OAuth 2.0 Token Introspection
draft-ietf-oauth-introspection-11

Abstract

   This specification defines a method for a protected resource to query
   an OAuth 2.0 authorization server to determine the active state of an
   OAuth 2.0 token and to determine meta-information about this token.
   OAuth 2.0 deployments can use this method to convey information about
   the authorization context of the token from the authorization server
   to the protected resource.

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 January 4, 2016.

Table of Contents

1. Introduction

   In OAuth 2.0, the contents of tokens are opaque to clients. This
   means that the client does not need to know anything about the
   content or structure of the token itself, if there is any. However,
   there is still a large amount of metadata that may be attached to a
   token, such as its current validity, approved scopes, and information
   about the context in which the token was issued. These pieces of
   information are often vital to protected resources making
   authorization decisions based on the tokens being presented. Since
   OAuth 2.0 [RFC6749] does not define a protocol for the resource
   server to learn meta-information about a token that is has received
   from an authorization server, several different approaches have been
   developed to bridge this gap. These include using structured token
   formats such as JWT [RFC7519] or proprietary inter-service
   communication mechanisms (such as shared databases and protected
   enterprise service buses) that convey token information.

This specification defines a protocol that allows authorized
protected resources to query the authorization server to determine
the set of metadata for a given token that was presented to them by
an OAuth 2.0 client. This metadata includes whether or not the token
is currently active (or if it has expired or otherwise been revoked),
what rights of access the token carries (usually conveyed through
OAuth 2.0 scopes), and the authorization context in which the token
was granted (including who authorized the token and which client it
was issued to). Token introspection allows a protected resource to
query this information regardless of whether or not it is carried in
the token itself, allowing this method to be used along with or
independently of structured token values. Additionally, a protected
resource can use the mechanism described in this specification to
introspect the token in a particular authorization decision context
and ascertain the relevant metadata about the token to make this
authorization decision appropriately.

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 [RFC2119].

Unless otherwise noted, all the protocol parameter names and values
are case sensitive.

1.2. Terminology

This section defines the terminology used by this specification.
This section is a normative portion of this specification, imposing
requirements upon implementations.

This specification uses the terms "access token", "authorization
endpoint", "authorization grant", "authorization server", "client",
"client identifier", "protected resource", "refresh token", "resource
owner", "resource server", and "token endpoint" defined by OAuth 2.0
[RFC6749], and the terms "claim names" and "claim values" defined by
JSON Web Token (JWT) [RFC7519].

This specification defines the following terms:

Token Introspection
   The act of inquiring about the current state of an OAuth 2.0 token
   through use of the network protocol defined in this document.

Introspection Endpoint
   The OAuth 2.0 endpoint through which the token introspection
   operation is accomplished..

2. Introspection Endpoint

The introspection endpoint is an OAuth 2.0 endpoint that takes a
parameter representing an OAuth 2.0 token and returns a JSON
[RFC7159] document representing the meta information surrounding the
token, including whether this token is currently active. The
definition of an active token is dependent upon the authorization
server, but this is commonly a token that has been issued by this
authorization server, is not expired, has not been revoked, and valid
for use at the protected resource making the introspection call.

The introspection endpoint MUST be protected by a transport-layer
security mechanism as described in Section 4. The means by which the
protected resource discovers the location of the introspection
endpoint are outside the scope of this specification.

2.1. Introspection Request

The protected resource calls the introspection endpoint using an HTTP
POST [RFC7231] request with parameters sent as "application/x-www-
form-urlencoded" data as defined in [W3C.REC-html5-20141028]. The
protected resource sends a parameter representing the token along
with optional parameters representing additional context that is
known by the protected resource to aid the authorization server in
its response.

token REQUIRED. The string value of the token. For access tokens,
   this is the "access_token" value returned from the token endpoint
   defined in OAuth 2.0 [RFC6749] section 5.1. For refresh tokens,
   this is the "refresh_token" value returned from the token endpoint
   as defined in OAuth 2.0 [RFC6749] section 5.1. Other token types
   are outside the scope of this specification.

token_type_hint OPTIONAL. A hint about the type of the token
   submitted for introspection. The protected resource MAY pass this
   parameter 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 the OAuth Token Type Hints
   registry defined in OAuth Token Revocation [RFC7009].

The introspection endpoint MAY accept other OPTIONAL parameters to
provide further context to the query. For instance, an authorization
server may desire to know the IP address of the client accessing the
protected resource to determine if the correct client is likely to be
presenting the token. The definition of this or any other parameters

are outside the scope of this specification, to be defined by service
documentation or extensions to this specification. If the
authorization server is unable to determine the state of the token
without additional information, it SHOULD return an introspection
response indicating the token is not active as described in
Section 2.2.

To prevent token scanning attacks, the endpoint MUST also require
some form of authorization to access this endpoint, such as client
authentication as described in OAuth 2.0 [RFC6749] or a separate
OAuth 2.0 access token such as the bearer token described in OAuth
2.0 Bearer Token Usage [RFC6750]. The methods of managing and
validating these authentication credentials are out of scope of this
specification.

For example, the following example shows a protected resource calling
the token introspection endpoint to query about an OAuth 2.0 bearer
token. The protected resource is using a separate OAuth 2.0 bearer
token to authorize this call.

Following is a non-normative example request:

```
POST /introspect HTTP/1.1
Host: server.example.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Bearer 23410913-abewfq.123483

token=2YotnFZFEjr1zCsicMWpAA
```

In this example, the protected resource uses a client identifier and
client secret to authenticate itself to the introspection endpoint as
well as send a token type hint.

Following is a non-normative example request:

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

token=mF_9.B5f-4.1JqM&token_type_hint=access_token
```

2.2. Introspection Response

   The server responds with a JSON object [RFC7159] in "application/
   json" format with the following top-level members.

   active
      REQUIRED. Boolean indicator of whether or not the presented token
      is currently active. The specifics of a token's "active" state
      will vary depending on the implementation of the authorization
      server, and the information it keeps about its tokens, but a
      "true" value return for the "active" property will generally
      indicate that a given token has been issued by this authorization
      server, has not been revoked by the resource owner, and is within
      its given time window of validity (e.g. after its issuance time
      and before its expiration time). See Section 4 for information on
      implementation of such checks.

   scope
      OPTIONAL. A JSON string containing a space-separated list of
      scopes associated with this token, in the format described in
      section 3.3 of OAuth 2.0 [RFC6749].

   client_id
      OPTIONAL. Client identifier for the OAuth 2.0 client that
      requested this token.

   username
      OPTIONAL. Human-readable identifier for the resource owner who
      authorized this token.

   token_type
      OPTIONAL. Type of the token as defined in section 5.1 of OAuth
      2.0 [RFC6749].

   exp
      OPTIONAL. Integer timestamp, measured in the number of seconds
      since January 1 1970 UTC, indicating when this token will expire,
      as defined in JWT [RFC7519].

   iat
      OPTIONAL. Integer timestamp, measured in the number of seconds
      since January 1 1970 UTC, indicating when this token was
      originally issued, as defined in JWT [RFC7519].

   nbf
      OPTIONAL. Integer timestamp, measured in the number of seconds
      since January 1 1970 UTC, indicating when this token is not to be
      used before, as defined in JWT [RFC7519].

sub
     OPTIONAL. Subject of the token, as defined in JWT [RFC7519].
     Usually a machine-readable identifier of the resource owner who
     authorized this token.

aud
     OPTIONAL. Service-specific string identifier or list of string
     identifiers representing the intended audience for this token, as
     defined in JWT [RFC7519].

iss
     OPTIONAL. String representing the issuer of this token, as
     defined in JWT [RFC7519].

jti
     OPTIONAL. String identifier for the token, as defined in JWT
     [RFC7519].

Specific implementations MAY extend this structure with their own
service-specific response names as top-level members of this JSON
object. Response names intended to be used across domains MUST be
registered in the OAuth Token Introspection Response registry defined
in Section 3.1.

The authorization server MAY respond differently to different
protected resources making the same request. For instance, an
authorization server MAY limit which scopes from a given token are
returned for each protected resource to prevent protected resources
from learning more about the larger network than is necessary for its
operation.

The response MAY be cached by the protected resource to improve
performance and reduce load on the introspection endpoint, but at the
cost of liveness of the information used by the protected resource.
See Section 4 for more information regarding the trade off when the
response is cached.

For example, the following response contains a set of information
about an active token:

Following is a non-normative example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "active": true,
 "client_id": "l238j323ds-23ij4",
 "username": "jdoe",
 "scope": "read write dolphin",
 "sub": "Z5O3upPC88QrAjx00dis",
 "aud": "https://protected.example.net/resource",
 "iss": "https://server.example.com/",
 "exp": 1419356238,
 "iat": 1419350238,
 "extension_field": "twenty-seven"
}
```

If the introspection call is properly authorized but the token is not
active, does not exist on this server, or the protected resource is
not allowed to introspect this particular token, the authorization
server MUST return an introspection response with the active field
set to false. Note that to avoid disclosing too much of the
authorization server's state to a third party, the authorization
server SHOULD NOT include any additional information about an
inactive token, including why the token is inactive. For example,
the response for a token that has been revoked or is otherwise
invalid would look like the following:

Following is a non-normative example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "active": false
}
```

2.3. Error Response

If the protected resource uses OAuth 2.0 client credentials to
authenticate to the introspection endpoint and its credentials are
invalid, the authorization server responds with an HTTP 401
(Unauthorized) as described in section 5.2 of OAuth 2.0 [RFC6749].

If the protected resource uses an OAuth 2.0 bearer token to authorize
its call to the introspection endpoint and the token used for
authorization does not contain sufficient privileges or is otherwise
invalid for this request, the authorization server responds with an
HTTP 401 code as described in section 3 of OAuth 2.0 Bearer Token
Usage [RFC6750].

Note that a properly formed and authorized query for an inactive or
otherwise invalid token (or a token the protected resource is not
allowed to know about) is not considered an error response by this
specification. In these cases, the authorization server MUST instead
respond with an introspection response with the "active" field set to
"false" as described in Section 2.2.

3. IANA Considerations

3.1. OAuth Token Introspection Response Registry

This specification establishes the OAuth Token Introspection Response
registry.

OAuth registration client metadata names and descriptions are
registered with a Specification Required ([RFC5226]) after a two-week
review period on the oauth-ext-review@ietf.org mailing list, on the
advice of one or more Designated Experts. However, to allow for the
allocation of names prior to publication, the Designated Expert(s)
may approve registration once they are satisfied that such a
specification will be published.

Registration requests sent to the mailing list for review should use
an appropriate subject (e.g., "Request to register OAuth Token
Introspection Response name: example").

Within the review period, the Designated Expert(s) will either
approve or deny the registration request, communicating this decision
to the review list and IANA. Denials should include an explanation
and, if applicable, suggestions as to how to make the request
successful.

IANA must only accept registry updates from the Designated Expert(s)
and should direct all requests for registration to the review mailing
list.

3.1.1. Registration Template

Name:
    The name requested (e.g., "example"). This name is case
    sensitive. Names that match other registered names in a case

      insensitive manner SHOULD NOT be accepted. Names that match
      claims registered in the JSON Web Token Claims registry
      established by [RFC7519] SHOULD have comparable definitions and
      semantics.

   Description:
      Brief description of the metadata value (e.g., "Example
      description").

   Change controller:
      For Standards Track RFCs, state "IESG". For others, give the name
      of the responsible party. Other details (e.g., postal address,
      email address, home page URI) may also be included.

   Specification document(s):
      Reference to the document(s) that specify the token endpoint
      authorization method, preferably including a URI that can be used
      to retrieve a copy of the document(s). An indication of the
      relevant sections may also be included but is not required.

3.1.2. Initial Registry Contents

   The initial contents of the OAuth Token Introspection Response
   registry are:

   o Name: "active"
   o Description: Token active status
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "username"
   o Description: User identifier of the resource owner
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "client_id"
   o Description: Client identifier of the client
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "scope"
   o Description: Authorized scopes of the token
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "token_type"
   o Description: Type of the token
   o Change Controller: IESG

   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "exp"
   o Description: Expiration timestamp of the token
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "iat"
   o Description: Issuance timestamp of the token
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "nbf"
   o Description: Timestamp which the token is not valid before
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "sub"
   o Description: Subject of the token
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "aud"
   o Description: Audience of the token
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "iss"
   o Description: Issuer of the token
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

   o Name: "jti"
   o Description: Unique identifier of the token
   o Change Controller: IESG
   o Specification Document(s): Section 2.2 of [[this document]].

4. Security Considerations

   Since there are many different and valid ways to implement an OAuth
   2.0 system, there are consequently many ways for an authorization
   server to determine whether or not a token is currently "active" or
   not. However, since resource servers using token introspection rely
   on the authorization server to determine the state of a token, the
   authorization server MUST perform all applicable checks against a
   token's state. For instance:

o If the token can expire, the authorization server MUST determine
   whether or not the token has expired.
o If the token can be issued before it is able to be used, the
   authorization server MUST determine whether or not a token's valid
   period has started yet.
o If the token can be revoked after it was issued, the authorization
   server MUST determine whether or not such a revocation has taken
   place.
o If the token has been signed, the authorization server MUST
   validate the signature.
o If the token can be used only at certain resource servers, the
   authorization server MUST determine whether or not the token can
   be used at the resource server making the introspection call.

If an authorization server fails to perform any applicable check, the
resource server could make an erroneous security decision based on
that response. Note that not all of these checks will be applicable
to all OAuth 2.0 deployments and it is up to the authorization server
to determine which of these checks (and any other checks) apply.

If left unprotected and un-throttled, the introspection endpoint
could present a means for an attacker to poll a series of possible
token values, fishing for a valid token. To prevent this, the
authorization server MUST require authentication of protected
resources that need to access the introspection endpoint and SHOULD
require protected resources to be specifically authorized to call the
introspection endpoint. The specifics of this authentication
credentials are out of scope of this specification, but commonly
these credentials could take the form of any valid client
authentication mechanism used with the token endpoint, an OAuth 2.0
access token, or other HTTP authorization or authentication
mechanism. A single piece of software acting as both a client and a
protected resource MAY re-use the same credentials between the token
endpoint and the introspection endpoint, though doing so potentially
conflates the activities of the client and protected resource
portions of the software and the authorization server MAY require
separate credentials for each mode.

Since the introspection endpoint takes in OAuth 2.0 tokens as
parameters and responds with information used to make authorization
decisions, the server MUST support TLS 1.2 RFC 5246 [RFC5246] and MAY
support additional transport-layer mechanisms meeting its security
requirements. When using TLS, the client or protected resource MUST
perform a TLS/SSL server certificate check, as specified in RFC 6125
[RFC6125]. Implementation security considerations can be found in
Recommendations for Secure Use of TLS and DTLS [TLS.BCP].

To prevent the values of access tokens from leaking into server-side
logs via query parameters, an authorization server offering token
introspection MAY disallow the use of HTTP GET on the introspection
endpoint and instead require the HTTP POST method to be used at the
introspection endpoint.

To avoid disclosing internal server state, an introspection response
for an inactive token SHOULD NOT contain any additional claims beyond
the required "active" claim (with its value set to "false").

Since a protected resource MAY cache the response of the
introspection endpoint, designers of an OAuth 2.0 system using this
protocol MUST consider the performance and security trade-offs
inherent in caching security information such as this. A less
aggressive cache with a short timeout will provide the protected
resource with more up to date information (due to it needing to query
the introspection endpoint more often) at the cost of increased
network traffic and load on the introspection endpoint. A more
aggressive cache with a longer duration will minimize network traffic
and load on the introspection endpoint, but at the risk of stale
information about the token. For example, the token may be revoked
while the protected resource is relying on the value of the cached
response to make authorization decisions. This creates a window
during which a revoked token could be used at the protected resource.
Consequently, an acceptable cache validity duration needs to be
carefully considered given the concerns and sensitivities of the
protected resource being accessed and the likelihood of a token being
revoked or invalidated in the interim period. Highly sensitive
environments can opt to disable caching entirely on the protected
resource to eliminate the risk of stale cached information entirely,
again at the cost of increased network traffic and server load. If
the response contains the "exp" parameter (expiration), the response
MUST NOT be cached beyond the time indicated therein.

An authorization server offering token introspection must be able to
understand the token values being presented to it during this call.
The exact means by which this happens is an implementation detail and
outside the scope of this specification. For unstructured tokens,
this could take the form of a simple server-side database query
against a data store containing the context information for the
token. For structured tokens, this could take the form of the server
parsing the token, validating its signature or other protection
mechanisms, and returning the information contained in the token back
to the protected resource (allowing the protected resource to be
unaware of the token's contents, much like the client). Note that
for tokens carrying encrypted information that is needed during the
introspection process, the authorization server must be able to
decrypt and validate the token to access this information. Also note

that in cases where the authorization server stores no information
about the token and has no means of accessing information about the
token by parsing the token itself, it can not likely offer an
introspection service.

5. Privacy Considerations

The introspection response may contain privacy-sensitive information
such as user identifiers for resource owners. When this is the case,
measures MUST be taken to prevent disclosure of this information to
unintended parties. One method is to transmit user identifiers as
opaque service-specific strings, potentially returning different
identifiers to each protected resource.

If the protected resource sends additional information about the
client's request to the authorization server (such as the client's IP
address) using an extension of this specification, such information
could have additional privacy considerations that the extension
should detail. However, the nature and implications of such
extensions are outside the scope of this specification.

Omitting privacy-sensitive information from an introspection response
is the simplest way of minimizing privacy issues.

6. Acknowledgements

Thanks to the OAuth Working Group and the User Managed Access Working
Group for feedback and review of this document, and to the various
implementors of both the client and server components of this
specification. In particular, the author would like to thank Amanda
Anganes, John Bradley, Thomas Broyer, Brian Campbell, George
Fletcher, Paul Freemantle, Thomas Hardjono, Eve Maler, Josh Mandel,
Steve Moore, Mike Schwartz, Prabath Siriwardena, Sarah Squire, and
Hannes Tschofennig.

7. References

7.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 5226,
              May 2008.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6125]   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.

[RFC6749]   Hardt, D., "The OAuth 2.0 Authorization Framework", RFC
            6749, October 2012.

[RFC6750]   Jones, M. and D. Hardt, "The OAuth 2.0 Authorization
            Framework: Bearer Token Usage", RFC 6750, October 2012.

[RFC7009]   Lodderstedt, T., Dronia, S., and M. Scurtescu, "OAuth 2.0
            Token Revocation", RFC 7009, August 2013.

[RFC7159]   Bray, T., "The JavaScript Object Notation (JSON) Data
            Interchange Format", RFC 7159, March 2014.

[RFC7231]   Fielding, R. and J. Reschke, "Hypertext Transfer Protocol
            (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7519]   Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token
            (JWT)", RFC 7519, May 2015.

[W3C.REC-html5-20141028]
            Hickson, I., Berjon, R., Faulkner, S., Leithead, T.,
            Navara, E., O'Connor, E., and S. Pfeiffer, "HTML5",
            World Wide Web Consortium Recommendation REC-
            html5-20141028, October 2014,
            <http://www.w3.org/TR/2014/REC-html5-20141028>.

7.2. Informative References

[TLS.BCP]   Sheffer, Y., Holz, R., and P. Saint-Andre,
            "Recommendations for Secure Use of TLS and DTLS", November
            2014.

Appendix A. Use with Proof of Posession Tokens

With bearer tokens such as those defined by OAuth 2.0 Bearer Token
Usage [RFC6750], the protected resource will have in its possession
the entire secret portion of the token for submission to the
introspection service. However, for proof-of-possession style
tokens, the protected resource will have only a token identifier used
during the request, along with the cryptographic signature on the
request. The protected resource would be able to submit the token
identifier to the authorization server's token endpoint to obtain the
necessary key information needed to validate the signature on the

request. The details of this usage are outside the scope of this
specification and will be defined in an extension to this
specification.

Appendix B. Document History

[[To be removed by the RFC Editor.]]

-11

o Minor wording tweaks from IESG review.

-10

o Added missing 2119 section to terminology.
o Removed optional HTTP GET at introspection endpoint.
o Added terminology.
o Renamed this "a protocol" instead of "web API".
o Moved JWT to normative reference.
o Reworded definition of "scope" value.
o Clarified extensibility of input parameters.
o Noted that discover is out of scope.
o Fixed several typos and imprecise references.

-09

o Updated JOSE, JWT, and OAuth Assertion draft references to final
   RFC numbers.

-08

o Added privacy considerations note about extensions.
o Added acknowledgements (finally).

-07

o Created a separate IANA registry for introspection responses,
   importing the values from JWT.

-06

o Clarified relationship between AS and RS in introduction.
o Used updated TLS text imported from Dyn-Reg drafts.
o Clarified definition of active state.
o Added some advice on caching responses.
o Added security considerations on active state implementation.
o Changed user_id to username based on WG feedback.

-05

o Typo fix.
o Updated author information.
o Removed extraneous "linewrap" note from examples.

- 04

o Removed "resource_id" from request.
o Added examples.

- 03

o Updated HTML and HTTP references.
o Call for registration of parameters in the JWT registry.

- 02

o Removed SAML pointer.
o Clarified what an "active" token could be.
o Explicitly declare introspection request as x-www-form-urlencoded
   format.
o Added extended example.
o Made protected resource authentication a MUST.

- 01

o Fixed casing and consistent term usage.
o Incorporated working group comments.
o Clarified that authorization servers need to be able to understand
   the token if they're to introspect it.
o Various editorial cleanups.

- 00

o Created initial IETF drafted based on draft-richer-oauth-
   introspection-06 with no normative changes.

Author's Address

   Justin Richer (editor)

   Email: ietf@justin.richer.org