|  |
| --- |
| **[TOC](" \l "toc)** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Draft | B. de Medeiros, Ed. | |  | M. Scurtescu | |  | Google | |  | P. Tarjan | |  | Facebook | |  | M. Jones | |  | Microsoft | |  | October 17, 2013 | |

# OAuth 2.0 Multiple Response Type Encoding Practices - draft 10

### Abstract

This specification aims to provide guidance on proper encoding of responses to OAuth 2.0 Authorization Requests, where the request specifies a response type that includes space characters.

This specification also serves as the registration document for several specific new response types, in accordance with the stipulations of the OAuth Parameters Registry.

### Table of Contents

[**1.**](#Introduction)  Introduction  
    [**1.1.**](#rnc)  Requirements Notation and Conventions  
    [**1.2.**](#Terminology)  Terminology  
**[2.](" \l "ResponseTypesAndEncodings)**  Response Types and Response Encodings  
    **[2.1.](" \l "ResponseEncodings)**  Response Encodings  
**[3.](" \l "MultiValueResponseTypes)**  Multiple-Valued Response Types  
**[4.](" \l "id_token)**  ID Token Response Type  
**[5.](" \l "none)**  None Response Type  
**[6.](" \l "Combinations)**  Registration of Some Multiple-Valued Response Type Combinations  
**[7.](" \l "IANA)**  IANA Considerations  
    **[7.1.](" \l "OAuthResponseTypesReg)**  OAuth Authorization Endpoint Response Types Registration  
        **[7.1.1.](" \l "RegistryContents)**  Registry Contents  
    **[7.2.](" \l "OAuthParametersRegistry)**  OAuth Parameters Registration  
        **[7.2.1.](" \l "ParametersContents)**  Registry Contents  
**[8.](" \l "Security)**  Security Considerations  
**[9.](" \l "rfc.references1)**  Normative References  
**[Appendix A.](" \l "POSTResponseExample)**  POST Response Encoding Example  
**[Appendix B.](" \l "Acknowledgements)**  Acknowledgements  
**[Appendix C.](" \l "Notices)**  Notices  
**[Appendix D.](" \l "History)**  Document History  
[**§**](#rfc.authors)  Authors' Addresses

|  |
| --- |
| [**TOC**](#toc) |

### 1.  Introduction

|  |
| --- |
| [**TOC**](#toc) |

### 1.1.  Requirements Notation and 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] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.)](#RFC2119) .

Throughout this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes MUST NOT be used as part of the value.

|  |
| --- |
| [**TOC**](#toc) |

### 1.2.  Terminology

This specification uses the terms "Access Token", "Refresh Token", "Authorization Code", "Authorization Grant", "Authorization Server", "Authorization Endpoint", "Client", "Client Identifier", "Client Secret", "Protected Resource", "Resource Owner", "Resource Server", and "Token Endpoint" defined by [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749]. This specification also defines the following terms:

Client and Server

In the traditional client-server authentication model, the client requests an access restricted resource (Protected Resource) on the server by authenticating with the server using the Resource Owner's credentials.

Response Type

The Client informs the Authorization Server of the desired authorization processing flow using the response\_type request parameter. This determines what parameters are returned from the endpoints used.

Response Encoding

The Client informs the Authorization Server of how parameters are to be returned from the Authorization Endpoint. Non-default encodings are specified using the response\_encoding request parameter. If response\_encoding is not present in a request, the default Response Encoding mechanism specified by the Response Type is used.

Authorization Endpoint Response Type Registry

Process established by the OAuth 2.0 specification for the registration of new response\_type parameters.

Multiple-Valued Response Types

The OAuth 2.0 specification allows for registration of space-separated response\_type values. If a response type contains one of more space characters (%20), it is compared as a space-delimited list of values in which the order of values does not matter.

|  |
| --- |
| [**TOC**](#toc) |

### 2.  Response Types and Response Encodings

The Response Type request parameter response\_type informs the Authorization Server of the desired authorization processing flow, including what parameters are returned from the endpoints used. The Response Encoding request parameter response\_encoding informs the Authorization Server of the mechanism to be used when returning parameters from the Authorization Endpoint. Each Response Type value also defines a default Authorization Encoding mechanism to be used, if no Authorization Encoding is specified using the request parameter.

|  |
| --- |
| **[TOC](" \l "toc)** |

### 2.1.  Response Encodings

This specification defines the following OAuth Authorization Request parameter:

response\_encoding

OPTIONAL. Informs the Authorization Server of the mechanism to be used when returning parameters from the Authorization Endpoint. This use of this parameter is NOT RECOMMENDED when the Response Encoding that would be requested is the default encoding specified by the Response Type.

This specification uses the following Response Encodings and defines the associated response\_encoding values:

In this encoding, response parameters are encoded in the query string added to the redirect\_uri when redirecting back to the Client.

In this encoding, response parameters are encoded in the fragment added to the redirect\_uri when redirecting back to the Client.

POST

In this encoding, response parameters are encoded as HTML form values that are auto-submitted in the user-agent, and thus are transmitted via the HTTP POST method to the client. The action attribute of the form MUST be the client's redirect URI. The method of the form attribute MUST be POST.

Any technique supported by the user agent MAY be used to cause the submission of the form, and any form content necessary to support this MAY be included, such as submit controls and client-side scripting commands. However, the client MUST be able to process the message without regard for the mechanism by which the form submission is initiated.

For purposes of this specification, the default Response Encoding for the OAuth 2.0 code response\_type is the query encoding. For purposes of this specification, the default Response Encoding for the OAuth 2.0 token response\_type is the fragment encoding.

Note that it is expected that additional Response Encodings may be defined by other specifications in the future, including possibly postMessage and CORS.

|  |
| --- |
|  |

### 3.  Multiple-Valued Response Types

When a multiple-valued response type is defined, it is RECOMMENDED that the following encoding rules be applied for the issued response from the Authorization Endpoint.

The all parameters returned from the Authorization Endpoint SHOULD use the same Response Encoding. This recommendation applies to both success and error responses.

Rationale: This significantly simplifies Client parameter processing. It also can have positive performance benefits, as described below.

For instance, if a response includes fragment encoded parts, a User-Agent Client component must be involved to complete processing of the response. If a new query parameter is added to the Client URI, it will cause the User-Agent to re-fetch the Client URI, causing discontinuity of operation of the User-Agent based Client components. If only fragment encoding is used, the User-Agent will simply reactivate the Client component, which can then process the fragment and also convey any parameters to a Client host as necessary, e.g., via XmlHttpRequest. Therefore, full fragment encoding always results in lower latency for response processing.

|  |
| --- |
| **[TOC](" \l "toc)** |

4

|  |
| --- |
|  |

### .  ID Token Response Type

This section registers a new response type, the id\_token, in accordance with the stipulations in the OAuth 2.0 specification, Section 8.4. The intended purpose of the id\_token is that it MUST provide an assertion of the identity of the Resource Owner as understood by the server. The assertion MUST specify a targeted audience, e.g. the requesting Client. However, the specific semantics of the assertion and how it can be validated are not specified in this document.

id\_token

When supplied as the response\_type parameter in an OAuth 2.0 Authorization Request, a successful response MUST include the parameter id\_token. The Authorization Server SHOULD NOT return an OAuth 2.0 Authorization Code, Access Token, or Access Token Type in a successful response to the grant request. If a redirect\_uri is supplied, the User-Agent SHOULD be redirected there after granting or denying access. The request MAY include a state parameter, and if so, the server MUST echo its value by adding it to the redirect\_uri when issuing either a successful response or an error response. The default Response Encoding for this Response Type is the fragment encoding and the query encoding MUST NOT be used. This applies to both successful responses and error responses.

Returning the id\_token in a fragment reduces the likelihood that the id\_token leaks during transport and mitigates the associated risks to the privacy of the user (Resource Owner).

|  |
| --- |
| [**TOC**](#toc) |

### 5.  None Response Type

This section registers the response type none, in accordance with the stipulations in the OAuth 2.0 specification, Section 8.4. The intended purpose is to enable use cases where a party requests the server to register a grant of access to a Protected Resource on behalf of a Client but requires no access credentials to be returned to the Client at that time. The means by which the Client eventually obtains the access credentials is left unspecified here.

One scenario is where a user wishes to purchase an application from a market, and desires to authorize application installation and grant the application access to Protected Resources in a single step. However, since the user is not presently interacting with the (not yet active) application, it is not appropriate to return access credentials simultaneously in the authorization step.

none

When supplied as the response\_type parameter in an OAuth 2.0 Authorization Request, the Authorization Server SHOULD NOT return an OAuth 2.0 Authorization Code, Access Token, Access Token Type, or ID Token in a successful response to the grant request. If a redirect\_uri is supplied, the User-Agent SHOULD be redirected there after granting or denying access. The request MAY include a state parameter, and if so, the server MUST echo its value by adding it to the redirect\_uri when issuing either a successful response or an error response. The default Response Encoding for this Response Type is the query encoding. This applies to both successful responses and error responses.

The response type none SHOULD NOT be combined with other response types.

|  |
| --- |
| [**TOC**](#toc) |

### 6.  Registration of Some Multiple-Valued Response Type Combinations

This section registers combinations of the values code, token, and id\_token, which are each individually registered response types.

code token

When supplied as the value for the response\_type parameter, a successful response MUST include an Access Token, an Access Token Type, and an Authorization Code. The default Response Encoding for this Response Type is the fragment encoding and the query encoding MUST NOT be used. This applies to both successful responses and error responses.

code id\_token

When supplied as the value for the response\_type parameter, a successful response MUST include both an Authorization Code and an id\_token. The default Response Encoding for this Response Type is the fragment encoding and the query encoding MUST NOT be used. This applies to both successful responses and error responses.

id\_token token

When supplied as the value for the response\_type parameter, a successful response MUST include an Access Token, an Access Token Type, and an id\_token. The default Response Encoding for this Response Type is the fragment encoding and the query encoding MUST NOT be used. This applies to both successful responses and error responses.

code id\_token token

When supplied as the value for the response\_type parameter, a successful response MUST include an Authorization Code, an id\_token, an Access Token, and an Access Token Type. The default Response Encoding for this Response Type is the fragment encoding and the query encoding MUST NOT be used. This applies to both successful responses and error responses.

For all these Response Types, the request MAY include a state parameter, and if so, the server MUST echo its value by adding it to the redirect\_uri when issuing either a successful response or an error response.

A non-normative request/response example as issued/received by the User-Agent (with extra line breaks for display purposes only) is:

GET /authorize?

response\_type=id\_token%20token

&client\_id=s6BhdRkqt3

&redirect\_uri=https%3A%2F%2Fclient.example.org%2Fcb

&state=af0ifjsldkj HTTP/1.1

Host: server.example.com

HTTP/1.1 302 Found

Location: https://client.example.org/cb#

access\_token=SlAV32hkKG

&token\_type=bearer

&id\_token=eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso

&expires\_in=3600

&state=af0ifjsldkj

|  |
| --- |
| [**TOC**](#toc) |

### 7.  IANA Considerations

|  |
| --- |
| **[TOC](" \l "toc)** |

### 7.1.  OAuth Authorization Endpoint Response Types Registration

This specification registers the response\_type values defined by this specification in the IANA OAuth Authorization Endpoint Response Types registry [[RFC6749] (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749).

|  |
| --- |
| [**TOC**](#toc) |

### 7.1.1.  Registry Contents

* Response Type Name: id\_token
* Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
* Specification Document(s): http://openid.net/specs/oauth-v2-multiple-response-types-1\_0.html
* Response Type Name: none
* Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
* Specification Document(s): http://openid.net/specs/oauth-v2-multiple-response-types-1\_0.html
* Response Type Name: code token
* Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
* Specification Document(s): http://openid.net/specs/oauth-v2-multiple-response-types-1\_0.html
* Response Type Name: code id\_token
* Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
* Specification Document(s): http://openid.net/specs/oauth-v2-multiple-response-types-1\_0.html
* Response Type Name: id\_token token
* Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
* Specification Document(s): http://openid.net/specs/oauth-v2-multiple-response-types-1\_0.html
* Response Type Name: code id\_token token
* Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
* Specification Document(s): http://openid.net/specs/oauth-v2-multiple-response-types-1\_0.html

|  |
| --- |
| [**TOC**](#toc) |

### 7.2.  OAuth Parameters Registration

This specification registers the following parameter in the IANA OAuth Parameters registry defined in [RFC 6749 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](" \l "RFC6749) [RFC6749].

|  |
| --- |
| **[TOC](" \l "toc)** |

### 7.2.1.  Registry Contents

* Parameter name: response\_encoding
* Parameter usage location: Authorization Request
* Change controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
* Specification document(s): [Section 2.1 (Response Encodings)](" \l "ResponseEncodings) of this document
* Related information: None

|  |
| --- |
| **[TOC](" \l "toc)** |

### 8.  Security Considerations

There are security implications to encoding response values in the query string.In no case should a set of response parameters whose default Response Encoding is the fragment encoding be encoded using the query encoding. However, it is safe to encode response parameters whose default Response Encoding using the POST encoding.

|  |
| --- |
| **[TOC](" \l "toc)** |

### 9. Normative References

|  |  |
| --- | --- |
| **[RFC2119]** | [Bradner, S.](mailto:sob@harvard.edu), “[Key words for use in RFCs to Indicate Requirement Levels](http://tools.ietf.org/html/rfc2119),” BCP 14, RFC 2119, March 1997 ([TXT](http://www.rfc-editor.org/rfc/rfc2119.txt), [HTML](http://xml.resource.org/public/rfc/html/rfc2119.html), [XML](http://xml.resource.org/public/rfc/xml/rfc2119.xml)). |
| **[RFC6749]** | Hardt, D., “[The OAuth 2.0 Authorization Framework](http://tools.ietf.org/html/rfc6749),” RFC 6749, October 2012 ([TXT](http://www.rfc-editor.org/rfc/rfc6749.txt)). |

|  |
| --- |
| [**TOC**](#toc) |

### Appendix A.  POST Response Encoding Example

Below is a non-normative request/response/request example as issued/received/issued by the User-Agent (with extra line breaks for display purposes only) demonstrating an auto-submitted POST encoded response:

Authorization Request to the Authorization Endpoint:

GET /authorize?

response\_type=id\_token

&response\_encoding=POST

&client\_id=some\_client

&scope=openid

&redirect\_uri=https%3A%2F%2Fclient.example.org%2Fcallback

&state=DcP7csa3hMlvybERqcieLHrRzKBra

&nonce=2T1AgaeRTGTMAJyeDMN9IJbgiUG HTTP/1.1

Host: server.example.com

After authentication and approval by the End-User, the Authorization Server issues the Authorization Response:

HTTP/1.1 200 OK

Content-Type: text/html;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

<html>

<head><title>Submit This Form</title></head>

<body onload="javascript:document.forms[0].submit()">

<form method="post" action="https://client.example.org/callback">

<input type="hidden" name="state"

value="DcP7csa3hMlvybERqcieLHrRzKBra"/>

<input type="hidden" name="id\_token"

value="eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJzdWIiOiJqb2huIiw

iYXVkIjoiZmZzMiIsImp0aSI6ImhwQUI3RDBNbEo0c2YzVFR2cllxUkIiLC

Jpc3MiOiJodHRwczpcL1wvbG9jYWxob3N0OjkwMzEiLCJpYXQiOjEzNjM5M

DMxMTMsImV4cCI6MTM2MzkwMzcxMywibm9uY2UiOiIyVDFBZ2FlUlRHVE1B

SnllRE1OOUlKYmdpVUciLCJhY3IiOiJ1cm46b2FzaXM6bmFtZXM6dGM6U0F

NTDoyLjA6YWM6Y2xhc3NlczpQYXNzd29yZCIsImF1dGhfdGltZSI6MTM2Mz

kwMDg5NH0.c9emvFayy-YJnO0kxUNQqeAoYu7sjlyulRSNrru1ySZs2qwqq

wwq-Qk7LFd3iGYeUWrfjZkmyXeKKs\_OtZ2tI2QQqJpcfrpAuiNuEHII-\_fk

IufbGNT\_rfHUcY3tGGKxcvZO9uvgKgX9Vs1v04UaCOUfxRjSVlumE6fWGcq

XVEKhtPadj1elk3r4zkoNt9vjUQt9NGdm1OvaZ2ONprCErBbXf1eJb4NW\_h

nrQ5IKXuNsQ1g9ccT5DMtZSwgDFwsHMDWMPFGax5Lw6ogjwJ4AQDrhzNCFc

0uVAwBBb772-86HpAkGWAKOK-wTC6ErRTcESRdNRe0iKb47XRXaoz5acA"/>

</form>

</body>

</html>

Which results in an HTTP POST to the client:

POST /callback HTTP/1.1

Host: client.example.org

Content-Type: application/x-www-form-urlencoded

id\_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IjEifQ.eyJzdWIiOiJqb2huIiwiYX

VkIjoiZmZzMiIsImp0aSI6ImhwQUI3RDBNbEo0c2YzVFR2cllxUkIiLCJpc

3MiOiJodHRwczpcL1wvbG9jYWxob3N0OjkwMzEiLCJpYXQiOjEzNjM5MDMx

MTMsImV4cCI6MTM2MzkwMzcxMywibm9uY2UiOiIyVDFBZ2FlUlRHVE1BSnl

lRE1OOUlKYmdpVUciLCJhY3IiOiJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTD

oyLjA6YWM6Y2xhc3NlczpQYXNzd29yZCIsImF1dGhfdGltZSI6MTM2MzkwM

Dg5NH0.c9emvFayy-YJnO0kxUNQqeAoYu7sjlyulRSNrru1ySZs2qwqqwwq

-Qk7LFd3iGYeUWrfjZkmyXeKKs\_OtZ2tI2QQqJpcfrpAuiNuEHII-\_fkIuf

bGNT\_rfHUcY3tGGKxcvZO9uvgKgX9Vs1v04UaCOUfxRjSVlumE6fWGcqXVE

KhtPadj1elk3r4zkoNt9vjUQt9NGdm1OvaZ2ONprCErBbXf1eJb4NW\_hnrQ

5IKXuNsQ1g9ccT5DMtZSwgDFwsHMDWMPFGax5Lw6ogjwJ4AQDrhzNCFc0uV

AwBBb772-86HpAkGWAKOK-wTC6ErRTcESRdNRe0iKb47XRXaoz5acA&

state=DcP7csa3hMlvybERqcieLHrRzKBra

|  |
| --- |
| **[TOC](" \l "toc)** |

### Appendix B.  Acknowledgements

The OpenID Community would like to thank the following people for the work they've done in the drafting and editing of this specification.

Naveen Agarwal (naa@google.com), Google

John Bradley (ve7jtb@ve7jtb.com), Ping Identity

Brian Campbell (bcampbell@pingidentity.com), Ping Identity

Michael B. Jones (mbj@microsoft.com), Microsoft

Breno de Medeiros (breno@google.com), Google

Nat Sakimura (n-sakimura@nri.co.jp), Nomura Research Institute, Ltd.

David Recordon (dr@fb.com), Facebook

Marius Scurtescu (mscurtescu@google.com), Google

Paul Tarjan (pt@fb.com), Facebook

|  |
| --- |
| [**TOC**](#toc) |

### Appendix C.  Notices

Copyright (c) 2013 The OpenID Foundation.

The OpenID Foundation (OIDF) grants to any Contributor, developer, implementer, or other interested party a non-exclusive, royalty free, worldwide copyright license to reproduce, prepare derivative works from, distribute, perform and display, this Implementers Draft or Final Specification solely for the purposes of (i) developing specifications, and (ii) implementing Implementers Drafts and Final Specifications based on such documents, provided that attribution be made to the OIDF as the source of the material, but that such attribution does not indicate an endorsement by the OIDF.

The technology described in this specification was made available from contributions from various sources, including members of the OpenID Foundation and others. Although the OpenID Foundation has taken steps to help ensure that the technology is available for distribution, it takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this specification or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any independent effort to identify any such rights. The OpenID Foundation and the contributors to this specification make no (and hereby expressly disclaim any) warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to this specification, and the entire risk as to implementing this specification is assumed by the implementer. The OpenID Intellectual Property Rights policy requires contributors to offer a patent promise not to assert certain patent claims against other contributors and against implementers. The OpenID Foundation invites any interested party to bring to its attention any copyrights, patents, patent applications, or other proprietary rights that may cover technology that may be required to practice this specification.

|  |
| --- |
| [**TOC**](#toc) |

### Appendix D.  Document History

[[ To be removed from the final specification ]]

-10

* Differentiated between Response Types and Response Encodings. Defined the response\_encoding request parameter for specifying non-default Response Encodings.
* Defined the POST Response Encoding.
* Added Security Considerations section.

-09

* Clarified that the response types none and id\_token do not return Authorization Code, Access Token, or Access Token Type values.
* Clarified that an Access Token Type value MUST be returned in the same cases that an Access Token is.
* Clarified that the state value must be included in both successful responses and error responses, when present in the request.
* Fixed #887 - Clarified the reasons that some parameters are described as "SHOULD be fragment encoded" and stated that these "MUST NOT be query encoded".

-08

* Corrected RFC 2119 keyword usage.

-07

* Fixed #817 - Removed duplicate definition of "Authorization Endpoint".
* Fixed #818 - Corrected HTTP GET example.

-06

* Added registry contents to IANA Considerations section
* Referenced OAuth 2.0 RFC -- RFC 6749

-05

* Changed client.example.com to client.example.org, per issue #251

-04

* Updated Notices

-03

* Use same section number structure as the OpenID Connect specs

-02

* Editorial corrections

-01

* Initial draft

|  |
| --- |
| [**TOC**](#toc) |

### Authors' Addresses

|  |  |
| --- | --- |
|  | Breno de Medeiros (editor) |
|  | Google |
| **Email:** | [breno@google.com](mailto:breno@google.com) |
| **URI:** | <http://stackoverflow.com/users/311376/breno> |
|  |  |
|  | Marius Scurtescu |
|  | Google |
| **Email:** | [mscurtescu@google.com](mailto:mscurtescu@google.com) |
| **URI:** | <https://twitter.com/mscurtescu> |
|  |  |
|  | Paul Tarjan |
|  | Facebook |
| **Email:** | [pt@fb.com](mailto:pt@fb.com) |
|  |  |
|  | Michael B. Jones |
|  | Microsoft |
| **Email:** | [mbj@microsoft.com](mailto:mbj@microsoft.com) |
| **URI:** | <http://self-issued.info/> |