

**OAuth 2.0 Token Revocation List**  
**draft-gpujol-oauth-atr1-01**

Abstract

This document defines a format and a standardised uri for a Token Revocation List. An OAuth 2.0 authorization server can use those to expose a current list of revoked access tokens identifiers that it previously issued, intended for use by OAuth 2.0 resource servers.

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 March 14, 2019.

Copyright Notice

Copyright (c) 2018 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>	Introduction . . . . .	<a href="#">2</a>
<a href="#">1.1.</a>	Notational Conventions . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">2.1.</a>	Token Revocation List . . . . .	<a href="#">4</a>
<a href="#">2.2.</a>	Symbols and abbreviated terms . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Authorization server metadata . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Token Revocation List Format . . . . .	<a href="#">4</a>
<a href="#">5.</a>	Token Revocation List Request . . . . .	<a href="#">6</a>
<a href="#">6.</a>	Token Revocation List Response . . . . .	<a href="#">6</a>
<a href="#">7.</a>	Security Considerations . . . . .	<a href="#">7</a>
<a href="#">8.</a>	IANA Considerations . . . . .	<a href="#">7</a>
<a href="#">8.1.</a>	OAuth 2.0 Authorization Server Metadata . . . . .	<a href="#">7</a>
<a href="#">8.1.1.</a>	Registry Contents . . . . .	<a href="#">7</a>
<a href="#">9.</a>	Revision History . . . . .	<a href="#">7</a>
<a href="#">10.</a>	Normative References . . . . .	<a href="#">8</a>
	Author's Address . . . . .	<a href="#">9</a>

**[1.](#) Introduction**

OAuth 2.0 Token Revocation [[RFC7009](#)] defines a way for OAuth 2.0 clients to revoke access tokens and refresh tokens issued by an OAuth 2.0 authorization server. While refresh token are typically only used by the authorization server itself, access tokens on the other hand are consumed by OAuth 2.0 resource servers; those are logically separated from the authorization server, and must learn about the revocation status of the access tokens they receive, if the local security requirements mandates it.

Some deployments of OAuth 2.0 (or derived protocols) use signed JWT as access tokens. Those access tokens are self-sufficient for resource server to validate that the access token is currently not expired, but provide no means for the resource server to obtain the revocation status of the token.

OAuth 2.0 Token Introspection [[RFC7662](#)] defines a way for resource servers to obtain the metadata attached to a given access token. For performance reasons, this metadata may be put in cache by the resource server, instead of calling the introspection endpoint synchronously every time the same access token is received. This voids the possibility for the resource server to be informed about a revocation of an access token that occurs after the first call to the introspection endpoint.

This specification defines a Token Revocation List (TRL), a document exposed by the OAuth 2.0 authorization server, containing a list of revoked access tokens identifiers. Resource servers can periodically



retrieve that list to obtain the revocation status of access tokens. By doing so, they can either flag currently cached introspected token metadata as revoked, or avoid unnecessary calls to the introspection endpoint for unknown tokens that are already expired when they are received by the resource server.

This allows better performance for the authorization server and lower response times for resource requests [[RFC6750](#)], since:

for resource server this allow caching the token introspection response until the expiration date of the access token. The TRL can be retrieved asynchronously to actual resource requests, so the round trip to the authorization server does not add up to the resource server response time.

for the authorization server this avoids unnecessary calls to the introspection endpoint

This is especially important in scenarios where the authorization server issues relatively long lived access tokens, and the authorization server and resource servers are loosely coupled (e.g. User Managed Access [UMA]), and the introspection endpoint is heavily used.

Note that using short-lived access tokens should be the preferred way to protect sensitive resources, rather than relying on the Token Revocation List. Issuing a TRL does not provide any assurance that resource servers will use it, nor does provide real time access token revocation; depending on their configuration, resource servers might take a few seconds or minutes to obtain a fresh TRL after any given token is expired. During that time, access tokens may still be considered as active by resource servers. This hardly avoidable delay may however be better than not checking the revocation status at all.

### **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](#)].

## **2. Terminology**

For the purposes of this specification, the following terms and definitions in addition to what is defined in OAuth 2.0 Framework [[RFC6749](#)], Authorization server [[RFC6749](#)], Resource Server [[RFC6749](#)], Access Token [[RFC6749](#)], and JSON Web Encryption [[RFC7519](#)] apply.



### **2.1. Token Revocation List**

JWT [[RFC7519](#)] that holds a list of access tokens identifiers issued by a given authorization server, that are revoked but not expired at the time it is issued.

### **2.2. Symbols and abbreviated terms**

The following abbreviations are common to this specification.

JWT JSON Web Token

URI Uniform Resource Identifier

URL Uniform Resource Locator

TRL Token Revocation List

## **3. Authorization server metadata**

Authorization servers can have metadata describing their configuration. The following authorization server metadata value is used by this specification and are registered in the IANA "OAuth Authorization Server Metadata" registry established in [Section 7.1](#) of OAuth 2.0 Authorization Server Metadata [[RFC8414](#)]:

token\_revocation\_list\_uri OPTIONAL. URL [[RFC3986](#)] of an authorization server, where resource servers can retrieve the Token Revocation List. This url MUST use the https scheme. The referenced document contains a JWT whose payload contains the list of currently revoked access tokens. The exact contents of this JWT is defined in the present specification.

If an authorization server advertises the presence of its token revocation list, resource servers SHOULD use it as their preferred way to obtain the revocation status of access tokens, rather than using the token introspection endpoint.

## **4. Token Revocation List Format**

A Token Revocation List (TRL) is a JWT. The JWT payload MUST contain the following claims:

iss REQUIRED. MUST be the authorization server issuer

iat REQUIRED. time at which the TRL was generated, as defined in the



exp REQUIRED. as defined in [[RFC7519](#)], time at which the TRL SHOULD be considered as expired by resource servers.

rev\_token\_ids REQUIRED. a JSON list whose items are the identifiers of the revoked and non-expired access tokens at the time the TRL was generated.

The JWT payload MAY contain other claims. Claims not known or not understood by resource servers MUST be ignored.

Token identifiers in the "rev\_token\_ids" MUST be unique token identifiers, allowing resource server to unambiguously identify revoked access tokens. For access tokens in JWT format, those identifiers MUST match "jti" values of issued tokens. For access tokens in CWT format, this must match the "cid" values. For access tokens in other formats, "rev\_token\_ids" claim MAY include any kind of token identifier, depending on the format of access tokens issued by the authorisation server, as long as both resource servers and authorization server have beforehand agreement on the identifier.

To allow resource servers to learn about the revocation status of a token, the resource server must be able to obtain this unique token identifier. This can be done either by reading the jti or cti claim from access tokens that are in JWT or CWT formats, respectively, or by reading the jti attribute as returned by the authorization server introspection endpoint.

The TRL exposed by an OAuth authorization server at any given time MUST always have an expiration date (exp) in the future and SHOULD have an expiration date reasonably far away in the future. The exact frequency for an authorization server to generate a new TRL and the lifetime of generated TRLs is deployment specific and is out of scope of this specification. Some authorization servers MAY choose to generate a new TRL every time an access token is revoked, while others MAY generate a new TRL periodically at a fixed time period. Some MAY also choose to generate a new TRL only when an access token considered as security sensitive is revoked (e.g. bound to a scope that is internally flagged as sensitive).

A TRL SHOULD contain only a list of token identifiers that are revoked and not expired at the time the TRL is generated. A TRL MUST NOT contain a token identifier of a token that has not been revoked before the TRL is generated. A TRL MAY contain some token identifiers that have been revoked but are already expired at the time the TRL is generated.

A TRL MUST be signed. That signature MUST use an asymmetric algorithm, and the JWT header MUST contain both the "alg" and the



"kid" claims as defined in [RFC7515](#) [[RFC7515](#)]; the JWK referenced by this kid MUST be part of the JWKS [[RFC7517](#)] exposed by the authorization server on its `jwtks_uri`.

A TRL MUST NOT be encrypted.

## 5. Token Revocation List Request

An authorization server Token Revocation List MUST be queried using an HTTP "GET" request [[RFC7230](#)] at the URL defined in the authorization server metadata.

The following is a non-normative example TRL request (with line breaks for display purposes only):

```
GET /token_revocation_list HTTP/1.1
Host: server.example.com
```

Once a resource server obtains a TRL and that TRL reaches its expiration date, a resource server SHOULD obtain a new TRL from the authorization server. A resource server MAY obtain a new TRL before the last TRL in its possession is expired. The frequency at which the resource server obtains updated TRLs is out of the scope of this specification, and depends on the resource server security requirements.

## 6. Token Revocation List Response

An authorization server responds to a Token Revocation List request with the most recently available TRL and the Content-Type "application/jwt".

The following is a non-normative example TRL response (with line breaks for display purposes only):

```
200 OK
Content-Type: application/json
Content-Length: 542
```

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IlRSTF8yMDE4MDcxMCJ9.eyJleHAiOiAxNTMxMzIzMDY0LjI3NzY2LCAnYW0iOiJzOGMTUzMTMyMzAwOC4yNzczNiwiGj2lzcyc6ICdZZXJ2ZXIuZXBhbXBsZS5jb20nLCAnbmV2X3Rva2VuX2lkcyY6IFsnMTIzNDU2NzkwJywgJzIzNDU2Nzg5MDE4MDcxMCJ9.LdMV-QMRsXwHxI4ZfQvQEv5_wNe22VHBA5x6CIbG-3H-0R2nMnB_tNeA8nngNNo_vdDRj6Z25Bu6w1TQ0M8VufPbUGyAM1Q3LLjPU8pcEnM79Z8LW305M09ILaumgg94HrFSTPyEnlIGkVFF_x2vYTF-FbYEF1z2he3WDatOXXXh9gVlfTeinPwVtEzV-k740nUHVJjoSLSS7f_ZVmRnT_wUF_Wisx5YRtrkcu8bXJqEyksWgYmrzxewCHsc03qEV3HwQPc15_MJBF8tQT9vLTwnYSdMXJh9J5uREzIEFqBQpQyIAtbqVT7eD90MQ0ttwfb5LVtlNAVHRRdFVkuWA
```



## **7. Security Considerations**

The TRL is publicly exposed to anyone, not just authorized resource servers. However the TRL does not contain security sensitive values, just identifiers of tokens, that are not sufficient by themselves to gain any unauthorized access to any protected resource. An alternative would be to require some form of authentication of TRL clients (which in this case, are themselves resource servers). However, the added complexity, and performance impact (however small) of managing that authentication would curb usage of the TRL.

The TRL MUST be signed by its issuer (the authorization server). The resource servers MUST validate the signature before trusting the contents of the TRL. This avoids misuse or denial of service when the party controlling the https (server-side) connection (which, in complex environments, may be different than the entity controlling the authorization servers) removes revoked tokens from the TRL, and/or issues a TRL containing non-revoked tokens. To effectively protect against such attacks, the `jwt_keys_uri` exposing the public keys must not be exposed by the same party as the `token_revocation_list_uri`.

## **8. IANA Considerations**

### **8.1. OAuth 2.0 Authorization Server Metadata**

This specification registers the following values in the IANA "OAuth 2.0 Authorization Server Metadata" registry [IANA.OAuth.Parameters] established by [RFC8414].

#### **8.1.1. Registry Contents**

- o Metadata name: `token_revocation_list_uri`
- o Metadata Description: The Token Revocation List Uri.
- o Change controller: IESG
- o Specification Document: [Section 3](#) of [[ this specification ]]

## **9. Revision History**

Note to the RFC Editor: Please remove this section from the final RFC.

-01

- o TRLs MUST be signed.
- o Single TRL claim "rev\_token\_ids" for all kind of token identifiers



- o Token IDs can include "jti" from JWTs, "cti" from CWT, or other form of ids.

## 10. Normative References

- [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>>.
- [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>>.
- [RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", [RFC 7009](#), DOI 10.17487/RFC7009, August 2013, <<https://www.rfc-editor.org/info/rfc7009>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", [RFC 7230](#), DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [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>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", [RFC 7517](#), DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.
- [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>>.
- [RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", [RFC 7662](#), DOI 10.17487/RFC7662, October 2015, <<https://www.rfc-editor.org/info/rfc7662>>.



- [RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", [RFC 8392](#), DOI 10.17487/RFC8392, May 2018, <<https://www.rfc-editor.org/info/rfc8392>>.
- [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>>.

#### Author's Address

Guillaume Pujol  
Digital Security  
France

Email: [guill.p.linux@gmail.com](mailto:guill.p.linux@gmail.com)

