

Web Authentication Protocol  
Internet-Draft  
Intended status: Standards Track  
Expires: November 27, 2012

T. Lodderstedt, Ed.  
Deutsche Telekom AG  
S. Dronia

M. Scurtescu  
Google  
May 26, 2012

**Token Revocation**  
**draft-ietf-oauth-revocation-00**

Abstract

This draft proposes an additional endpoint for OAuth authorization servers for revoking tokens.

Requirements Language

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](#)].

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 November 27, 2012.

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

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Token Revocation . . . . .	<a href="#">3</a>
<a href="#">2.1.</a>	JSONP Support . . . . .	<a href="#">5</a>
<a href="#">3.</a>	Acknowledgements . . . . .	<a href="#">5</a>
<a href="#">4.</a>	IANA Considerations . . . . .	<a href="#">5</a>
<a href="#">5.</a>	Security Considerations . . . . .	<a href="#">5</a>
<a href="#">6.</a>	References . . . . .	<a href="#">6</a>
<a href="#">6.1.</a>	Normative References . . . . .	<a href="#">6</a>
<a href="#">6.2.</a>	Informative References . . . . .	<a href="#">6</a>
	Authors' Addresses . . . . .	<a href="#">6</a>



## 1. Introduction

The OAuth 2.0 core specification [[I-D.ietf-oauth-v2](#)] defines several ways for a client to obtain refresh and access tokens. This specification supplements the core specification with a mechanism to revoke both types of tokens and facilitates the following use cases:

- o The end-user triggers revocation from within the client that sends the appropriate revocation request to the authorization server. From the end-user's perspective, this looks like a "logout" or "reset" function. The request causes the removal of the client permissions associated with the particular token to access the end-user's protected resources. This use case makes it even more comfortable to the end-user to revoke his access grant immediately via the client.
- o In contrast to revocation by a client, the authorization server (or a related entity) may offer its end-users a self-care portal to delete access grants given to clients independent of any token storing devices. Such a portal offers the possibility to an end-user to look at and revoke all access grants he once authorized. In cases the token storing device is not available, e.g. it is lost or stolen, revocation by a self-care portal is the only possibility to limit or avoid abuse.

In the end, security, usability, and ease of use are increased by token revocation.

By using an additional endpoint, the token revocation endpoint, clients can request the revocation of a particular token. Compliant implementation **MUST** support the revocation of refresh tokens, access token revocation **MAY** be supported.

## 2. Token Revocation

The client requests the revocation of a particular token by making an HTTP POST request to the token revocation endpoint. The location of the token revocation endpoint can be found in the authorization servers documentation. The token endpoint URI **MAY** include a query component.

Since requests to the token revocation endpoint result in the transmission of plain text credentials in the HTTP request, the authorization server **MUST** require the use of a transport-layer security mechanism when sending requests to the token revocation endpoints. The authorization server **MUST** support TLS 1.0 ([\[RFC2246\]](#)), **SHOULD** support TLS 1.2 ([\[RFC5246\]](#)) and its future



replacements, and MAY support additional transport-layer mechanisms meeting its security requirements.

The client constructs the request by including the following parameters using the "application/x-www-form-urlencoded" format in the HTTP request entity-body:

token REQUIRED. The token that the client wants to get revoked.  
Note: the authorization server is supposed to detect the token type automatically.

The client also includes its authentication credentials as described in Section 2.3. of [[I-D.ietf-oauth-v2](#)].

For example, a client may request the revocation of a refresh token with the following request (line breaks are for display purposes only):

```
POST /revoke HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
```

```
token=45ghiukldjahdnhdauz&
```

The authorization server first validates the client credentials (if present) and verifies whether the client is authorized to revoke the particular token based on the client identity and its policy. For example, only the client the token has been issued for might be allowed to revoke it. It is also conceivable to allow a dedicated user self-care portal to revoke all kinds of tokens.

In the next step, the authorization server invalidates the token. Whether the revocation takes effect instantly or with some delay depends on the architecture of the particular deployment. The client MUST NOT make any assumptions about the timing and MUST NOT use the token again.

If the processed token is a refresh token and the authorization server supports the revocation of access tokens, then the authorization server SHOULD also invalidate all access tokens issued for that refresh token.

The authorization server indicates a successful processing of the request by a HTTP status code 200. Status code 401 indicates a failed client authentication, whereas a status code 403 is used if the client is not authorized to revoke the particular token. For all other error conditions, a status code 400 is used along with an error



response as defined in section 5.2. of [[I-D.ietf-oauth-v2](#)]. The following error codes are defined for the token revocation endpoint:

`unsupported_token_type` The authorization server does not support the revocation of the presented token type. I.e. the client tried to revoke an access token on a server not supporting this feature.

`invalid_token` The presented token is invalid.

### [2.1.](#) JSONP Support

The revocation endpoint MAY support JSONP [[jsonp](#)] by allowing GET requests with an additional parameter:

`callback` The qualified name of a JavaScript function.

Example request:

```
https://example.com/revoke?token=45ghiukldjahdnhzdauz&
callback=package.myCallback
```

Successful response:

```
package.myCallback();
```

Error response:

```
package.myCallback({"error":"invalid_token"});
```

## [3.](#) Acknowledgements

We would like to thank Sebastian Ebling, Christian Stuebner, Brian Campbell, Igor Faynberg, Lukas Rosenstock, and Justin P. Richer for their valuable feedback.

## [4.](#) IANA Considerations

This draft includes no request to IANA.

## [5.](#) Security Considerations

All relevant security considerations have been given in the functional specification.





## **6. References**

### **6.1. Normative References**

- [I-D.ietf-oauth-v2]  
Hammer-Lahav, E., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Framework", [draft-ietf-oauth-v2-26](#) (work in progress), May 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", [RFC 2246](#), January 1999.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.

### **6.2. Informative References**

- [jsonp] Ippolito, B., "Remote JSON - JSONP", December 2005.

#### Authors' Addresses

Torsten Lodderstedt (editor)  
Deutsche Telekom AG

Email: [torsten@lodderstedt.net](mailto:torsten@lodderstedt.net)

Stefanie Dronia

Email: [sdronia@gmx.de](mailto:sdronia@gmx.de)

Marius Scurtescu  
Google

Email: [mscurtescu@google.com](mailto:mscurtescu@google.com)

