

OAuth Working Group	M. Jones
Internet-Draft	Microsoft
Intended status: Standards Track	B. Campbell
Expires: June 30, 2013	Ping Identity
	C. Mortimore
	Salesforce
	December 27, 2012

JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0

draft-ietf-oauth-jwt-bearer-04

Abstract

This specification defines the use of a JSON Web Token (JWT) Bearer Token as a means for requesting an OAuth 2.0 access token as well as for use as a means of client authentication.

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 June 30, 2013.

Copyright Notice

Copyright (c) 2012 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**
 - 1.1. Notational Conventions**
 - 1.2. Terminology**
- 2. HTTP Parameter Bindings for Transporting Assertions**
 - 2.1. Using JWTs as Authorization Grants**
 - 2.2. Using JWTs for Client Authentication**
- 3. JWT Format and Processing Requirements**
 - 3.1. Authorization Grant Processing**
 - 3.2. Client Authentication Processing**
- 4. Authorization Grant Example**
- 5. Security Considerations**

6. IANA Considerations

6.1. Sub-Namespace Registration of urn:ietf:params:oauth:grant-type:jwt-bearer

6.2. Sub-Namespace Registration of urn:ietf:params:oauth:client-assertion-type:jwt-bearer

7. References

7.1. Normative References

7.2. Informative References

Appendix A. Acknowledgements

Appendix B. Document History

§ Authors' Addresses

1. Introduction

TOC

JSON Web Token (JWT) [JWT] is a JavaScript Object Notation (JSON) **[RFC4627]** based security token encoding that enables identity and security information to be shared across security domains. A security token is generally issued by an identity provider and consumed by a relying party that relies on its content to identify the token's subject for security related purposes.

The OAuth 2.0 Authorization Framework [RFC6749] provides a method for making authenticated HTTP requests to a resource using an access token. Access tokens are issued to third-party clients by an authorization server (AS) with the (sometimes implicit) approval of the resource owner. In OAuth, an authorization grant is an abstract term used to describe intermediate credentials that represent the resource owner authorization. An authorization grant is used by the client to obtain an access token. Several authorization grant types are defined to support a wide range of client types and user experiences. OAuth also allows for the definition of new extension grant types to support additional clients or to provide a bridge between OAuth and other trust frameworks. Finally, OAuth allows the definition of additional authentication mechanisms to be used by clients when interacting with the authorization server.

The **Assertion Framework for OAuth 2.0** [I-D.ietf-oauth-assertions] is an abstract extension to OAuth 2.0 that provides a general framework for the use of Assertions (a.k.a. Security Tokens) as client credentials and/or authorization grants with OAuth 2.0. This specification profiles the **Assertion Framework for OAuth 2.0** [I-D.ietf-oauth-assertions] to define an extension grant type that uses a JSON Web Token (JWT) Bearer Token to request an OAuth 2.0 access token as well as for use as client credentials. The format and processing rules for the JWT defined in this specification are intentionally similar, though not identical, to those in the closely related **SAML 2.0 Bearer Assertion Profiles for OAuth 2.0** [I-D.ietf-oauth-saml2-bearer].

This document defines how a JSON Web Token (JWT) Bearer Token can be used to request an access token when a client wishes to utilize an existing trust relationship, expressed through the semantics of (and digital signature calculated over) the JWT, without a direct user approval step at the authorization server. It also defines how a JWT can be used as a client authentication mechanism. The use of a security token for client authentication is orthogonal to and separable from using a security token as an authorization grant. They can be used either in combination or separately. Client authentication using a JWT is nothing more than an alternative way for a client to authenticate to the token endpoint and must be used in conjunction with some grant type to form a complete and meaningful protocol request. JWT authorization grants may be used with or without client authentication or identification. Whether or not client authentication is needed in conjunction with a JWT authorization grant, as well as the supported types of client authentication, are policy decisions at the discretion of the authorization server.

The process by which the client obtains the JWT, prior to exchanging it with the authorization server or using it for client authentication, is out of scope.

1.1. Notational Conventions

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",

"SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **RFC 2119** [RFC2119].

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

1.2. Terminology TOC

All terms are as defined in **The OAuth 2.0 Authorization Framework** [RFC6749], **Assertion Framework for OAuth 2.0** [I-D.ietf-oauth-assertions], and **JSON Web Token (JWT)** [JWT].

2. HTTP Parameter Bindings for Transporting Assertions TOC

The **Assertion Framework for OAuth 2.0** [I-D.ietf-oauth-assertions] defines generic HTTP parameters for transporting Assertions (a.k.a. Security Tokens) during interactions with a token endpoint. This section defines the values of those parameters for use with JWT Bearer Tokens.

2.1. Using JWTs as Authorization Grants TOC

To use a JWT Bearer Token as an authorization grant, use the following parameter values and encodings.

The value of the `grant_type` parameter MUST be `urn:ietf:params:oauth:grant-type:jwt-bearer`.

The value of the `assertion` parameter MUST contain a single JWT.

The following non-normative example demonstrates an Access Token Request with a JWT as an authorization grant (with extra line breaks for display purposes only):

```
POST /token.oauth2 HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
&assertion=eyJhbGciOiJIUzI1NiJ9.
eyJpc3Mi[...omitted for brevity...].
J9l-ZhWP[...omitted for brevity...]
```

2.2. Using JWTs for Client Authentication TOC

To use a JWT Bearer Token for client authentication, use the following parameter values and encodings.

The value of the `client_assertion_type` parameter MUST be `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.

The value of the `client_assertion` parameter MUST contain a single JWT.

The following non-normative example demonstrates client authentication using a JWT during the presentation of an authorization code grant in an Access Token Request (with extra line breaks for display purposes only):

```
POST /token.oauth2 HTTP/1.1
```

```
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=vAZEIHjQTHuGgaSvyW9h00RpusLzkvT0ww3trZBxZpo&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJIUzI1NiJ9.
eyJpc3Mi[...omitted for brevity...].
cC4hiUPo[...omitted for brevity...]
```

3. JWT Format and Processing Requirements

TOC

In order to issue an access token response as described in **The OAuth 2.0 Authorization Framework** [RFC6749] or to rely on a JWT for client authentication, the authorization server **MUST** validate the JWT according to the criteria below. Application of additional restrictions and policy are at the discretion of the authorization server.

- The JWT **MUST** contain an **iss** (issuer) claim that contains a unique identifier for the entity that issued the JWT.
- The JWT **MUST** contain a **sub** (subject) claim identifying the subject of the transaction. The subject **MAY** identify the resource owner for whom the access token is being requested. For client authentication, the subject **MUST** be the **client_id** of the OAuth client. When using a JWT as an authorization grant, the subject **SHOULD** identify an authorized accessor for whom the access token is being requested (typically the resource owner, or an authorized delegate).
- The JWT **MUST** contain an **aud** (audience) claim containing an audience value that identifies the authorization server, or the service provider principal entity of its controlling domain, as an intended audience. The token endpoint URL of the authorization server **MAY** be used as an acceptable value for an **aud** element. The authorization server **MUST** verify that it is an intended audience for the JWT.
- The JWT **MUST** contain an **exp** (expiration) claim that limits the time window during which the JWT can be used. The authorization server **MUST** verify that the expiration time has not passed, subject to allowable clock skew between systems. The authorization server **MAY** reject JWTs with an **exp** claim value that is unreasonably far in the future.
- The JWT **MAY** contain an **nbf** (not before) claim that identifies the time before which the token **MUST NOT** be accepted for processing.
- The JWT **MAY** contain an **iat** (issued at) claim that identifies the time at which the JWT was issued. The authorization server **MAY** reject JWTs with an **iat** claim value that is unreasonably far in the past.
- The JWT **MAY** contain a **jti** (JWT ID) claim that provides a unique identifier for the token. The authorization server **MAY** ensure that JWTs are not replayed by maintaining the set of used **jti** values for the length of time for which the JWT would be considered valid based on the applicable **exp** instant.
- The JWT **MAY** contain other claims.
- The JWT **MUST** be digitally signed by the issuer and the authorization server **MUST** verify the signature.
- The authorization server **MUST** verify that the JWT is valid in all other respects per **JSON Web Token (JWT)** [JWT].

3.1. Authorization Grant Processing

TOC

If present, the authorization server **MUST** also validate the client credentials.

If the JWT is not valid, or the current time is not within the token's valid time window for use, the authorization server **MUST** construct an error response as defined in **OAuth 2.0** [RFC6749]. The value of the **error** parameter **MUST** be the **invalid_grant** error code. The authorization server **MAY** include additional information regarding the reasons the JWT was considered invalid using the **error_description** or **error_uri** parameters.

For example:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store

{
  "error": "invalid_grant",
  "error_description": "Audience validation failed"
}
```

3.2. Client Authentication Processing

TOC

If the client JWT is not valid, or its subject confirmation requirements cannot be met, the authorization server **MUST** construct an error response as defined in **OAuth 2.0** [RFC6749]. The value of the `error` parameter **MUST** be the `invalid_client` error code. The authorization server **MAY** include additional information regarding the reasons the JWT was considered invalid using the `error_description` or `error_uri` parameters.

4. Authorization Grant Example

TOC

Though non-normative, the following examples illustrate what a conforming JWT and access token request would look like.

Below is an example JSON object that could be encoded to produce the JWT Claims Object for a JWT:

```
{ "iss": "https://jwt-idp.example.com",
  "sub": "mailto:mike@example.com",
  "aud": "https://jwt-rp.example.net",
  "nbf": 1300815780,
  "exp": 1300819380,
  "http://claims.example.com/member": true }
```

The following example JSON object, used as the header of a JWT, declares that the JWT is signed with the ECDSA P-256 SHA-256 algorithm.

```
{ "alg": "ES256" }
```

To present the JWT with the claims and header shown in the previous example as part of an access token request, for example, the client might make the following HTTPS request (with extra line breaks for display purposes only):

```
POST /token.oauth2 HTTP/1.1
Host: authz.example.net
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer
&assertion=eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJmcm9udGVudC5kaW9kIiwiaWF0IjoxMjM0NTY3ODk5LCJ1aWkiOiJmcm9udGVudC5kaW9kIn0.eyJpc3MiOiJmcm9udGVudC5kaW9kIiwiaWF0IjoxMjM0NTY3ODk5LCJ1aWkiOiJmcm9udGVudC5kaW9kIn0.eyJpc3MiOiJmcm9udGVudC5kaW9kIiwiaWF0IjoxMjM0NTY3ODk5LCJ1aWkiOiJmcm9udGVudC5kaW9kIn0.
```

5. Security Considerations

TOC

No additional security considerations apply beyond those described within **The OAuth 2.0 Authorization Framework** [RFC6749], the **Assertion Framework for OAuth 2.0** [I-D.ietf-oauth-assertions], and the **JSON Web Token (JWT)** [JWT] specification.

6. IANA Considerations TOC

6.1. Sub-Namespace Registration of urn:ietf:params:oauth:grant-type:jwt-bearer TOC

This specification registers the value `grant-type:jwt-bearer` in the IANA urn:ietf:params:oauth registry established in **An IETF URN Sub-Namespace for OAuth** [RFC6755].

- URN: urn:ietf:params:oauth:grant-type:jwt-bearer
- Common Name: JWT Bearer Token Grant Type Profile for OAuth 2.0
- Change controller: IETF
- Specification Document: [[this document]]

6.2. Sub-Namespace Registration of urn:ietf:params:oauth:client-assertion-type:jwt-bearer TOC

This specification registers the value `client-assertion-type:jwt-bearer` in the IANA urn:ietf:params:oauth registry established in **An IETF URN Sub-Namespace for OAuth** [RFC6755].

- URN: urn:ietf:params:oauth:client-assertion-type:jwt-bearer
- Common Name: JWT Bearer Token Profile for OAuth 2.0 Client Authentication
- Change controller: IETF
- Specification Document: [[this document]]

7. References TOC

7.1. Normative References TOC

- [I-D.ietf-oauth-assertions]** Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "**Assertion Framework for OAuth 2.0**," draft-ietf-oauth-assertions-08 (work in progress), November 2012 ([TXT](#), [PDF](#)).
- [JWT]** **Jones, M., Bradley, J., and N. Sakimura**, "**JSON Web Token (JWT)**," draft-ietf-oauth-json-web-token (work in progress), December 2012 ([HTML](#)).
- [RFC2119]** **Bradner, S.**, "**Key words for use in RFCs to Indicate Requirement Levels**," BCP 14, RFC 2119, March 1997 ([TXT](#), [HTML](#), [XML](#)).
- [RFC4627]** Crockford, D., "**The application/json Media Type for JavaScript Object Notation (JSON)**," RFC 4627, July 2006 ([TXT](#)).
- [RFC6749]** Hardt, D., "**The OAuth 2.0 Authorization Framework**," RFC 6749, October 2012 ([TXT](#)).
- [RFC6755]** Campbell, B. and H. Tschofenig, "**An IETF URN Sub-Namespace for OAuth**," RFC 6755, October 2012 ([TXT](#)).

7.2. Informative References TOC

- [I-D.ietf-oauth-saml2-bearer]** Campbell, B. and C. Mortimore, "**SAML 2.0 Bearer Assertion Profiles for OAuth 2.0**," draft-ietf-oauth-saml2-bearer-15 (work in progress), November 2012 ([TXT](#), [PDF](#)).

Appendix A. Acknowledgements

TOC

This profile was derived from **SAML 2.0 Bearer Assertion Profiles for OAuth 2.0** [I-D.ietf-oauth-saml2-bearer] by Brian Campbell and Chuck Mortimore.

Appendix B. Document History

TOC

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

-04

- Changed the name of the `prn` claim to `sub` (subject) both to more closely align with SAML name usage and to use a more intuitive name.
- Added seriesInfo information to Internet Draft references.

-03

- Reference RFC 6749 and RFC 6755.

-02

- Add more text to intro explaining that an assertion/JWT grant type can be used with or without client authentication/identification and that client assertion/JWT authentication is nothing more than an alternative way for a client to authenticate to the token endpoint
- Add examples to Sections 2.1 and 2.2
- Update references

-01

- Tracked specification name changes: "The OAuth 2.0 Authorization Protocol" to "The OAuth 2.0 Authorization Framework" and "OAuth 2.0 Assertion Profile" to "Assertion Framework for OAuth 2.0".
- Merged in changes between draft-ietf-oauth-saml2-bearer-11 and draft-ietf-oauth-saml2-bearer-13. All changes were strictly editorial.

-00

- Created the initial IETF draft based upon draft-jones-oauth-jwt-bearer-04 with no normative changes.

Authors' Addresses

TOC

Michael B. Jones
Microsoft

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

Brian Campbell
Ping Identity Corp.

Email: brian.d.campbell@gmail.com

Chuck Mortimore
Salesforce

Email: cmortimore@salesforce.com