

OAuth Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 15, 2014

M. Jones  
Microsoft  
B. Campbell  
Ping Identity  
C. Mortimore  
Salesforce  
July 14, 2013

**JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and  
Authorization Grants  
draft-ietf-oauth-jwt-bearer-06**

**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 January 15, 2014.

**Copyright Notice**

Copyright (c) 2013 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

|                                    |   |                    |
|------------------------------------|---|--------------------|
| <a href="#">1.</a>                 | <a href="#">Introduction . . . . .</a>  | <a href="#">3</a>  |
| <a href="#">1.1.</a>               | <a href="#">Notational Conventions . . . . .</a>  | <a href="#">4</a>  |
| <a href="#">1.2.</a>               | <a href="#">Terminology . . . . .</a>   | <a href="#">4</a>  |
| <a href="#">2.</a>                 | <a href="#">HTTP Parameter Bindings for Transporting Assertions . . . . .</a>                     | <a href="#">4</a>  |
| <a href="#">2.1.</a>               | <a href="#">Using JWTs as Authorization Grants . . . . .</a>                                      | <a href="#">4</a>  |
| <a href="#">2.2.</a>               | <a href="#">Using JWTs for Client Authentication . . . . .</a>                                    | <a href="#">5</a>  |
| <a href="#">3.</a>                 | <a href="#">JWT Format and Processing Requirements . . . . .</a>                                  | <a href="#">6</a>  |
| <a href="#">3.1.</a>               | <a href="#">Authorization Grant Processing . . . . .</a>  | <a href="#">7</a>  |
| <a href="#">3.2.</a>               | <a href="#">Client Authentication Processing . . . . .</a>  | <a href="#">8</a>  |
| <a href="#">4.</a>                 | <a href="#">Authorization Grant Example . . . . .</a>   | <a href="#">8</a>  |
| <a href="#">5.</a>                 | <a href="#">Interoperability Considerations . . . . .</a>   | <a href="#">9</a>  |
| <a href="#">6.</a>                 | <a href="#">Security Considerations . . . . .</a>   | <a href="#">9</a>  |
| <a href="#">7.</a>                 | <a href="#">IANA Considerations . . . . .</a>   | <a href="#">10</a> |
| 7.1.                               | Sub-Namespace Registration of<br>urn:ietf:params:oauth:grant-type:jwt-bearer . . . . .            | <a href="#">10</a> |
| 7.2.                               | Sub-Namespace Registration of<br>urn:ietf:params:oauth:client-assertion-type:jwt-bearer . . . . . | <a href="#">10</a> |
| <a href="#">8.</a>                 | <a href="#">References . . . . .</a>  | <a href="#">10</a> |
| <a href="#">8.1.</a>               | <a href="#">Normative References . . . . .</a>  | <a href="#">10</a> |
| <a href="#">8.2.</a>               | <a href="#">Informative References . . . . .</a>  | <a href="#">11</a> |
| <a href="#">Appendix A.</a>        | <a href="#">Acknowledgements . . . . .</a>  | <a href="#">11</a> |
| <a href="#">Appendix B.</a>        | <a href="#">Document History . . . . .</a>  | <a href="#">11</a> |
| <a href="#">Authors' Addresses</a> | <a href="#">. . . . .</a>   | <a href="#">13</a> |



## 1. Introduction

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 Client Authentication and Authorization Grants [[I-D.ietf-oauth-assertions](#)] specification 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 Client Authentication and Authorization Grants [[I-D.ietf-oauth-assertions](#)] specification 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 Profile for OAuth 2.0 Client Authentication and Authorization Grants [[I-D.ietf-oauth-saml2-bearer](#)] specification.

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 or keyed message digest 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**

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**

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

## **2. HTTP Parameter Bindings for Transporting Assertions**

The Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants [[I-D.ietf-oauth-assertions](#)] specification defines generic HTTP parameters for transporting Assertions (a.k.a. Security Tokens) during interactions with a token endpoint. This section defines specific parameters and treatments of those parameters for use with JWT bearer tokens.

### **2.1. Using JWTs as Authorization Grants**

To use a Bearer JWT as an authorization grant, use an access token request as defined in [Section 4](#) of the Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants [[I-D.ietf-oauth-assertions](#)] specification with the following specific 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 "scope" parameter may be used, as defined in the Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants [[I-D.ietf-oauth-assertions](#)] specification, to indicate the requested scope.

Authentication of the client is optional, as described in [Section 3.2.1](#) of OAuth 2.0 [[RFC6749](#)] and consequently, the "client\_id" is only needed when a form of client authentication that relies on the parameter is used.

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**

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

In order to issue an access token response as described in OAuth 2.0 [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.

1. The JWT MUST contain an "iss" (issuer) claim that contains a unique identifier for the entity that issued the JWT. Issuer values SHOULD be compared using the Simple String Comparison method defined in [Section 6.2.1 of RFC 3986](#) [RFC3986], unless otherwise specified by the application.
2. 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.
  - A. 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).
  - B. For client authentication, the subject MUST be the "client\_id" of the OAuth client.
3. The JWT MUST contain an "aud" (audience) claim containing a value that identifies the authorization server as an intended audience. The token endpoint URL of the authorization server MAY be used as a value for an "aud" element to identify the authorization server as an intended audience of the JWT. JWTs that do not identify the authorization server as an intended audience MUST be rejected. Audience values SHOULD be compared using the Simple String Comparison method defined in [Section 6.2.1 of RFC 3986](#) [RFC3986], unless otherwise specified by the



application.

4. 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, and reject expired JWTs. The authorization server MAY reject JWTs with an "exp" claim value that is unreasonably far in the future.
5. The JWT MAY contain an "nbf" (not before) claim that identifies the time before which the token MUST NOT be accepted for processing.
6. 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.
7. 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.
8. The JWT MAY contain other claims.
9. The JWT MUST be digitally signed or have a keyed message digest applied by the issuer. The authorization server MUST reject JWTs with an invalid signature or keyed message digest.
10. 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**

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. However, if client credentials are present in the request, the authorization server MUST validate them.

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**

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**

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

The example shows a JWT issued and signed by the system entity identified as "https://jwt-idp.example.com". The subject of the JWT is identified by email address as "mike@example.com". The intended audience of the JWT is "https://jwt-rp.example.net", which is an identifier with which the authorization server identifies itself. The JWT is sent as part of an access token request to the authorization server's token endpoint at "https://authz.example.net/token.oauth2".

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.
eyJpc3Mi[...omitted for brevity...].
J9l-ZhwP[...omitted for brevity...]
```

## 5. Interoperability Considerations

Agreement between system entities regarding identifiers, keys, and endpoints is required in order to achieve interoperable deployments of this profile. Specific items that require agreement are as follows: values for the issuer and audience identifiers, the location of the token endpoint, and the key used to apply and verify the digital signature or keyed message digest over the JWT. The exchange of such information is explicitly out of scope for this specification. In some cases, additional profiles may be created that constrain or prescribe these values or specify how they are to be exchanged. Examples of such profiles include the OAuth 2.0 Dynamic Client Registration Protocol [[I-D.ietf-oauth-dyn-reg](#)], OpenID Connect Dynamic Client Registration 1.0 [[OpenID.Registration](#)], and OpenID Connect Discovery 1.0 [[OpenID.Discovery](#)].

## 6. Security Considerations

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





## **7. IANA Considerations**

### **7.1. Sub-Namespace Registration of urn:ietf:params:oauth:grant-type:jwt-bearer**

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](#)].

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

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

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](#)].

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

## **8. References**

### **8.1. Normative References**

- [I-D.ietf-oauth-assertions] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", [draft-ietf-oauth-assertions](#) (work in progress), July 2013.
- [JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", [draft-ietf-oauth-json-web-token](#) (work in progress), July 2013.



- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), January 2005.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", [RFC 4627](#), July 2006.
- [RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), October 2012.
- [RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", [RFC 6755](#), October 2012.

## **[8.2. Informative References](#)**

- [I-D.ietf-oauth-dyn-reg]  
Richer, J., Bradley, J., Jones, M., and M. Machulak,  
"OAuth 2.0 Dynamic Client Registration Protocol",  
[draft-ietf-oauth-dyn-reg-13](#) (work in progress), July 2013.
- [I-D.ietf-oauth-saml2-bearer]  
Campbell, B., Mortimore, C., and M. Jones, "SAML 2.0  
Profile for OAuth 2.0 Client Authentication and  
Authorization Grants", [draft-ietf-oauth-saml2-bearer](#) (work  
in progress), July 2013.
- [OpenID.Discovery]  
Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID  
Connect Discovery 1.0", July 2013.
- [OpenID.Registration]  
Sakimura, N., Bradley, J., and M. Jones, "OpenID Connect  
Dynamic Client Registration 1.0", July 2013.

## **[Appendix A. Acknowledgements](#)**

This profile was derived from SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants [[I-D.ietf-oauth-saml2-bearer](#)] by Brian Campbell and Chuck Mortimore.

## **[Appendix B. Document History](#)**

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



-06

- o Stated that issuer and audience values SHOULD be compared using the Simple String Comparison method defined in [Section 6.2.1 of RFC 3986](#) unless otherwise specified by the application.

-05

- o Changed title from "JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0" to "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants" to be more explicit about the scope of the document per <http://www.ietf.org/mail-archive/web/oauth/current/msg11063.html>.
- o Numbered the list of processing rules.
- o Smallish editorial cleanups to try and improve readability and comprehensibility.
- o Cleaner split out of the processing rules in cases where they differ for client authentication and authorization grants.
- o Clarified the parameters that are used/available for authorization grants.
- o Added Interoperability Considerations section.
- o Added more explanatory context to the example in [Section 4](#).

-04

- o 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.
- o Added seriesInfo information to Internet Draft references.

-03

- o Reference [RFC 6749](#) and [RFC 6755](#).

-02

- o 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



- o Add examples to Sections [2.1](#) and [2.2](#)
- o Update references

-01

- o 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".
- o Merged in changes between [draft-ietf-oauth-saml2-bearer-11](#) and [draft-ietf-oauth-saml2-bearer-13](#). All changes were strictly editorial.

-00

- o Created the initial IETF draft based upon [draft-jones-oauth-jwt-bearer-04](#) with no normative changes.

#### Authors' Addresses

Michael B. Jones  
Microsoft

Email: [mbj@microsoft.com](mailto:mbj@microsoft.com)

URI: <http://self-issued.info/>

Brian Campbell  
Ping Identity

Email: [brian.d.campbell@gmail.com](mailto:brian.d.campbell@gmail.com)

Chuck Mortimore  
Salesforce

Email: [cmortimore@salesforce.com](mailto:cmortimore@salesforce.com)



