

**JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens**  
**draft-ietf-oauth-access-token-jwt-02**

Abstract

This specification defines a profile for issuing OAuth2 access tokens in JSON web token (JWT) format. Authorization servers and resource servers from different vendors can leverage this profile to issue and consume access tokens in interoperable manner.

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 <https://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 24, 2020.

Copyright Notice

Copyright (c) 2019 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 (<https://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="#">2</a>
<a href="#">1.1.</a>	<a href="#">Requirements Notation and Conventions</a>	<a href="#">3</a>
<a href="#">1.2.</a>	<a href="#">Terminology</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">JWT Access Token Header and Data Structure</a>	<a href="#">4</a>
<a href="#">2.1.</a>	<a href="#">Header</a>	<a href="#">4</a>
<a href="#">2.2.</a>	<a href="#">Data Structure</a>	<a href="#">4</a>
<a href="#">2.2.1.</a>	<a href="#">Identity Claims</a>	<a href="#">5</a>
<a href="#">2.2.2.</a>	<a href="#">Authorization Claims</a>	<a href="#">6</a>
<a href="#">2.2.2.1.</a>	<a href="#">Claims for Authorization Outside of Delegation Scenarios</a>	<a href="#">6</a>
<a href="#">3.</a>	<a href="#">Requesting a JWT Access Token</a>	<a href="#">6</a>
<a href="#">4.</a>	<a href="#">Validating JWT Access Tokens</a>	<a href="#">7</a>
<a href="#">5.</a>	<a href="#">Security Considerations</a>	<a href="#">9</a>
<a href="#">6.</a>	<a href="#">Privacy Considerations</a>	<a href="#">10</a>
<a href="#">7.</a>	<a href="#">IANA Considerations</a>	<a href="#">10</a>
<a href="#">7.1.</a>	<a href="#">Media Type Registration</a>	<a href="#">10</a>
<a href="#">7.1.1.</a>	<a href="#">Registry Content</a>	<a href="#">10</a>
<a href="#">7.2.</a>	<a href="#">Claims Registration</a>	<a href="#">11</a>
<a href="#">7.2.1.</a>	<a href="#">Registry Contents</a>	<a href="#">12</a>
<a href="#">8.</a>	<a href="#">References</a>	<a href="#">12</a>
<a href="#">8.1.</a>	<a href="#">Normative References</a>	<a href="#">12</a>
<a href="#">8.2.</a>	<a href="#">Informative References</a>	<a href="#">14</a>
<a href="#">Appendix A.</a>	<a href="#">Acknowledgements</a>	<a href="#">14</a>
<a href="#">Appendix B.</a>	<a href="#">Document History</a>	<a href="#">15</a>
	<a href="#">Author's Address</a>	<a href="#">16</a>

## [1. Introduction](#)

The original OAuth 2.0 Authorization Framework [[RFC6749](#)] specification does not mandate any specific format for access tokens. While that remains perfectly appropriate for many important scenario, in-market use has shown that many commercial OAuth2 implementations elected to issue access tokens using a format that can be parsed and validated by resource servers directly, without further authorization server involvement. The approach is particularly common in topologies where the authorization server and resource server are not co-located, are not ran by the same entity, or are otherwise separated by some boundary. All of the known commercial implementations known at this time leverage the JSON Web Tokens(JWT) [[RFC7519](#)] format.

Most vendor specific JWT access tokens share the same functional layout, including information in forms of claims meant to support the same scenarios: token validation, transporting authorization information in forms of scopes and entitlements, carrying identity information about the subject, and so on. The differences are mostly

Bertocci

Expires January 24, 2020

[Page 2]

confined to the claim names and syntax used to represent the same entities, suggesting that interoperability could be easily achieved by standardizing on a common set of claims and validation rules.

The assumption that access tokens are associated to specific information doesn't appear only in commercial implementations. Various specifications in the OAuth2 family (such as resource indicators [[ResourceIndicators](#)], bearer token usage [[RFC6750](#)] and others) postulate the presence in access tokens of scoping mechanisms, such as an audience. The family of specifications associated to introspection also indirectly suggest a fundamental set of information access tokens are expected to carry or at least be associated with.

This specification aims to provide a standardized and interoperable profile as an alternative to the proprietary JWT access tokens layouts going forward. Besides defining a common set of mandatory and optional claims, the profile provides clear indications on how authorization requests parameters determine the content of the issued JWT access token, how an authorization server can publish metadata relevant to the JWT access tokens it issues, and how a resource server should validate incoming JWT access tokens.

Finally, this specification provides security and privacy considerations meant to prevent common mistakes and anti patterns that are likely to occur in naive use of the JWT format to represent access tokens.

### **[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 [BCP 14](#) [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

### **[1.2.](#) Terminology**

**JWT access token** An OAuth 2.0 access token encoded in JWT format and complying with the requirements described in this specification.

This specification uses the terms "access token", "refresh token", "authorization server", "resource server", "authorization endpoint", "authorization request", "authorization response", "token endpoint", "grant type", "access token request", "access token response", and "client" defined by The OAuth 2.0 Authorization Framework [[RFC6749](#)].



## **2. JWT Access Token Header and Data Structure**

JWT access tokens are regular JWT tokens complying with the requirements described in this section.

### **2.1. Header**

Although JWT access tokens can use any signing algorithm, use of asymmetric algorithms is RECOMMENDED as it simplifies the process of acquiring validation information for resource servers (see [Section 4](#)).

NOTE: there were discussions about adding a reference to authenticated encryption methods as well, but there's no internet draft specifying interoperable public key methods at this time

The typ header parameter for a JWT access token MUST be at+jwt. See the security considerations section for details on the importance of preventing JWT access tokens to be interpreted as id\_tokens.

### **2.2. Data Structure**

The following claims are used in the JWT access token data structure.

iss REQUIRED - as defined in [section 4.1.1 of \[RFC7519\]](#).

exp REQUIRED - as defined in [section 4.1.4 of \[RFC7519\]](#).

aud REQUIRED - as defined in [section 4.1.3 of \[RFC7519\]](#). See [Section 3](#) for indications on how an authorization server should determine the value of aud depending on the request. The aud claim MAY include a list of individual resource indicators if they are all aliases referring to the same requested resource known by the authorization server.

sub REQUIRED - as defined in [section 4.1.2 of \[RFC7519\]](#). In case of access tokens obtained through grants where no resource owner is involved, such as the client credentials grant, the value of sub SHOULD correspond to an identifier the authorization server uses to indicate the client application. Please see [Section 5](#) for more details on this scenario.

client\_id REQUIRED - as defined in section 4.3 of [\[TokenExchange\]](#).

iat OPTIONAL - as defined in [section 4.1.6 of \[RFC7519\]](#).

auth\_time OPTIONAL - as defined in section 2 of [\[OpenID.Core\]](#).

Important: as this claim represents the time at which the end user



last authenticated, its value will either remain the same for all the JWT access tokens issued within that session or be updated to the time of latest authentication if reauthentication occurred mid-session (as it is the case for step up authentication and similar occurrences). For example: all the JWT access tokens obtained with a given refresh token will all have the same value of `auth_time`, corresponding to the instant in which the user first authenticated to obtain the refresh token.

`jti` OPTIONAL - as defined in [section 4.1.7 of \[RFC7519\]](#).

`acr`, `amr` OPTIONAL - as defined in section 2 of [\[OpenID.Core\]](#). The same considerations presented for `auth_time` apply to `acr` and `amr`: those values reflect the authentication context and method used when the end user originally authenticated, and will remain unchanged for the JWT access tokens issued within the context of that session.

### **[2.2.1](#). Identity Claims**

Commercial authorization servers will often include resource owner attributes directly in access tokens, so that resource servers can consume them directly for authorization or other purposes without any further roundtrips to introspection ([\[RFC7662\]](#)) or userinfo ([\[OpenID.Core\]](#)) endpoints. This is particularly common in scenarios where the client and the resource server belong to the same entity and are part of the same solution, as it is the case for first party clients invoking their own backend API.

This profile does not introduce any mechanism for a client to directly request the presence of specific claims in JWT access tokens, as the authorization server can determine what additional claims are required by a particular resource server by taking in consideration the `client_id` of the client, the scope and the resource parameters included in the request.

Any additional attributes whose semantic is well described by the attributes description found in section 5.1 of [\[OpenID.Core\]](#) SHOULD be codified in JWT access tokens via the corresponding claim names in that section of the OpenID Connect specification. The same holds for attributes defined in [\[RFC7662\]](#) and other identity related specifications.

Authorization server MAY return arbitrary attributes not defined in any existing specification, as long as the corresponding claim names are collision resistant or the access tokens are meant to be used only within a private subsystem.





Authorization servers including resource owner attributes in JWT access tokens should exercise care and verify that all privacy requirements are met, as discussed in [Section 6](#).

### **2.2.2. Authorization Claims**

If an authorization request includes a scope parameter, the corresponding issued JWT access token SHOULD include a scope claim as defined in section 4.2 of [[TokenExchange](#)].

All the individual scopes strings in the scope claim MUST have meaning for the resource indicated in the aud claim.

#### **2.2.2.1. Claims for Authorization Outside of Delegation Scenarios**

Many authorization servers embed in the access tokens they issue authorization attributes that go beyond the delegated scenarios described by [[RFC7519](#)]. Typical examples include resource owner memberships in roles and groups that are relevant to the resource being accessed, entitlements assigned to the resource owner for the targeted resource that the authorization server knows about, and so on.

An authorization server wanting to include such attributes in a JWT access token SHOULD use as claim types the attributes described by [section 4.1.2](#) of SCIM Core ([[RFC7643](#)]) and in particular roles, groups and entitlements. As in their original definition in [[RFC7643](#)] , this profile does not provide a specific vocabulary for those entities. The [Section 7](#) section of this document does provide entries for registering the roles, groups and entitlements attributes from [[RFC7643](#)] as claim types to be used in this profile.

## **3. Requesting a JWT Access Token**

An authorization server can issue a JWT access token in response to any authorization grant defined by [[RFC6749](#)] and subsequent extensions meant to result in an access token.

Every JWT access token MUST include an aud claim (see [Section 2.2](#)).

If the request includes a resource parameter (as defined in [[ResourceIndicators](#)]), the resulting JWT access token aud claim MUST have the same value as the resource parameter in the request.

Example request below:



```
GET /as/authorization.oauth2?response_type=token
    &client_id=s6BhdRkqt3&state=laeb
    &scope=openid%20profile%20reademail
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
    &resource=https%3A%2F%2Frs.example.com%2F HTTP/1.1
Host: authorization-server.example.com
```

Figure 1: Authorization Request with Resource and Scope Parameters

Once redeemed, the code obtained from the request above will result in a JWT access token in the form shown below:

```
{ "typ": "at+JWT", "alg": "RS256", "kid": "RjEwOwOA" }
{
  "iss": "https://authorization-server.example.com/",
  "sub": " 5ba552d67",
  "aud": "https://rs.example.com/",
  "exp": 1544645174,
  "client_id": "s6BhdRkqt3_",
  "scope": "openid profile reademail"
}
```

Figure 2: A JWT Access Token

If it receives a request for an access token containing more than one resource parameter, an authorization server issuing JWT access tokens **MUST** reject the request and fail with `invalid_request` as described in [section 4.1.2.1 of \[RFC6749\]](#). See [Section 2.2](#) and [Section 5](#) for more details on how this measure ensures there's no confusion on to what resource the access token granted scopes apply.

If the request does not include a resource parameter, the authorization server **MUST** use in the `aud` claim a default resource indicator. If a scope parameter is present in the request, the authorization server **SHOULD** use it to infer the value of the default resource indicator to be used in the `aud` claim. The mechanism through which scopes are associated to default resource indicator values is outside the scope of this specification. If the values in the scope parameter refer to different default resource indicator values, the authorization server **SHOULD** reject the request with `invalid_scope` as described in [section 4.1.2.1 of \[RFC6749\]](#).

#### 4. Validating JWT Access Tokens

For the purpose of facilitating validation data retrieval, it is **RECOMMENDED** that authorization servers sign JWT access tokens with an asymmetric algorithm.



Authorization servers SHOULD implement OAuth 2.0 Authorization Server Metadata [[RFC8414](#)] to advertise to resource servers its signing keys via `jwks_uri` and what `iss` claim value to expect via the issuer metadata value. Alternatively, authorization servers implementing OpenID connect MAY use the Openid connect discovery document for the same purpose. If an authorization server supports both AS metadata and Openid discovery, the values provided MUST be consistent across the two publication methods.

An authorization server MAY elect to use different keys to sign `id_tokens` and JWT access tokens.

When invoked as described in OAuth2 bearer token usage, resource servers receiving a JWT access token MUST validate it in the following manner.

1. The resource server MUST verify that the `typ` header value is `at+jwt` and reject tokens carrying any other value.
2. If the JWT access token is encrypted, decrypt it using the keys and algorithms that the resource server specified during registration. If encryption was negotiated with the authorization server at registration time and the incoming JWT access token is not encrypted, the resource server SHOULD reject it.
3. The Issuer Identifier for the authorization server (which is typically obtained during discovery) MUST exactly match the value of the `iss` claim.
4. The resource server MUST validate that the `aud` claim contains the resource indicator value corresponding to the identifier the resource server expects for itself. The `aud` claim MAY contain an array with more than one element. The JWT access token MUST be rejected if `aud` does not list the resource indicator of the current resource server as a valid audience, or if it contains additional audiences that are not known aliases of the resource indicator of the current resource server.
5. The resource server MUST validate the signature of all incoming JWT access token according to [[RFC7515](#)] using the algorithm specified in the JWT `alg` Header Parameter. The resource server MUST use the keys provided by the authorization server.
6. The current time MUST be before the time represented by the `exp` Claim.



7. If the `auth_time` claim is present, the resource server SHOULD check the `auth_time` value and request re-authentication if it determines too much time has elapsed since the last resource owner authentication.

If the JWT access token includes authorization claims as described in the authorization claims section, the resource server SHOULD use them in combination with any other contextual information available to determine whether the current call should be authorized or rejected. Details about how a resource server performs those checks is beyond the scope of this profile specification.

## 5. Security Considerations

The JWT access token data layout described here is very similar to the one of the `id_token` as defined by [\[OpenID.Core\]](#). Without the explicit typing required in this profile, in line with the recommendations in [\[JWT.BestPractices\]](#) there would be the risk of attackers using JWT access tokens in lieu of `id_tokens`.

Authorization servers should prevent scenarios where clients can affect the value of the `sub` claim in ways that could confuse resource servers. For example: if the authorization server elects to use the `client_id` as the `sub` value for access tokens issued client credentials grant, the authorization server should prevent clients to register an arbitrary `client_id` value, as this would allow malicious clients to select the `sub` of a high privilege resource owner and confuse any authorization logic on the resource server relying on the `sub` value. For more details please refer to section 4.13 of [\[OAuth2.Security.BestPractices\]](#).

To preventing cross-JWT confusion, authorization servers MUST use a distinct identifier as `"aud"` claim value to uniquely identify access tokens issued by the same issuer for distinct resources.

This profile explicitly forbids the use of multi value `aud` claim when the individual values refer to different resources, as that would introduce confusion about what scopes apply to which resource-possibly opening up avenues for elevation of delegated privileges attacks. Alternative techniques to prevent scope confusion include "scope stuffing", imposing to every individual scope string to include a reference to the resource they are meant to be applied to, but its application is problematic (scope opacity violations, size inflation, more error conditions become possible when the combination of requested scopes and resource indicators is invalid) and the observed frequency of the scenario doesn't warrant complicating the more common cases.





## **6. Privacy Considerations**

As JWT access tokens carry information by value, it now becomes possible for requestors and receivers to directly peek inside the token claims collection. The client **MUST NOT** inspect the content of the access token: the authorization server and the resource server might decide to change token format at any time (for example by switching from this profile to opaque tokens) hence any logic in the client relying on the ability to read the access token content would break without recourse. Nonetheless, authorization servers should not assume that clients will comply with the above. Whenever client access to the access token content presents privacy issues for a given scenario, the authorization server should take explicit steps to prevent it as described above.

In scenarios in which JWT access tokens are accessible to the end user, it should be evaluated whether the information can be accessed without privacy violations (for example, if an end user would simply access his or her own personal information) or if steps must be taken to enforce confidentiality. Possible measure include: encrypting the access token, encrypting the sensitive claims, omitting the sensitive claims or not using this profile, falling back on opaque access tokens.

In every scenario, the content of the JWT access token will eventually be accessible to the resource server. It's important to evaluate whether the resource server gained the proper entitlement to have access to any content received in form of claims, for example through user consent in some form, policies and agreements with the organization running the authorization servers, and so on.

## **7. IANA Considerations**

### **7.1. Media Type Registration**

#### **7.1.1. Registry Content**

This section registers the "application/at+jwt" media type [[RFC2046](#)] in the "Media Types" registry [IANA.MediaType] in the manner described in [[RFC6838](#)], which can be used to indicate that the content is an access token encoded in JWT format.

- o Type name: application
- o Subtype name: at+jwt
- o Required parameters: N/A



- o Optional parameters: N/A
- o Encoding considerations: binary; JWT values are encoded as a series of base64url-encoded values (with trailing '=' characters removed), some of which may be the empty string, separated by period ('.') characters.
- o Security considerations: See the Security Considerations Section of `[[TODO: update once there's a RFC number for the JWT AT profile]]`
- o Interoperability considerations: N/A
- o Published specification: `[[TODO: update once there's a RFC number for the JWT AT profile]]`
- o Applications that use this media type: Applications that access resource servers using OAuth2 access tokens encoded in JWT format
- o Fragment identifier considerations: N/A
- o Additional information: Magic number(s): N/A File extension(s): N/A Macintosh file type code(s): N/A
- o Person and email address to contact for further information: Vittorio Bertocci, `vittorio@auth0.com`
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author: Vittorio Bertocci, `vittorio@auth0.com`
- o Change controller: IESG
- o Provisional registration? No

## **7.2. Claims Registration**

Section [Section 2.2.2.1](#) of this specification refers to the attributes "roles", "groups", "entitlements" defined in [\[RFC7643\]](#) to express authorization information in JWT access tokens. This section registers those attributes as claims in the JSON Web Token (JWT) IANA registry introduced in [\[RFC7519\]](#).



### **7.2.1. Registry Contents**

- o Claim Name: "roles"
- o Claim Description: Roles
- o Change Controller: IESG
- o Specification Document(s): [section 4.1.2 of \[RFC7643\]](#)
- o Claim Name: "groups"
- o Claim Description: Groups
- o Change Controller: IESG
- o Specification Document(s): [section 4.1.2 of \[RFC7643\]](#)
- o Claim Name: "entitlements"
- o Claim Description: Entitlements
- o Change Controller: IESG
- o Specification Document(s): [section 4.1.2 of \[RFC7643\]](#)

## **8. References**

### **8.1. Normative References**

- [IANA.OAuth.Parameters]  
IANA, "OAuth Parameters",  
<<http://www.iana.org/assignments/oauth-parameters>>.
- [JWT.BestPractices]  
Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", November 2018.
- [OAuth2.Security.BestPractices]  
Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett,  
"OAuth 2.0 Security Best Current Practice", July 2019.
- [OpenID.Core]  
Sakimura, N., Bradley, J., Jones, M., Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014.



## [ResourceIndicators]

Campbell, B., Bradley, J., and H. Tschofenig, "OAuth 2.0 Token Exchange", November 2016.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", [RFC 2046](#), DOI 10.17487/RFC2046, November 1996, <<https://www.rfc-editor.org/info/rfc2046>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", [BCP 13](#), [RFC 6838](#), DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", [RFC 7515](#), DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.

[RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", [RFC 7643](#), DOI 10.17487/RFC7643, September 2015, <<https://www.rfc-editor.org/info/rfc7643>>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", [RFC 8414](#), DOI 10.17487/RFC8414, June 2018, <<https://www.rfc-editor.org/info/rfc8414>>.





[TokenExchange]

Nadalin, A., Bradley, J., Jones, M., Campbell, B., and C. Mortimore, "OAuth 2.0 Token Exchange", October 2018.

## 8.2. Informative References

- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", [RFC 6750](#), DOI 10.17487/RFC6750, October 2012, <<https://www.rfc-editor.org/info/rfc6750>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", [RFC 7519](#), DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.
- [RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", [RFC 7644](#), DOI 10.17487/RFC7644, September 2015, <<https://www.rfc-editor.org/info/rfc7644>>.
- [RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", [RFC 7662](#), DOI 10.17487/RFC7662, October 2015, <<https://www.rfc-editor.org/info/rfc7662>>.

## Appendix A. Acknowledgements

The initial set of requirements informing this specification was extracted by numerous examples of access tokens issued in JWT format by production systems. Thanks to Dominick Bauer (IdentityServer), Brian Campbell (PingIdentity), Daniel Dobalian (Microsoft), Karl Guinness (Okta) for providing sample tokens issued by their products and services. Brian Campbell and Filip Skokan provided early feedback that shaped the direction of the specification. This profile was discussed at length during the OAuth Security Workshop 2019, with several individuals contributing ideas and feedback. The author would like to acknowledge the contributions of:

John Bradley, Brian Campbell, Vladimir Dzhuvinov, Torsten Lodderstedt, Nat Sakimura, Hannes Tschofenig and everyone who actively participated in the unconference discussions.

The following individuals contributed useful feedback and insights on the initial draft, both on the IETF OAuth2 WG DL and during the IIW28 conference:

Dale Olds, George Fletcher, David Waite, Michael Engan, Mike Jones, Hans Zandbelt, Vladimir Dzhuvinov, Martin Schanzenbach and everyone



who actively participated in the IIW28 unconference discussions and the IETF OAuth2 WG DL discussions.

## **Appendix B. Document History**

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

### [draft-ietf-oauth-access-token-jwt-02](#)

- o In 2.2.1, opened the sources of identity attributes to any identity related specification.
- o In 2.2.2, relaxed from MUST to SHOULD the requirement that requests including a scope always result in access tokens containing a corresponding scope claim.
- o In the security considerations setting, added a requirement for the authorization server to assign unique identifiers for different resources- to prevent cross JWT confusion.
- o Added IANA registration for the authorization attributes borrowed from SCIM CORE

### [draft-ietf-oauth-access-token-jwt-01](#)

- o Added note on authenticated encryption.
- o Added a mention to the 1st party clients scenarios in the identity claims section.
- o Changed the definition reference for the iss, exp, aud, sub, iat claims from OpenID.Core to [RFC7519](#).
- o Added a mention of the client\_id==sub case in the security considerations section, added a reference to [draft-ietf-oauth-security-topics-13](#). Added a reference to the security considerations from the sub claim definition section.
- o Specified invalid\_request as the error code the authorization server should return in case of multiple resources in the access token request.
- o Specified invalid\_scope as the error code the authorization server should return in case it isn't possible to determine to which resource the requested scopes refers to.
- o In the identity claims section, added a reference to introspection as possible source of claim types and added language explicitly stating that the AS can add arbitrary attributes as long as they are collision resistant or private.
- o Updated language for the auth\_time claim to include the case in which the AS reauthenticates the user mid-session (e.g. during step up auth).
- o Removed note about adding a mechanism for establishing whether the token was obtained on behalf of the resource owner or of the client itself (client credentials grant).



- o Removed note about adding a mechanism for indicating whether the authorization server sent the resource owner to authenticate with a federated identity provider, and the identity of that federated provider.
- o Removed the note in the security consideration sections about discussing the purpose of aud, iss, exp validation (redundant).
- o In the authorization claims section, stated intent to register roles, groups and entitlements as claim types in IANA
- o Clarified in the privacy considerations that clients should not inspect access tokens.
- o Expanded the privacy considerations with more explicit guidance about privacy preserving approaches.
- o Added IANA registry content for the at+JWT MIME type.
- o Updated acknowledgements.

[draft-ietf-oauth-access-token-jwt-00](#)

- o Initial draft to define a JWTt profile for OAuth 2.0 access tokens.

#### Author's Address

Vittorio Bertocci  
Auth0

Email: vittorio@auth0.com

