

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 4, 2015

E. Wahlstroem  
Nexus Technology  
October 1, 2014

OAuth 2.0 Introspection over the Constrained Application Protocol (CoAP)

[draft-wahlstroem-ace-oauth-introspection-00.txt](#)

## Abstract

This document defines a method for a client or resource server to query an OAuth authorization server to determine meta- information about an OAuth token using the Constrained Application Protocol (CoAP) [4]. An client in possession of a OAuth2 token can use it to get metadata about the token like validity and approved scopes.

## 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 April 4, 2015.

## Copyright Notice

Copyright (c) 2014 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="#">2</a>
<a href="#">2.</a>	Terminology . . . . .	<a href="#">2</a>
<a href="#">3.</a>	Introspection CoAP Endpoint . . . . .	<a href="#">3</a>
<a href="#">4.</a>	Introspection Request . . . . .	<a href="#">3</a>
<a href="#">5.</a>	Introspection Response . . . . .	<a href="#">4</a>
<a href="#">6.</a>	Example . . . . .	<a href="#">4</a>
<a href="#">7.</a>	Security Considerations . . . . .	<a href="#">5</a>
<a href="#">8.</a>	IANA Considerations . . . . .	<a href="#">5</a>
<a href="#">9.</a>	Acknowledgements . . . . .	<a href="#">5</a>
<a href="#">10.</a>	References . . . . .	<a href="#">5</a>
<a href="#">10.1.</a>	Normative References . . . . .	<a href="#">5</a>
<a href="#">10.2.</a>	Informative References . . . . .	<a href="#">6</a>
	Author's Address . . . . .	<a href="#">7</a>

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

OAuth2 enables clients to access protected resources by obtaining an access token, which is defined in "The OAuth 2.0 Authorization Framework" [[2](#)] as "a string representing an access authorization issued to the client", rather than using the resource owner's credentials directly.

Tokens are issued to clients by an authorization server and the client uses the access token to access the protected resources hosted by the resource server. This document defines a way for a holder of this token, mostly Clients and Resource Servers, to get metadata like validity and scopes for the token. The OAuth Token Introspection specification [[14](#)] defines a way to validate the token using HTTP POST or HTTP GET. This document reuses the work done in the OAuth Token Introspection and defines a mapping of the request and response to CoAP [[4](#)] to be used by constrained devices.

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

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 "Key words for use in RFCs to Indicate Requirement Levels" [[1](#)].

This document also re-uses terminology from [RFC 6749](#) [[2](#)] and [RFC 6750](#) [[6](#)].



### **3. Introspection CoAP Endpoint**

The Introspection CoAP Endpoint responds to CoAP Confirmable requests from token holders, particularly Clients and Resource Servers. The endpoint takes a single parameter representing the token (and optionally further authentication) and returns a JSON [7] document representing the meta information surrounding the token. The endpoint MUST be protected by DTLS [5] or equivalent.

### **4. Introspection Request**

The token holder makes a request to the Introspection CoAP Endpoint by adding the following parameters using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 in the CoAP request entity-body:

In order to prevent man-in-the-middle attacks, the client MUST require the use of DTLS with server authentication for any request sent to the authorization and token endpoints. If certificate-based server authentication is used then the client MUST validate the TLS certificate of the authorization server, as defined by [RFC6125](#).

The Endpoint SHOULD also require some form of authentication to access this endpoint, such as the Client Authentication as described in OAuth 2.0 [[RFC6749](#)] or equivalent.

t: REQUIRED. The string value of the token.

rid: OPTIONAL. A service-specific string identifying the resource that the client doing the introspection is asking about.

hint: OPTIONAL. A hint about the type of the token submitted for introspection. Clients MAY pass this parameter in order to help the authorization server to optimize the token lookup. If the server is unable to locate the token using the given hint, it MUST extend its search across all of its supported token types. An authorization server MAY ignore this parameter, particularly if it is able to detect the token type automatically. Values for this field are defined in OAuth Token Revocation [[RFC7009](#)].

cid: OPTIONAL. Client id if client password defined in [section 2.3.1 in RFC 6750](#) [6] is used.

csid: OPTIONAL. Client secret if client password defined in [section 2.3.1 in RFC 6750](#) [6] is used.



Question: Should we keep, the longer, original attribute names to make it easier for CoAP-HTTP proxies and Authorization Server implementations?

## **5. Introspection Response**

If the introspection request is valid and authorized, the authorization server responds with a CoAP message using Content response code with the response encoded as a JSON structure in the payload of the message. If the request failed client authentication or is invalid, the authorization server returns an error response using the CoAP 4.00 'Bad Request' response code with the error messages defined in [Section 5.2 of RFC 6749](#) [2].

The JSON structure in the payload response includes the top-level members defined in [Section 2.2](#) in the OAuth Token Introspection specification [14]. It's recommended to only return the active attribute considering the lossy and constrained nature of CoAP client and server network and capacity.

Note that the HTTP "Cache-Control" parameters MAY be used in the CoAP response message.

## **6. Example**

For example, the client makes a CoAP request carrying the introspection request protected with DTLS to the authorization server. It then receives a response containing metadata about the token.

In the example below content-type 51 corresponds to the 'application/x-www-form-urlencoded'.



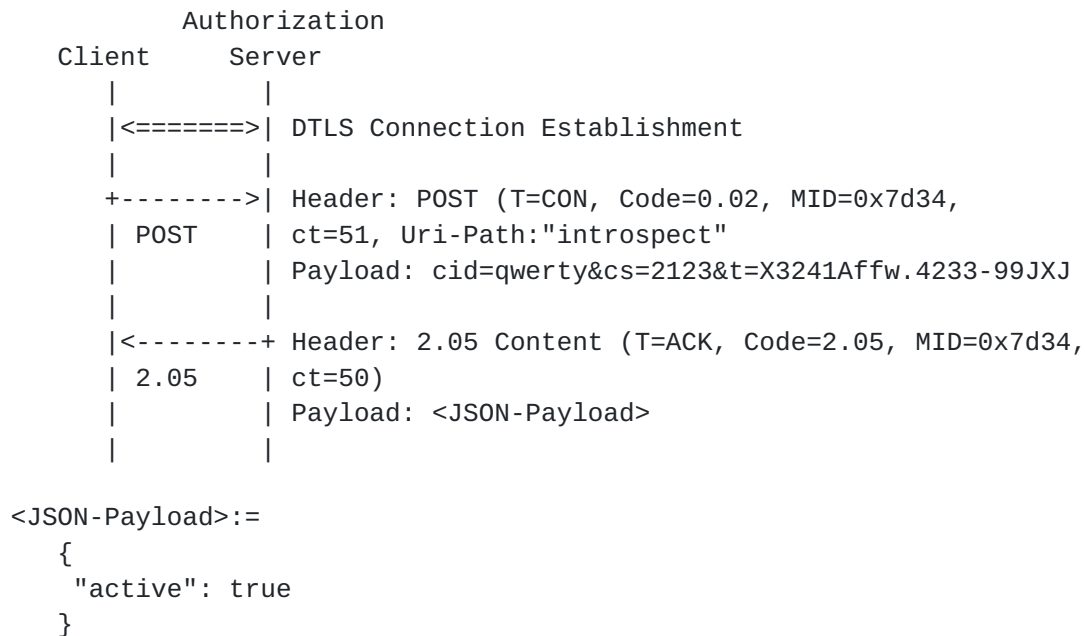


Figure 1: Example CoAP Introspection Exchange.

## 7. Security Considerations

TBD

## 8. IANA Considerations

TBD

## 9. Acknowledgements

The author would like to thank Justin Richer for his work on [14] richer-oauth-introspection and Hannes Tschofenigs [10] and [11]. This document is heavily inspired from those document's and it borrows a lot of texts from there work. The author would also like to thank Samuel Erdtman for valuable input.

## 10. References

### 10.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [2] Hardt, D., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), October 2012.





- [3]        Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", [RFC 6125](#), March 2011.
- [4]        Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", [RFC 7252](#), June 2014.
- [5]        Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), January 2012.
- [6]        Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", [RFC 6750](#), October 2012.
- [7]        Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", [RFC 4627](#), July 2006.
- [8]        Tschofenig, H., "A Datagram Transport Layer Security (DTLS) 1.2 Profile for the Internet of Things", [draft-ietf-dice-profile-04](#) (work in progress), August 2014.
- [9]        Tschofenig, H., "OAuth 2.0: Audience Information", [draft-tschofenig-oauth-audience-00](#) (work in progress), February 2013.

## **10.2.   Informative References**

- [10]        Tschofenig, H., "The OAuth 2.0 Bearer Token Usage over the Constrained Application Protocol (CoAP)", [draft-tschofenig-ace-oauth-bt-00](#) (work in progress), July 2014.
- [11]        Tschofenig, H., "The OAuth 2.0 Internet of Things (IoT) Client Credentials Grant", [draft-tschofenig-ace-oauth-iot-00](#) (work in progress), July 2014.
- [12]        Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", [draft-ietf-oauth-json-web-token-27](#) (work in progress), September 2014.
- [13]        Hunt, P., Richer, J., Mills, W., Mishra, P., and H. Tschofenig, "OAuth 2.0 Proof-of-Possession (PoP) Security Architecture", [draft-hunt-oauth-pop-architecture-02](#) (work in progress), June 2014.
- [14]        Richer, J., "OAuth Token Introspection", [draft-richer-oauth-introspection-06](#) (work in progress), July 2014.



- [15]        Bormann, C., "An Authorization Information Format (AIF) for ACE", [draft-bormann-core-ace-aif-01](#) (work in progress), July 2014.
  
- [16]        Seitz, L., Gerdes, S., Selander, G., Mani, M., and S. Kumar, "ACE use cases", [draft-seitz-ace-usecases-01](#) (work in progress), July 2014.

Author's Address

Erik Wahlstroem  
Nexus Technology  
Sweden

Email: [erik.wahlstrom@nexusgroup.com](mailto:erik.wahlstrom@nexusgroup.com)  
URI:    <https://www.nexusgroup.com>

