OAuth Working Group   M. Jones
Internet-Draft   Microsoft
Intended status: Standards Track   N. Sakimura
Expires: September 11, 2017   NRI
                                                        J. Bradley
                                                     Ping Identity
                                                    March 10, 2017

                OAuth 2.0 Authorization Server Metadata
                    draft-ietf-oauth-discovery-06

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 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 September 11, 2017.

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 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.

The metadata for an authorization server is retrieved from a well-
known location as a JSON [RFC7159] document, which declares its

endpoint locations and authorization server capabilities. This
process is described in Section 3.

This metadata can either be communicated in a self-asserted fashion
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 obtains the location of the
authorization server metadata document is out of scope. In some
cases, the location 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 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].

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. This is the location where ".well-known" RFC 5785
      [RFC5785] resources containing information about the authorization
      server are published. Using these well-known resources is
      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" [I-D.ietf-oauth-mix-up-mitigation].

   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
      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(s), 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. 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 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].

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 7.2.)

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].

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 are 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 concatenating a well-known URI string such as "/.well-known/oauth-authorization-server" to the authorization server's issuer identifier. The syntax and semantics of ".well-known" are defined in RFC 5785 [RFC5785]. The well-known URI path 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 path 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 path suffix and publish the metadata document at the path formed by concatenating "/.well-known/example-configuration" to the authorization server's issuer identifier.

An OAuth 2.0 application using this specification MUST specify what well-known URI string it will use for this purpose. The same authorization server MAY choose to publish its metadata at multiple well-known locations relative to 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 path suffix "openid-configuration" and publish the metadata document at the path formed by concatenating "/.well-known/openid-configuration" to the authorization server's issuer identifier. 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 path 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 appending "/.well-known/" and
the well-known URI path suffix. The client would make the following
request when the issuer identifier is "https://example.com/issuer1"
and the well-known URI path suffix is "oauth-authorization-server" to
obtain the metadata, since the issuer identifier contains a path
component:

```
GET /issuer1/.well-known/oauth-authorization-server 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 that was concatenated with the well-
known URI path suffix 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 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.

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.

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 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.

[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 which 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 by 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 [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
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 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.

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.

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 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

   o Metadata Name: "issuer"
   o Metadata Description: Authorization server's issuer identifier URL
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "authorization_endpoint"
   o Metadata Description: URL of the authorization server's
      authorization endpoint
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "token_endpoint"
   o Metadata Description: URL of the authorization server's token
      endpoint
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "jwks_uri"
   o Metadata Description: URL of the authorization server's JWK Set
      document
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "registration_endpoint"

   o Metadata Description: URL of the authorization server's OAuth 2.0
      Dynamic Client Registration Endpoint
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

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

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

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

   o Metadata Name: "grant_types_supported"
   o Metadata Description: JSON array containing a list of the OAuth
      2.0 grant type values that this authorization server supports
   o Change Controller: IESG
   o Specification Document(s): [Section 2](#) of [[this specification]]

   o Metadata Name: "token_endpoint_auth_methods_supported"
   o Metadata Description: JSON array containing a list of client
      authentication methods supported by this token endpoint
   o Change Controller: IESG
   o Specification Document(s): [Section 2](#) of [[this specification]]

   o Metadata Name: "token_endpoint_auth_signing_alg_values_supported"
   o 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
   o Change Controller: IESG
   o Specification Document(s): [Section 2](#) of [[this specification]]

   o Metadata Name: "service_documentation"
   o Metadata Description: URL of a page containing human-readable
      information that developers might want or need to know when using
      the authorization server
   o Change Controller: IESG

   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "ui_locales_supported"
   o Metadata Description: Languages and scripts supported for the user
      interface, represented as a JSON array of BCP47 language tag
      values
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "op_policy_uri"
   o 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
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "op_tos_uri"
   o Metadata Description: URL that the authorization server provides
      to the person registering the client to read about the
      authorization server's terms of service
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "revocation_endpoint"
   o Metadata Description: URL of the authorization server's OAuth 2.0
      revocation endpoint
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "revocation_endpoint_auth_methods_supported"
   o Metadata Description: JSON array containing a list of client
      authentication methods supported by this revocation endpoint
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name:
      "revocation_endpoint_auth_signing_alg_values_supported"
   o 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
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "introspection_endpoint"
   o Metadata Description: URL of the authorization server's OAuth 2.0

introspection endpoint

---

   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "introspection_endpoint_auth_methods_supported"
   o Metadata Description: JSON array containing a list of client
      authentication methods supported by this introspection endpoint
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name:
      "introspection_endpoint_auth_signing_alg_values_supported"
   o 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
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

   o Metadata Name: "code_challenge_methods_supported"
   o Metadata Description: PKCE code challenge methods supported by
      this authorization server
   o Change Controller: IESG
   o Specification Document(s): Section 2 of [[this specification]]

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]:

   o OAuth Access Token Types
   o 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.

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

   o URI suffix: "oauth-authorization-server"
   o Change controller: IESG
   o Specification document: Section 3 of [[this specification]]
   o 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, 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>.

   [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>.

   [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>.

   [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/>.

8.2. Informative References

[I-D.ietf-oauth-mix-up-mitigation]
            Jones, M., Bradley, J., and N. Sakimura, "OAuth 2.0 Mix-Up
            Mitigation", draft-ietf-oauth-mix-up-mitigation-01 (work
            in progress), July 2016.

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

[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.

The authors would like to thank the following people for their
reviews of this specification: Brian Campbell, William Denniss,
Vladimir Dzhuvinov, Samuel Erdtman, George Fletcher, Phil Hunt, Tony
Nadalin, Justin Richer, Hannes Tschofenig, and Hans Zandbelt.

Appendix B. Document History

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

   -06

   o Incorporated resolutions to working group last call comments.

   -05

   o Removed the "protected_resources" element and the reference to
      draft-jones-oauth-resource-metadata.

   -04

   o Added the ability to list protected resources with the
      "protected_resources" element.
   o Added ability to provide signed metadata with the
      "signed_metadata" element.
   o Removed "Discovery" from the name, since this is now just about
      authorization server metadata.

   -03

   o Changed term "issuer URL" to "issuer identifier" for terminology
      consistency, paralleling the same terminology consistency change
      in the mix-up mitigation spec.

   -02

   o Changed the title to OAuth 2.0 Authorization Server Discovery
      Metadata.
   o Made "jwks_uri" and "registration_endpoint" OPTIONAL.
   o Defined the well-known URI string "/.well-known/oauth-
      authorization-server".

o Added security considerations about publishing authorization
      server discovery metadata in a standard format.
   o Added security considerations about protected resources.
   o Added more information to the "grant_types_supported" and
      "response_types_supported" definitions.
   o Referenced the working group Mix-Up Mitigation draft.
   o Changed some example metadata values.
   o Acknowledged individuals for their contributions to the
      specification.

   -01

   o Removed WebFinger discovery.
   o Clarified the relationship between the issuer identifier URL and
      the well-known URI path relative to it at which the discovery
      metadata document is located.

   -00

   o Created the initial working group version based on draft-jones-
      oauth-discovery-01, with no normative changes.

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/