OAuth Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 31, 2016

M. Jones
Microsoft
N. Sakimura
NRI
J. Bradley
Ping Identity
January 28, 2016

OAuth 2.0 Discovery
draft-jones-oauth-discovery-01

Abstract

This specification defines a mechanism for an OAuth 2.0 client to discover the resource owner's OAuth 2.0 authorization server and obtain information needed to interact with it, including its OAuth 2.0 endpoint locations and authorization server capabilities.

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 http://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 July 31, 2016.

Copyright Notice

Copyright (c) 2016 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 (http://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 Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Requirements Notation and Conventions 3
1.2. Terminology 3
2. Authorization Server WebFinger Discovery 4
3. Authorization Server Metadata 6
4. Obtaining Authorization Server Configuration Information . . . 10
4.1. Authorization Server Configuration Information Request . . 10
4.2. Authorization Server Configuration Information Response . 11
4.3. Authorization Server Configuration Information Validation 12
5. String Operations 12
6. Compatibility Notes 13
7. Security Considerations 13
7.1. TLS Requirements 13
7.2. Impersonation Attacks 14
8. IANA Considerations 14
8.1. OAuth Discovery Metadata Registry 15
8.1.1. Registration Template 15
8.1.2. Initial Registry Contents 16
8.2. Updated Registration Instructions 19
9. References 19
9.1. Normative References 19
9.2. Informative References 22
Appendix A. Acknowledgements 22
Appendix B. Document History 22
Authors' Addresses 23

1. Introduction

This specification generalizes the discovery mechanisms 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 the "OAuth 2.0 Dynamic Client Registration Protocol" [RFC7591] specification generalized the dynamic client registration mechanisms defined by "OpenID Connect Dynamic Client Registration 1.0" [OpenID.Registration] in a way that was compatible with it.

In order for an OAuth client to utilize OAuth 2.0 services for a resource owner, the client needs to know where the OAuth 2.0 authorization server is. This specification uses WebFinger [RFC7033] to locate the authorization server for an resource owner. This process is described in Section 2.

Once the authorization server has been identified, the configuration information for that authorization server is retrieved from a well-known location as a JSON [RFC7159] document, including its OAuth 2.0 endpoint locations and authorization server capabilities. This process is described in Section 4.

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 RFC 2119 [RFC2119].

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].

This specification also defines the following terms:

Resource
: Entity that is the target of a request in WebFinger.

Host
: Server where a WebFinger service is hosted.

2. Authorization Server WebFinger Discovery

Authorization server WebFinger discovery is a means of determining the location of the authorization server's configuration information.

WebFinger discovery is OPTIONAL; if a client knows the authorization server's configuration information location through an out-of-band mechanism, it can skip this step and proceed to Section 4.

WebFinger discovery requires the following information to make a discovery request:

resource
: Identifier for the target resource owner that is the subject of the discovery request.

host
: Server where the WebFinger service is hosted.

rel
: URI identifying the type of service whose location is being requested.

OAuth discovery uses the following "rel" value in WebFinger [RFC7033]:

Rel Type	URI
OAuth 2.0 Configuration Information Location URL	http://openid.net/specs/connect/1.0/issuer

To start discovery of OAuth 2.0 configuration information, the resource owner supplies a URI to the client that can be used to discover the corresponding authorization server. In some cases, the client may know this URI without involvement of the resource owner.

This URI might, for instance, be an e-mail address, an account identifier, a profile URL, or a service or tenant URL.

The host to which the WebFinger request will be made is obtained from the URI. The client then makes an HTTP "GET" request to the host's WebFinger [RFC7033] endpoint using the URI as the "resource" parameter value and the "rel" value "http://openid.net/specs/connect/1.0/issuer" to obtain the authorization server's configuration information location.

The configuration information location MUST be returned in the WebFinger response as the value of the "href" member of a "links" array element with "rel" member value "http://openid.net/specs/connect/1.0/issuer". As described in Section 6, despite the identifier "http://openid.net/specs/connect/1.0/issuer" 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. (Per Section 7 of WebFinger [RFC7033], obtaining the WebFinger response may first involve following some redirects.)

The returned configuration information location MUST be a URI RFC 3986 [RFC3986] with a scheme component that MUST be "https", a host component, and optionally, port and path components and no query or fragment components. Note that the WebFinger response can return a configuration information location value using a completely different scheme, host, port, and path from any contained in the input URI, and no relationship can be assumed between the input URI and the resulting configuration information location.

An example WebFinger discovery request follows. To find the authorization server's configuration information location for the account identified using the e-mail address syntax "joe@example.com" and corresponding account URI "acct:joe@example.com", the WebFinger parameters are as follows:

WebFinger Parameter	Value
resource	acct:joe@example.com
host	example.com
rel	http://openid.net/specs/connect/1.0/issuer

The client would make the following WebFinger request to discover the authorization server's configuration information location (with line wraps within lines for display purposes only):

```
  GET /.well-known/webfinger
    ?resource=acct%3Ajoe%40example.com
    &rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
    HTTP/1.1
  Host: example.com

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

  {
   "subject": "acct:joe@example.com",
   "links":
    [
     {
      "rel": "http://openid.net/specs/connect/1.0/issuer",
      "href": "https://server.example.com"
     }
    ]
  }
```

The discovered authorization server configuration information location is "https://server.example.com".

3. Authorization Server Metadata

Authorization servers can have metadata describing their configuration. These authorization server metadata values are used by this specification:

issuer
: REQUIRED. URL of the authorization server's configuration information location, which uses the "https" scheme and has no query or fragment components. This is the location where ".well-known" RFC 5785 [RFC5785] resources containing information about the authorization server are published, and in particular, the "/.well-known/openid-configuration" resource defined in Section 4. If WebFinger discovery is supported (see Section 2), this value MUST be identical to the configuration information location value returned by WebFinger.

authorization_endpoint
: REQUIRED. URL of the authorization server's authorization endpoint [RFC6749].

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

jwks_uri
: REQUIRED. URL of the authorization server's JWK Set [JWK] document. This contains the signing key(s) the client uses to validate signatures from the authorization server. The JWK Set MAY also contain the Server's encryption key(s), which are used by RPs 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. Although some algorithms allow the same key to be used for both signatures and encryption, doing so is NOT RECOMMENDED, as it is less secure. The JWK "x5c" parameter MAY be used to provide X.509 representations of keys provided. When used, the bare key values MUST still be present and MUST match those in the certificate.

registration_endpoint
: RECOMMENDED. 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.

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. 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. 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 BCP47 [RFC5646] language tag values.

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 6, 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 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 6, 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].

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. 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 8.2.)

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. The value "none" MUST

NOT be used.

code_challenge_methods_supported
OPTIONAL. JSON array containing a list of 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].

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

4. Obtaining Authorization Server Configuration Information

Using the configuration information location discovered as described in Section 2 or by other means, the authorization server's configuration information can be retrieved.

Authorization servers supporting discovery MUST make a JSON document available at the path formed by concatenating the string "/.well-known/openid-configuration" to the configuration information location. The syntax and semantics of ".well-known" are defined in RFC 5785 [RFC5785] and apply to the configuration information location value when it contains no path component. "openid-configuration" MUST point to a JSON document compliant with this specification and MUST be returned using the "application/json" content type. As described in Section 6, 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.

4.1. Authorization Server Configuration Information Request

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

The client would make the following request to the configuration information location "https://example.com" to obtain its configuration information, since the configuration information location contains no path component:

```
GET /.well-known/openid-configuration HTTP/1.1
Host: example.com
```

If the configuration information location value contains a path component, any terminating "/" MUST be removed before appending "/.well-known/openid-configuration". The client would make the following request to the configuration information location "https://example.com/issuer1" to obtain its configuration information, since the configuration information location contains a path component:

```
GET /issuer1/.well-known/openid-configuration 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.

4.2. Authorization Server Configuration Information 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 3. 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/connect/authorize",
 "token_endpoint":
   "https://server.example.com/connect/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/connect/userinfo",
 "jwks_uri":
   "https://server.example.com/jwks.json",
 "registration_endpoint":
   "https://server.example.com/connect/register",
 "scopes_supported":
   ["openid", "profile", "email", "address",
    "phone", "offline_access"],
 "response_types_supported":
   ["code", "code token"],
 "service_documentation":
   "http://server.example.com/connect/service_documentation.html",
 "ui_locales_supported":
   ["en-US", "en-GB", "en-CA", "fr-FR", "fr-CA"]
}
```

4.3. Authorization Server Configuration Information Validation

If any of the validation procedures defined in this specification fail, any operations requiring the information that failed to correctly validate MUST be aborted and the information that failed to validate MUST NOT be used.

The "issuer" value returned MUST be identical to the configuration information location URL that was directly used to retrieve the configuration information.

5. String Operations

Processing some OAuth 2.0 messages requires comparing values in the

messages to known values. For example, the member names in the configuration information 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 to 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.

6. Compatibility Notes

The identifiers "/.well-known/openid-configuration", "http://openid.net/specs/connect/1.0/issuer", "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.

7. Security Considerations

7.1. 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 transport-layer security 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 TLS and 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.

7.2. Impersonation Attacks

TLS certificate checking MUST be performed by the client, as described in Section 7.1, when making an authorization server configuration information request. Checking that the server certificate is valid for the configuration information location 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 discovery document that contains an "issuer" claim using the configuration information location 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, RPs MUST ensure that the configuration information location URL they are using for the configuration information request exactly matches the value of the "issuer" metadata value in the authorization server configuration information document received by the client.

8. IANA Considerations

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

Values are registered on a Specification Required [RFC5226] 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 Discovery 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 includes 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 Expert, that Expert should defer to the judgment of the other Experts.

8.1. OAuth Discovery Metadata Registry

This specification establishes the IANA "OAuth Discovery 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.

8.1.1. Registration Template

Discovery 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 unless the Designated Experts state that there is a compelling reason to allow an exception.

Discovery Metadata Description:
: Brief description of the discovery metadata (e.g., "Issuer 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.

8.1.2. Initial Registry Contents

- Discovery Metadata Name: "issuer"
- Discovery Metadata Description: URL of the authorization server's configuration information location
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "authorization_endpoint"
- Discovery Metadata Description: URL of the authorization server's authorization endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "token_endpoint"
- Discovery Metadata Description: URL of the authorization server's token endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "jwks_uri"
- Discovery Metadata Description: URL of the authorization server's JWK Set document
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "registration_endpoint"
- Discovery Metadata Description: URL of the authorization server's OAuth 2.0 Dynamic Client Registration Endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "scopes_supported"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "response_types_supported"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "response_modes_supported"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "grant_types_supported"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "token_endpoint_auth_methods_supported"
- Discovery Metadata Description: JSON array containing a list of client authentication methods supported by this token endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "token_endpoint_auth_signing_alg_values_supported"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "service_documentation"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "ui_locales_supported"
- Discovery Metadata Description: Languages and scripts supported for the user interface, represented as a JSON array of BCP47 language tag values
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "op_policy_uri"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "op_tos_uri"
- Discovery Metadata Description: URL that the authorization server provides to the person registering the client to read about authorization server's terms of service
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "revocation_endpoint"
- Discovery Metadata Description: URL of the authorization server's OAuth 2.0 revocation endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "revocation_endpoint_auth_methods_supported"
- Discovery Metadata Description: JSON array containing a list of client authentication methods supported by this revocation endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "revocation_endpoint_auth_signing_alg_values_supported"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "introspection_endpoint"
- Discovery Metadata Description: URL of the authorization server's OAuth 2.0 introspection endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "introspection_endpoint_auth_methods_supported"
- Discovery Metadata Description: JSON array containing a list of client authentication methods supported by this introspection endpoint
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

- Discovery Metadata Name: "introspection_endpoint_auth_signing_alg_values_supported"
- Discovery 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 3 of [[this specification]]

- Discovery Metadata Name: "code_challenge_methods_supported"
- Discovery Metadata Description: PKCE code challenge methods supported by this authorization server
- Change Controller: IESG
- Specification Document(s): Section 3 of [[this specification]]

8.2. 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. [[RFC Editor: The above sentence is written in the past tense as it would appear in the final specification, even though these links won't actually be created until after the IESG has requested publication of the specification. Please delete this note after the links are in place.]]

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.

9. References

9.1. 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, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/bcp195>.

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

[JWA] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518,
DOI 10.17487/RFC7518, May 2015,
<http://tools.ietf.org/html/rfc7518>.

[JWE] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)",
RFC 7516, DOI 10.17487/RFC7516, May 2015,
<http://tools.ietf.org/html/rfc7516>.

[JWK] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/
RFC7517, May 2015, <http://tools.ietf.org/html/rfc7517>.

[JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web
Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515,
May 2015, <http://tools.ietf.org/html/rfc7515>.

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token
(JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015,
<http://tools.ietf.org/html/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,
<http://www.rfc-editor.org/info/rfc2119>.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0",
RFC 2246, DOI 10.17487/RFC2246, January 1999,
<http://www.rfc-editor.org/info/rfc2246>.

[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,
<http://www.rfc-editor.org/info/rfc3986>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6125>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7033>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7565] Saint-Andre, P., "The 'acct' URI Scheme", RFC 7565, DOI 10.17487/RFC7565, May 2015, <http://www.rfc-editor.org/info/rfc7565>.

[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,
<http://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, <http://www.rfc-editor.org/info/rfc7636>.

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

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

[USA15] Davis, M. and K. Whistler, "Unicode Normalization Forms", Unicode Standard Annex 15, June 2015, <http://www.unicode.org/reports/tr15/>.

9.2. Informative References

[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>.

Appendix A. 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.

Appendix B. Document History

[[to be removed by the RFC Editor before publication as an RFC]]

-01

- Added "revocation_endpoint_auth_methods_supported" and "revocation_endpoint_auth_signing_alg_values_supported" for the revocation endpoint.

- Added "introspection_endpoint_auth_methods_supported" and "introspection_endpoint_auth_signing_alg_values_supported" for the introspection endpoint.

- Added "code_challenge_methods_supported" for PKCE.

-00

- Created the initial version based on OpenID Connect Discovery 1.0 draft 26.

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
Ping Identity

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