorization server

does not support Dynamic Client Registration, then information on how to register clients needs to be provided in this documentation.

ui_locales_supported
OPTIONAL. Languages and scripts supported for the user interface, represented as a JSON array of language tag values from BCP 47 [RFC5646]. If omitted, the set of supported languages and scripts is unspecified.

op_policy_uri
OPTIONAL. URL that the authorization server provides to the person registering the client to read about the authorization server's requirements on how the client can use the data provided by the authorization server. The registration process SHOULD display this URL to the person registering the client if it is given. As described in Section 5, despite the identifier "op_policy_uri" appearing to be OpenID-specific, its usage in this specification is actually referring to a general OAuth 2.0 feature that is not specific to OpenID Connect.

op_tos_uri
OPTIONAL. URL that the authorization server provides to the person registering the client to read about the authorization server's terms of service. The registration process SHOULD display this URL to the person registering the client if it is given. As described in Section 5, despite the identifier "op_tos_uri", appearing to be OpenID-specific, its usage in this specification is actually referring to a general OAuth 2.0 feature that is not specific to OpenID Connect.

revocation_endpoint
OPTIONAL. URL of the authorization server's OAuth 2.0 revocation endpoint [RFC7009].

revocation_endpoint_auth_methods_supported
OPTIONAL. JSON array containing a list of client authentication methods supported by this revocation endpoint. The valid client authentication method values are those registered in the IANA "OAuth Token Endpoint Authentication Methods" registry [IANA.OAuth.Parameters]. If omitted, the default is "client_secret_basic" -- the HTTP Basic Authentication Scheme specified in Section 2.3.1 of OAuth 2.0 [RFC6749].

revocation_endpoint_auth_signing_alg_values_supported
OPTIONAL. JSON array containing a list of the JWS signing algorithms ("alg" values) supported by the revocation endpoint for the signature on the JWT [JWT] used to authenticate the client at

the revocation endpoint for the "private_key_jwt" and "client_secret_jwt" authentication methods. This metadata entry MUST be present if either of these authentication methods are specified in the "revocation_endpoint_auth_methods_supported" entry. No default algorithms are implied if this entry is omitted. The value "none" MUST NOT be used.

introspection_endpoint
: OPTIONAL. URL of the authorization server's OAuth 2.0 introspection endpoint [RFC7662].

introspection_endpoint_auth_methods_supported
: OPTIONAL. JSON array containing a list of client authentication methods supported by this introspection endpoint. The valid client authentication method values are those registered in the IANA "OAuth Token Endpoint Authentication Methods" registry [IANA.OAuth.Parameters] or those registered in the IANA "OAuth Access Token Types" registry [IANA.OAuth.Parameters]. (These values are and will remain distinct, due to Section 7.2.) If omitted, the set of supported authentication methods MUST be determined by other means.

introspection_endpoint_auth_signing_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWS signing algorithms ("alg" values) supported by the introspection endpoint for the signature on the JWT [JWT] used to authenticate the client at the introspection endpoint for the "private_key_jwt" and "client_secret_jwt" authentication methods. This metadata entry MUST be present if either of these authentication methods are specified in the "introspection_endpoint_auth_methods_supported" entry. No default algorithms are implied if this entry is omitted. The value "none" MUST NOT be used.

code_challenge_methods_supported
: OPTIONAL. JSON array containing a list of Proof Key for Code Exchange (PKCE) [RFC7636] code challenge methods supported by this authorization server. Code challenge method values are used in the "code_challenge_method" parameter defined in Section 4.3 of [RFC7636]. The valid code challenge method values are those registered in the IANA "PKCE Code Challenge Methods" registry [IANA.OAuth.Parameters]. If omitted, the authorization server does not support PKCE.

Additional authorization server metadata parameters MAY also be used. Some are defined by other specifications, such as OpenID Connect Discovery 1.0 [OpenID.Discovery].

2.1. Signed Authorization Server Metadata

In addition to JSON elements, metadata values MAY also be provided as a "signed_metadata" value, which is a JSON Web Token (JWT) [JWT] that asserts metadata values about the authorization server as a bundle. A set of claims that can be used in signed metadata is defined in Section 2. The signed metadata MUST be digitally signed or MACed using JSON Web Signature (JWS) [JWS] and MUST contain an "iss" (issuer) claim denoting the party attesting to the claims in the signed metadata. Consumers of the metadata MAY ignore the signed metadata if they do not support this feature. If the consumer of the metadata supports signed metadata, metadata values conveyed in the signed metadata MUST take precedence over the corresponding values conveyed using plain JSON elements.

Signed metadata is included in the authorization server metadata JSON object using this OPTIONAL member:

signed_metadata
: A JWT containing metadata values about the authorization server as claims. This is a string value consisting of the entire signed JWT. A "signed_metadata" metadata value SHOULD NOT appear as a claim in the JWT.

3. Obtaining Authorization Server Metadata

Authorization servers supporting metadata MUST make a JSON document containing metadata as specified in Section 2 available at a path formed by inserting a well-known URI string into the authorization server's issuer identifier between the host component and the path component, if any. By default, the well-known URI string used is "/.well-known/oauth-authorization-server". This path MUST use the "https" scheme. The syntax and semantics of ".well-known" are defined in RFC 5785 [RFC5785]. The well-known URI suffix used MUST be registered in the IANA "Well-Known URIs" registry [IANA.well-known].

Different applications utilizing OAuth authorization servers in application-specific ways may define and register different well-known URI suffixes used to publish authorization server metadata as used by those applications. For instance, if the example application uses an OAuth authorization server in an example-specific way, and there are example-specific metadata values that it needs to publish, then it might register and use the "example-configuration" URI suffix and publish the metadata document at the path formed by inserting "/.well-known/example-configuration" between the host and path components of the authorization server's issuer identifier. Alternatively, many such applications will use the default well-known

URI string "/.well-known/oauth-authorization-server", which is the right choice for general-purpose OAuth authorization servers, and not register an application-specific one.

An OAuth 2.0 application using this specification MUST specify what well-known URI suffix it will use for this purpose. The same authorization server MAY choose to publish its metadata at multiple well-known locations derived from its issuer identifier, for example, publishing metadata at both "/.well-known/example-configuration" and "/.well-known/oauth-authorization-server".

Some OAuth applications will choose to use the well-known URI suffix "openid-configuration". As described in Section 5, despite the identifier "/.well-known/openid-configuration", appearing to be OpenID specific, its usage in this specification is actually referring to a general OAuth 2.0 feature that is not specific to OpenID Connect.

3.1. Authorization Server Metadata Request

An authorization server metadata document MUST be queried using an HTTP "GET" request at the previously specified path.

The client would make the following request when the issuer identifier is "https://example.com" and the well-known URI suffix is "oauth-authorization-server" to obtain the metadata, since the issuer identifier contains no path component:

```
GET /.well-known/oauth-authorization-server HTTP/1.1
Host: example.com
```

If the issuer identifier value contains a path component, any terminating "/" MUST be removed before inserting "/.well-known/" and the well-known URI suffix between the host component and the path component. The client would make the following request when the issuer identifier is "https://example.com/issuer1" and the well-known URI suffix is "oauth-authorization-server" to obtain the metadata, since the issuer identifier contains a path component:

```
GET /.well-known/oauth-authorization-server/issuer1 HTTP/1.1
Host: example.com
```

Using path components enables supporting multiple issuers per host. This is required in some multi-tenant hosting configurations. This use of ".well-known" is for supporting multiple issuers per host; unlike its use in RFC 5785 [RFC5785], it does not provide general information about the host.

3.2. Authorization Server Metadata Response

The response is a set of claims about the authorization server's configuration, including all necessary endpoints and public key location information. A successful response MUST use the 200 OK HTTP status code and return a JSON object using the "application/json" content type that contains a set of claims as its members that are a subset of the metadata values defined in Section 2. Other claims MAY also be returned.

Claims that return multiple values are represented as JSON arrays. Claims with zero elements MUST be omitted from the response.

An error response uses the applicable HTTP status code value.

The following is a non-normative example response:

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

{
 "issuer":
   "https://server.example.com",
 "authorization_endpoint":
   "https://server.example.com/authorize",
 "token_endpoint":
   "https://server.example.com/token",
 "token_endpoint_auth_methods_supported":
   ["client_secret_basic", "private_key_jwt"],
 "token_endpoint_auth_signing_alg_values_supported":
   ["RS256", "ES256"],
 "userinfo_endpoint":
   "https://server.example.com/userinfo",
 "jwks_uri":
   "https://server.example.com/jwks.json",
 "registration_endpoint":
   "https://server.example.com/register",
 "scopes_supported":
   ["openid", "profile", "email", "address",
    "phone", "offline_access"],
 "response_types_supported":
   ["code", "code token"],
 "service_documentation":
   "http://server.example.com/service_documentation.html",
 "ui_locales_supported":
   ["en-US", "en-GB", "en-CA", "fr-FR", "fr-CA"]
}
```

3.3. Authorization Server Metadata Validation

The "issuer" value returned MUST be identical to the authorization server's issuer identifier value into which the well-known URI string was inserted to create the URL used to retrieve the metadata. If these values are not identical, the data contained in the response MUST NOT be used.

4. String Operations

Processing some OAuth 2.0 messages requires comparing values in the messages to known values. For example, the member names in the metadata response might be compared to specific member names such as "issuer". Comparing Unicode [UNICODE] strings, however, has significant security implications.

Therefore, comparisons between JSON strings and other Unicode strings MUST be performed as specified below:

1. Remove any JSON-applied escaping to produce an array of Unicode code points.

2. Unicode Normalization [USA15] MUST NOT be applied at any point to either the JSON string or the string it is to be compared against.

3. Comparisons between the two strings MUST be performed as a Unicode code-point-to-code-point equality comparison.

Note that this is the same equality comparison procedure described in Section 8.3 of [RFC8259].

5. Compatibility Notes

The identifiers "/.well-known/openid-configuration", "op_policy_uri", and "op_tos_uri" contain strings referring to the OpenID Connect [OpenID.Core] family of specifications that were originally defined by "OpenID Connect Discovery 1.0" [OpenID.Discovery]. Despite the reuse of these identifiers that appear to be OpenID specific, their usage in this specification is actually referring to general OAuth 2.0 features that are not specific to OpenID Connect.

The algorithm for transforming the issuer identifier to an authorization server metadata location defined in Section 3 is equivalent to the corresponding transformation defined in Section 4 of "OpenID Connect Discovery 1.0" [OpenID.Discovery], provided that the issuer identifier contains no path component. However, they are different when there is a path component, because OpenID Connect

Discovery 1.0 specifies that the well-known URI string is appended to the issuer identifier (e.g., "https://example.com/issuer1/.well-known/openid-configuration"), whereas this specification specifies that the well-known URI string is inserted before the path component of the issuer identifier (e.g., "https://example.com/.well-known/openid-configuration/issuer1").

Going forward, OAuth authorization server metadata locations should use the transformation defined in this specification. However, when deployed in legacy environments in which the OpenID Connect Discovery 1.0 transformation is already used, it may be necessary during a transition period to publish metadata for issuer identifiers containing a path component at both locations. During this transition period, applications should first apply the transformation defined in this specification and attempt to retrieve the authorization server metadata from the resulting location; only if the retrieval from that location fails should they fall back to attempting to retrieve it from the alternate location obtained using the transformation defined by OpenID Connect Discovery 1.0. This backwards-compatible behavior should only be necessary when the well-known URI suffix employed by the application is "openid-configuration".

6. Security Considerations

6.1. TLS Requirements

Implementations MUST support TLS. Which version(s) ought to be implemented will vary over time and depend on the widespread deployment and known security vulnerabilities at the time of implementation. The authorization server MUST support TLS version 1.2 [RFC5246] and MAY support additional TLS mechanisms meeting its security requirements. When using TLS, the client MUST perform a TLS/SSL server certificate check, per RFC 6125 [RFC6125]. Implementation security considerations can be found in "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)" [BCP195].

To protect against information disclosure and tampering, confidentiality protection MUST be applied using TLS with a ciphersuite that provides confidentiality and integrity protection.

6.2. Impersonation Attacks

TLS certificate checking MUST be performed by the client, as described in Section 6.1, when making an authorization server metadata request. Checking that the server certificate is valid for the issuer identifier URL prevents man-in-middle and DNS-based

attacks. These attacks could cause a client to be tricked into using an attacker's keys and endpoints, which would enable impersonation of the legitimate authorization server. If an attacker can accomplish this, they can access the resources that the affected client has access to using the authorization server that they are impersonating.

An attacker may also attempt to impersonate an authorization server by publishing a metadata document that contains an "issuer" claim using the issuer identifier URL of the authorization server being impersonated, but with its own endpoints and signing keys. This would enable it to impersonate that authorization server, if accepted by the client. To prevent this, the client MUST ensure that the issuer identifier URL it is using as the prefix for the metadata request exactly matches the value of the "issuer" metadata value in the authorization server metadata document received by the client.

6.3. Publishing Metadata in a Standard Format

Publishing information about the authorization server in a standard format makes it easier for both legitimate clients and attackers to use the authorization server. Whether an authorization server publishes its metadata in an ad hoc manner or in the standard format defined by this specification, the same defenses against attacks that might be mounted that use this information should be applied.

6.4. Protected Resources

Secure determination of appropriate protected resources to use with an authorization server for all use cases is out of scope of this specification. This specification assumes that the client has a means of determining appropriate protected resources to use with an authorization server and that the client is using the correct metadata for each authorization server. Implementers need to be aware that if an inappropriate protected resource is used by the client, that an attacker may be able to act as a man-in-the-middle proxy to a valid protected resource without it being detected by the authorization server or the client.

The ways to determine the appropriate protected resources to use with an authorization server are, in general, application dependent. For instance, some authorization servers are used with a fixed protected resource or set of protected resources, the locations of which may be well known or could be published as metadata values by the authorization server. In other cases, the set of resources that can be used with an authorization server can be dynamically changed by administrative actions. Many other means of determining appropriate associations between authorization servers and protected resources are also possible.

7. IANA Considerations

The following registration procedure is used for the registry established by this specification.

Values are registered on a Specification Required [RFC8126] basis after a two-week review period on the oauth-ext-review@ietf.org mailing list, on the advice of one or more Designated Experts. However, to allow for the allocation of values prior to publication, the Designated Experts may approve registration once they are satisfied that such a specification will be published.

Registration requests sent to the mailing list for review should use an appropriate subject (e.g., "Request to register OAuth Authorization Server Metadata: example").

Within the review period, the Designated Experts will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention (using the iesg@ietf.org mailing list) for resolution.

Criteria that should be applied by the Designated Experts include determining whether the proposed registration duplicates existing functionality, determining whether it is likely to be of general applicability or whether it is useful only for a single application, and whether the registration makes sense.

IANA must only accept registry updates from the Designated Experts and should direct all requests for registration to the review mailing list.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly-informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Designated Expert, that Designated Expert should defer to the judgment of the other Designated Experts.

7.1. OAuth Authorization Server Metadata Registry

This specification establishes the IANA "OAuth Authorization Server Metadata" registry for OAuth 2.0 authorization server metadata names. The registry records the authorization server metadata member and a reference to the specification that defines it.

The Designated Experts must either:

(a) require that metadata names and values being registered use only printable ASCII characters excluding double quote (’"’) and backslash (’\’) (the Unicode characters with code points U+0021, U+0023 through U+005B, and U+005D through U+007E), or

(b) if new metadata members or values are defined that use other code points, require that their definitions specify the exact sequences of Unicode code points used to represent them. Furthermore, proposed registrations that use Unicode code points that can only be represented in JSON strings as escaped characters must not be accepted.

7.1.1. Registration Template

Metadata Name:
: The name requested (e.g., "issuer"). This name is case-sensitive. Names may not match other registered names in a case-insensitive manner (one that would cause a match if the Unicode toLowerCase() operation were applied to both strings) unless the Designated Experts state that there is a compelling reason to allow an exception.

Metadata Description:
: Brief description of the metadata (e.g., "Issuer identifier URL").

Change Controller:
: For Standards Track RFCs, list the "IESG". For others, give the name of the responsible party. Other details (e.g., postal address, email address, home page URI) may also be included.

Specification Document(s):
: Reference to the document or documents that specify the parameter, preferably including URIs that can be used to retrieve copies of the documents. An indication of the relevant sections may also be included but is not required.

7.1.2. Initial Registry Contents

- Metadata Name: issuer
- Metadata Description: Authorization server's issuer identifier URL
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: authorization_endpoint
- Metadata Description: URL of the authorization server's authorization endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: token_endpoint
- Metadata Description: URL of the authorization server's token endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: jwks_uri
- Metadata Description: URL of the authorization server's JWK Set document
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: registration_endpoint
- Metadata Description: URL of the authorization server's OAuth 2.0 Dynamic Client Registration Endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: scopes_supported
- Metadata Description: JSON array containing a list of the OAuth 2.0 "scope" values that this authorization server supports
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: response_types_supported
- Metadata Description: JSON array containing a list of the OAuth 2.0 "response_type" values that this authorization server supports
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: response_modes_supported
- Metadata Description: JSON array containing a list of the OAuth 2.0 "response_mode" values that this authorization server supports
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: grant_types_supported
- Metadata Description: JSON array containing a list of the OAuth 2.0 grant type values that this authorization server supports
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: token_endpoint_auth_methods_supported
- Metadata Description: JSON array containing a list of client authentication methods supported by this token endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: token_endpoint_auth_signing_alg_values_supported
- Metadata Description: JSON array containing a list of the JWS signing algorithms supported by the token endpoint for the signature on the JWT used to authenticate the client at the token endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: service_documentation
- Metadata Description: URL of a page containing human-readable information that developers might want or need to know when using the authorization server
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: ui_locales_supported
- Metadata Description: Languages and scripts supported for the user interface, represented as a JSON array of language tag values from BCP 47
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: op_policy_uri
- Metadata Description: URL that the authorization server provides to the person registering the client to read about the authorization server’s requirements on how the client can use the data provided by the authorization server
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: op_tos_uri
- Metadata Description: URL that the authorization server provides to the person registering the client to read about the authorization server’s terms of service
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: revocation_endpoint
- Metadata Description: URL of the authorization server’s OAuth 2.0 revocation endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: revocation_endpoint_auth_methods_supported
- Metadata Description: JSON array containing a list of client authentication methods supported by this revocation endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: revocation_endpoint_auth_signing_alg_values_supported
- Metadata Description: JSON array containing a list of the JWS signing algorithms supported by the revocation endpoint for the signature on the JWT used to authenticate the client at the revocation endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: introspection_endpoint
- Metadata Description: URL of the authorization server’s OAuth 2.0 introspection endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: introspection_endpoint_auth_methods_supported
- Metadata Description: JSON array containing a list of client authentication methods supported by this introspection endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: introspection_endpoint_auth_signing_alg_values_supported
- Metadata Description: JSON array containing a list of the JWS signing algorithms supported by the introspection endpoint for the signature on the JWT used to authenticate the client at the introspection endpoint
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: code_challenge_methods_supported
- Metadata Description: PKCE code challenge methods supported by this authorization server
- Change Controller: IESG
- Specification Document(s): Section 2 of RFC 8414

- Metadata Name: signed_metadata
- Metadata Description: Signed JWT containing metadata values about the authorization server as claims
- Change Controller: IESG
- Specification Document(s): Section 2.1 of RFC 8414

7.2. Updated Registration Instructions

This specification adds to the instructions for the Designated Experts of the following IANA registries, both of which are in the "OAuth Parameters" registry [IANA.OAuth.Parameters]:

- OAuth Access Token Types
- OAuth Token Endpoint Authentication Methods

IANA has added a link to this specification in the Reference sections of these registries.

For these registries, the Designated Experts must reject registration requests in one registry for values already occurring in the other registry. This is necessary because the "introspection_endpoint_auth_methods_supported" parameter allows for the use of values from either registry. That way, because the values in the two registries will continue to be mutually exclusive, no ambiguities will arise.

7.3. Well-Known URI Registry

This specification registers the well-known URI defined in Section 3 in the IANA "Well-Known URIs" registry [IANA.well-known] established by RFC 5785 [RFC5785].

7.3.1. Registry Contents

- URI suffix: oauth-authorization-server
- Change controller: IESG
- Specification document: Section 3 of RFC 8414
- Related information: (none)

8. References

8.1. Normative References

[BCP195] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, May 2015, <http://www.rfc-editor.org/info/bcp195>.

[IANA.OAuth.Parameters] IANA, "OAuth Parameters", <https://www.iana.org/assignments/oauth-parameters>.

[JWE] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[JWK] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[JWT] 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>.

[OAuth.Post] Jones, M. and B. Campbell, "OAuth 2.0 Form Post Response Mode", April 2015, <http://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html>.

[OAuth.Responses] de Medeiros, B., Ed., Scurtescu, M., Tarjan, P., and M. Jones, "OAuth 2.0 Multiple Response Type Encoding Practices", February 2014, <http://openid.net/specs/oauth-v2-multiple-response-types-1_0.html>.

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[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>.

[RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <https://www.rfc-editor.org/info/rfc7009>.

[RFC7033] Jones, P., Salgueiro, G., Jones, M., and J. Smarr, "WebFinger", RFC 7033, DOI 10.17487/RFC7033, September 2013, <https://www.rfc-editor.org/info/rfc7033>.

[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/info/rfc7591>.

[RFC7636] Sakimura, N., Ed., Bradley, J., and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", RFC 7636, DOI 10.17487/RFC7636, September 2015, <https://www.rfc-editor.org/info/rfc7636>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[UNICODE] The Unicode Consortium, "The Unicode Standard", <http://www.unicode.org/versions/latest/>.

[USA15] Davis, M., Ed. and K. Whistler, Ed., "Unicode Normalization Forms", Unicode Standard Annex #15, May 2018, <http://www.unicode.org/reports/tr15/>.

8.2. Informative References

[IANA.well-known]
IANA, "Well-Known URIs", <https://www.iana.org/assignments/well-known-uris>.

[MIX-UP] Jones, M., Bradley, J., and N. Sakimura, "OAuth 2.0 Mix-Up Mitigation", Work in Progress, draft-ietf-oauth-mix-up-mitigation-01, July 2016.

[OpenID.Core]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[OpenID.Discovery]
Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0", November 2014, <http://openid.net/specs/openid-connect-discovery-1_0.html>.

[OpenID.Registration]
Sakimura, N., Bradley, J., and M. Jones, "OpenID Connect Dynamic Client Registration 1.0", November 2014, <http://openid.net/specs/openid-connect-registration-1_0.html>.

Acknowledgements

This specification is based on the OpenID Connect Discovery 1.0 specification, which was produced by the OpenID Connect working group of the OpenID Foundation. This specification standardizes the de facto usage of the metadata format defined by OpenID Connect Discovery to publish OAuth authorization server metadata.

The authors would like to thank the following people for their reviews of this specification: Shwetha Bhandari, Ben Campbell, Brian Campbell, Brian Carpenter, William Denniss, Vladimir Dzhuvinov, Donald Eastlake, Samuel Erdtman, George Fletcher, Dick Hardt, Phil Hunt, Alexey Melnikov, Tony Nadalin, Mark Nottingham, Eric Rescorla, Justin Richer, Adam Roach, Hannes Tschofenig, and Hans Zandbelt.

Authors' Addresses

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: http://self-issued.info/

Nat Sakimura
Nomura Research Institute, Ltd.

Email: n-sakimura@nri.co.jp
URI: http://nat.sakimura.org/

John Bradley
Yubico

Email: RFC8414@ve7jtb.com
URI: http://www.thread-safe.com/