OAuth Working Group M. Jones
Internet-Draft Microsoft
Intended status: Standards Track P. Hunt
Expires: February 4, 2017 Oracle
August 3, 2016

OAuth 2.0 Protected Resource Metadata
draft-jones-oauth-resource-metadata-00

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 protected resource.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This specification defines a metadata format enabling OAuth 2.0 clients to obtain information needed to interact with an OAuth 2.0 protected resource. This specification is intentionally as parallel as possible to "OAuth 2.0 Dynamic Client Registration Protocol" [RFC7591], which enables a client to provide metadata about itself to an OAuth 2.0 authorization server and to OAuth 2.0 Authorization Server Metadata [OAuth.AuthorizationMetadata], which enables a client to obtain metadata about an OAuth 2.0 authorization server.

The metadata for a protected resource is retrieved from a well-known location as a JSON [RFC7159] document, which declares information about its capabilities and optionally, its relationships to other services. 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 protected resource. This is analogous to the role that the Software Statement plays in OAuth Dynamic Client Registration [RFC7591].

Each protected resource publishing metadata about itself makes its own metadata document available at a well-known location rooted at the protect resource's URL, even when the resource server implements multiple protected resources. This prevents attackers from publishing metadata supposedly describing the protected resource, but that is not actually authoritative for the protected resource, as described in Section 5.2.

The means by which the client obtains the location of the protected resource 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], in a manner related to the description 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].

2. Protected Resource Metadata

Protected resources can have metadata describing their configuration. The following protected resource metadata values are used by this

specification and are registered in the IANA "OAuth Protected Resource Metadata" registry established in Section 6.1:

resource
REQUIRED. The protected resource's resource identifier, which is a URL that uses the "https" scheme and has no fragment components. This is the location where ".well-known" RFC 5785 [RFC5785] resources containing information about the protected resource are published. Using these well-known resources is described in Section 3.

authorization_servers
OPTIONAL. JSON array containing a list of OAuth authorization server issuer identifiers, as defined in [OAuth.AuthorizationMetadata], for authorization servers that can be used with this protected resource. Protected resources MAY choose not to advertise some supported authorization servers even when this parameter is used. In some use cases, the set of authorization servers will not be enumerable, in which case this metadata parameter would not be used.

jwks_uri
OPTIONAL. URL of the protected resource's JWK Set [JWK] document. This contains keys belonging to the protected resource. For instance, this JWK Set MAY contain encryption key(s) that are used to encrypt access tokens to the protected resource. 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.

scopes_provided
RECOMMENDED. JSON array containing a list of the OAuth 2.0 [RFC6749] "scope" values that are used in authorization requests to request access to this protected resource. Protected resources MAY choose not to advertise some scope values provided even when this parameter is used.

bearer_methods_supported
OPTIONAL. JSON array containing a list of the OAuth 2.0 Bearer Token [RFC6750] presentation methods that this protected resource supports. Defined values are "["header", "fragment", "query"]", corresponding to Sections 2.1, 2.2, and 2.3 of RFC 6750.

resource_signing_alg_values_supported
OPTIONAL. JSON array containing a list of the JWS [JWS] signing algorithms ("alg" values) [JWA] supported by the protected resource for signed content. The value "none" MAY be included.

resource_encryption_alg_values_supported
   OPTIONAL. JSON array containing a list of the JWE [JWE] encryption algorithms ("alg" values) [JWA] supported by the protected resource for encrypted content.

resource_encryption_enc_values_supported
   OPTIONAL. JSON array containing a list of the JWE encryption algorithms ("enc" values) [JWA] supported by the protected resource for encrypted content.

resource_documentation
   OPTIONAL. URL of a page containing human-readable information that developers might want or need to know when using the protected resource

resource_policy_uri
   OPTIONAL. URL that the protected resource provides to read about the protected resource's requirements on how the client can use the data provided by the protected resource

resource_tos_uri
   OPTIONAL. URL that the protected resource provides to read about the protected resource's terms of service

Additional protected resource metadata parameters MAY also be used.

2.1. Signed Protected Resource 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 protected resource 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 those conveyed using plain JSON elements.

Signed metadata is included in the protected resource metadata JSON object using this OPTIONAL member:

signed_metadata
   A JWT containing metadata values about the protected resource 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 Protected Resource Metadata

Protected resources 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-protected-resource" to the protected resource's resource 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 protected resources in application-specific ways may define and register different well-known URI path suffixes used to publish protected resource metadata as used by those applications. For instance, if the Example application uses an OAuth protected resource 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-resource-configuration" URI path suffix and publish the metadata document at the path formed by concatenating "/.well-known/example-resource-configuration" to the protected resource's resource identifier.

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

3.1. Protected Resource Metadata Request

A protected resource metadata document MUST be queried using an HTTP "GET" request at the previously specified path.

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

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

If the resource identifier value contains a path component, any terminating "/" MUST be removed before appending "/.well-known/" and the well-known URI path suffix. The consumer of the metadata would

make the following request when the resource identifier is "https://resource.example.com/resource1" and the well-known URI path suffix is "oauth-protected-resource" to obtain the metadata, since the resource identifier contains a path component:

```
  GET /resource1/.well-known/oauth-protected-resource HTTP/1.1
  Host: resource.example.com
```

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

3.2. Protected Resource Metadata Response

The response is a set of claims about the protected resource's configuration. 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

  {
   "resource":
     "https://resource.example.com",
   "authorization_servers":
     ["https://as1.example.com",
      "https://as2.example.net"],
   "bearer_methods_supported":
     ["header", "body"],
   "resource_documentation":
     "http://resource.example.com/resource_documentation.html"
  }
```

3.3. Protected Resource Metadata Validation

The "resource" value returned MUST be identical to the protected resource's resource 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 "resource". 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. Security Considerations

5.1. 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 protected resource 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.

5.2. Impersonation Attacks

TLS certificate checking MUST be performed by the client, as described in Section 5.1, when making a protected resource metadata request. Checking that the server certificate is valid for the resource identifier URL prevents man-in-middle and DNS-based attacks. These attacks could cause a client to be tricked into using an attacker's resource server, which would enable impersonation of the legitimate protected resource. If an attacker can accomplish this, they can access the resources that the affected client has access to using the protected resource that they are impersonating.

An attacker may also attempt to impersonate a protected resource by publishing a metadata document that contains a "resource" claim using the resource identifier URL of the protected resource being impersonated, but containing information of the attacker's choosing. This would enable it to impersonate that protected resource, if accepted by the client. To prevent this, the client MUST ensure that the resource identifier URL it is using as the prefix for the metadata request exactly matches the value of the "resource" metadata value in the protected resource metadata document received by the client.

5.3. Publishing Metadata in a Standard Format

Publishing information about the protected resource in a standard format makes it easier for both legitimate clients and attackers to use the protected resource. Whether a protected resource 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.

5.4. Authorization Servers

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

The ways to determine the appropriate authorization servers to use with a protected resource are in general, application-dependent. For instance, some protected resources are used with a fixed

authorization server or set of authorization servers, the locations of which may be well known, or which could be published as metadata values by the protected resource. In other cases, the set of authorization servers that can be used with a protected resource can by dynamically changed by administrative actions or by changes to the set of authorization servers adhering to a trust framework. Many other means of determining appropriate associations between protected resources and authorization servers are also possible.

To support use cases in which the set of legitimate authorization servers to use with the protected resource is fixed and enumerable, this specification defines the "authorization_servers" metadata value, which enables explicitly listing them. Note that if the set of legitimate protected resources to use with an authorization server is also fixed and enumerable, lists in the protected resource metadata and authorization server metadata should be cross-checked against one another for consistency when these lists are used by the application profile.

6. 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 Protected Resource 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.

6.1. OAuth Protected Resource Metadata Registry

This specification establishes the IANA "OAuth Protected Resource Metadata" registry for OAuth 2.0 protected resource metadata names. The registry records the protected resource metadata member and a reference to the specification that defines it.

6.1.1. Registration Template

Metadata Name:
The name requested (e.g., "resource"). 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., "Resource 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.

6.1.2. Initial Registry Contents

- Metadata Name: "resource"
- Metadata Description: Protected resource's resource identifier URL
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "authorization_servers"
- Metadata Description: JSON array containing a list of OAuth authorization server issuer identifiers
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

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

- Metadata Name: "scopes_provided"
- Metadata Description: JSON array containing a list of the OAuth 2.0 "scope" values that are used in authorization requests to request access this protected resource
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "bearer_methods_supported"
- Metadata Description: JSON array containing a list of the OAuth 2.0 Bearer Token presentation methods that this protected resource supports
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "resource_signing_alg_values_supported"
- Metadata Description: JSON array containing a list of the JWS signing algorithms ("alg" values) supported by the protected resource for signed content
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "resource_encryption_alg_values_supported"
- Metadata Description: JSON array containing a list of the JWE encryption algorithms ("alg" values) supported by the protected resource for encrypted content
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "resource_encryption_enc_values_supported"
- Metadata Description: JSON array containing a list of the JWE encryption algorithms ("enc" values) supported by the protected resource for encrypted content
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "resource_documentation"

- Metadata Description: URL of a page containing human-readable information that developers might want or need to know when using the protected resource
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "resource_policy_uri"
- Metadata Description: URL that the protected resource provides to read about the protected resource's requirements on how the client can use the data provided by the protected resource
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

- Metadata Name: "resource_tos_uri"
- Metadata Description: URL that the protected resource provides to read about the protected resource's terms of service
- Change Controller: IESG
- Specification Document(s): Section 2 of [[this specification]]

6.2. 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].

6.2.1. Registry Contents

- URI suffix: "oauth-protected-resource"
- Change controller: IESG
- Specification document: Section 3 of [[this specification]]
- Related information: (none)

7. References

7.1. 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>.

[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.AuthorizationMetadata]
Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", draft-ietf-oauth-discovery-04 (work in progress), August 2016, <http://tools.ietf.org/html/draft-ietf-oauth-discovery-04>.

[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>.

[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>.

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

[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>.

[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/>.

7.2. 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.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>.

Appendix A. Acknowledgements

Thanks to George Fletcher and Tony Nadalin for their input on the specification.

Appendix B. Document History

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

-00

- Created the initial version. This draft reuses some text from draft-ietf-oauth-discovery-03.

Authors' Addresses

Michael B. Jones
Microsoft

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

Phil Hunt
Oracle

Email: phil.hunt@yahoo.com