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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Draft | N. Sakimura, Ed. | |  | NRI | |  | D. Recordon | |  | Facebook | |  | J. Bradley | |  | Protiviti | |  | B. de Medeiros | |  | Google | |  | M. Jones | |  | Microsoft | |  | E. Jay | |  | MGI1 | |  | October 14, 2011 | |

# OpenID Connect Messages 1.0 - draft 06

### Abstract

OpenID Connect is an identity framework that provides authentication, authorization, and attribute transmission capability. It allows third party attested claims from distributed sources. The specification suite builds on OAuth 2.0 and consists of Building Blocks (Messages, Discovery, Dynamic Client Registration, Session Management, JSON Web Token, JSON Web Signature, JSON WEB Encryption, JSON Web Keys, Simple Web Discovery), Protocol Bindings (e.g., Standard and Basic Client) and Extensions. This specification covers the core "Messages" of the suite that defines the messages used and abstract flow which will be further constrained or extended in the companion specifications in the suite.

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

### Table of Contents

[**1.**](#terminology)  Terminology  
[**2.**](#anchor1)  Overview  
[**3.**](#anchor2)  Messages  
    [**3.1.**](#anchor3)  Authorization Endpoint  
        [**3.1.1.**](#id_token)  ID Token  
        [**3.1.2.**](#auth_req)  Authorization Request  
        [**3.1.3.**](#anchor6)  Authorization Response  
        [**3.1.4.**](#anchor7)  Authorization Error Response  
    [**3.2.**](#token_ep)  Token Endpoint  
        [**3.2.1.**](#access_token_request)  Access Token Request  
        [**3.2.2.**](#access_token_response)  Access Token Response  
        [**3.2.3.**](#anchor9)  Token Error Response  
    [**3.3.**](#userinfo_ep)  UserInfo Endpoint  
        [**3.3.1.**](#anchor11)  Requests  
        [**3.3.2.**](#anchor12)  Responses  
        [**3.3.3.**](#anchor13)  Errors  
        [**3.3.4.**](#anchor14)  Claim Types  
    [**3.4.**](#check_id_ep)  Check ID Endpoint  
        [**3.4.1.**](#anchor17)  Check ID Request  
        [**3.4.2.**](#anchor18)  Check ID Response  
        [**3.4.3.**](#anchor19)  Error Codes  
    [**3.5.**](#anchor20)  Session Management Endpoints  
[**4.**](#Serializations)  Serializations  
    [**4.1.**](#qss)  Query String Serialization  
    [**4.2.**](#js)  JSON Serialization  
[**5.**](#sigenc)  Signatures and Encryption  
    [**5.1.**](#sigs)  Signing  
    [**5.2.**](#enc)  Encryption  
[**6.**](#anchor21)  Verification  
    [**6.1.**](#anchor22)  Authorization Request Verification  
    [**6.2.**](#anchor23)  Authorization Response Verification  
    [**6.3.**](#anchor24)  Token Request Verification  
    [**6.4.**](#anchor25)  Token Response Verification  
    [**6.5.**](#anchor26)  UserInfo Request Verification  
    [**6.6.**](#anchor27)  UserInfo Response Verification  
    [**6.7.**](#anchor28)  Check ID Request Verification  
    [**6.8.**](#anchor29)  Check ID Response Verification  
[**7.**](#related)  Related Specifications  
[**8.**](#security_considerations)  Security Considerations  
    [**8.1.**](#assertion_manufacture)  Assertion Manufacture/Modification  
    [**8.2.**](#assertion_disclosure)  Assertion Disclosure  
    [**8.3.**](#assertion_repudiation)  Assertion Repudiation  
    [**8.4.**](#assertion_redirect)  Assertion Redirect  
    [**8.5.**](#assertion_reuse)  Assertion Reuse  
    [**8.6.**](#artifact_manufacture)  Secondary Authenticator Manufacture  
    [**8.7.**](#artifact_capture)  Secondary Authenticator Capture  
    [**8.8.**](#assertion_substitution)  Assertion Substitution  
    [**8.9.**](#auth_req_disclosure)  Authentication Request Disclosure  
    [**8.10.**](#anchor30)  Timing Attack  
    [**8.11.**](#authn_proc_threats)  Authentication Process Threats  
[**9.**](#iana)  IANA Considerations  
    [**9.1.**](#oauth_params)  OAuth Parameters Registry  
        [**9.1.1.**](#anchor31)  Scope Parameters  
        [**9.1.2.**](#anchor32)  Authorization Request Parameter (display)  
        [**9.1.3.**](#anchor33)  Authorization Request Parameter (prompt)  
        [**9.1.4.**](#anchor34)  Authorization Request Parameter (nonce)  
        [**9.1.5.**](#anchor35)  Authorization Request Parameter (audience)  
        [**9.1.6.**](#anchor36)  Authorization Request Parameter (request)  
        [**9.1.7.**](#anchor37)  Authorization Request Parameter (request\_uri)  
        [**9.1.8.**](#anchor38)  ID Token Response Parameters  
    [**9.2.**](#anchor39)  OAuth Extensions Error Registry  
        [**9.2.1.**](#anchor40)  Authorization endpoint error (invalid\_request\_redirect\_uri)  
        [**9.2.2.**](#anchor41)  Authorization endpoint error (login\_required)  
        [**9.2.3.**](#anchor42)  Authorization endpoint error (session\_selection\_required)  
        [**9.2.4.**](#anchor43)  Authorization endpoint error (approval\_required)  
        [**9.2.5.**](#anchor44)  Authorization endpoint error (user\_mismatched)  
        [**9.2.6.**](#anchor45)  Token endpoint error (invalid\_authorization\_code)  
        [**9.2.7.**](#anchor46)  UserInfo endpoint error (invalid\_schema)  
        [**9.2.8.**](#anchor47)  Check ID endpoint error (invalid\_id\_token)  
[**10.**](#rfc.references1)  References  
    [**10.1.**](#rfc.references1)  Normative References  
    [**10.2.**](#rfc.references2)  Informative References  
[**Appendix A.**](#anchor50)  Acknowledgements  
[**Appendix B.**](#anchor51)  Document History  
[**§**](#rfc.authors)  Authors' Addresses

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

### 1.  Terminology

In addition to 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" that are defined by [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0], this specification defines the following terms:

Assertion

A set of Claims that are attested to by an Issuer.

Authentication

An act of verifying End-User's identity through the verification of the credential.

Claim

A piece of information about an Entity that a Claims Provider asserts about that Entity.

Claims Provider

An Authorization Server that can return claims about a user.

End-User

A human resource owner.

Entity

Something that has separate and distinct existence and that can be identified in context.

ID Token

A token that contains claims about the authentication event.

Issuer

An Entity who issues an Assertion.

Issuer Identifier

A verifiable identifier of an Issuer. An issuer Identifier is a HTTPS URL with no path component.

Message

A request or a response between an OpenID Relying Party and an OpenID Provider.

OpenID Provider (OP)

A service capable of providing identity information to a Relying Party.

OP Endpoints

End-User Authentication Endpoint, Authorization Endpoint, and Token Endpoint.

OpenID Request Object

A JSON object that holds the request variables. It holds OpenID request variables. It MAY also contain other OAuth parameters for the request signing purpose, in which case the parameter values MUST match with the OAuth request parameters.

Relying Party (RP)

An application requiring identity information from an OpenID Provider.

RP Endpoint

The endpoint to which the OP responses are returned through redirect.

Check ID Endpoint

A resource that, when presented with an ID Token by the client, returns authentication information about the user session represented by that ID Token.

UserInfo Endpoint

A protected resource that, when presented with an access token by the client, returns authorized information about the user represented by that access token.

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

### 2.  Overview

The OpenID Connect protocol, in abstract, follows the following steps.

1. The Client sends a request to the Authorization Server's end-user authorization endpoint.
2. The Authorization Server authenticates the user and obtains appropriate authorization.
3. The Authorization Server responds with access\_token, id\_token, and a few other variables.
4. The Client sends a request with the access\_token to the [UserInfo endpoint (UserInfo Endpoint)](#userinfo_ep).
5. UserInfo endpoint returns the additional user information supported by the Resource Server.
6. OPTIONAL. The Client sends a request with the id\_token to the Authorization Server's [Check ID endpoint (Check ID Endpoint)](#check_id_ep).
7. OPTIONAL. The Check ID endpoint responds with authentication information pertaining to the supplied id\_token.

This specification only defines the abstract message flow and message formats. The actual use MUST be based on one of the companion protocol bindings specifications such as [OpenID Connect Basic Client (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Basic Client 1.0,” October 2011.)](#OpenID.Basic) [OpenID.Basic] or [OpenID Connect Standard (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Standard 1.0,” October 2011.)](#OpenID.Standard) [OpenID.Standard].

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

### 3.  Messages

In OpenID Connect protocols, in abstract, the process proceeds by the client interacting with endpoints. There are a number of endpoints involved.

1. Authorization Endpoint: The Client sends a request to the Authorization Server at the authorization endpoint. The Authorization Server then authenticates the End-User to find out if he is eligible to make the authorization. Then, upon the authorization action of the End-User, the Authorization Server returns an Authorization Response that includes Authorization Code, code. For some Clients, Implicit Grant may be used to obtain access\_token without using code. In this case, response\_type MUST be set to token.
2. Token Endpoint: The Client sends the access token request to the token endpoint to obtain Access Token Response which includes an access\_token.
3. UserInfo Endpoint: The access\_token MAY be sent to the UserInfo endpoint to obtain claims about the user.
4. Check ID Endpoint: An id\_token MAY be sent to the Check ID endpoint to obtain information about the authentication event.
5. Session Management Endpoints: The ID Token MAY be sent to these endpoints to manage the session.

Both Request and Response may either be serialized into [Query String Serialization (Query String Serialization)](#qss) or [JSON (Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” July 2006.)](#RFC4627) [RFC4627].

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

### 3.1.  Authorization Endpoint

The client sends an Authorization Request to the authorization endpoint to obtain an Authorization Response and an [ID Token (ID Token)](#id_token).

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

### 3.1.1.  ID Token

The ID Token is a token that contains claims pertinent to the authentication event. The Token is a [JWT (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token,” July 2011.)](#JWT) [JWT] which contains JSON claims.

The ID Token is used to manage the authentication event and user identifier and is scoped to a particular client via the audience and nonce claims. It MUST NOT be used as an access token to access OAuth 2.0 protected resources.

The ID Token MUST attest minimally to the following claims:

iss

REQUIRED. The unique identifier of the issuer of the response.

user\_id

REQUIRED. A locally unique and never reassigned identifier for the user, which is intended to be consumed by the Client. e.g. 24400320 or AItOawmwtWwcT0k51BayewNvutrJUqsvl6qs7A4. It MUST NOT exceed 255 ASCII characters in length.

aud

REQUIRED. This member identifies the audience that this ID Token is intended for. It is RECOMENDED that aud be the OAuth client\_id of the RP.

exp

REQUIRED. Type Integer. Identifies the expiration time on or after which the ID Token MUST NOT be accepted for processing. The processing of this parameter requires that the current date/time MUST be before the expiration date/time listed in the value. Implementers MAY provide for some small leeway, usually no more than a few minutes, to account for clock skew. The value is number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the desired date/time. See [RFC 3339 (Klyne, G., Ed. and C. Newman, “Date and Time on the Internet: Timestamps,” July 2002.)](#RFC3339) [RFC3339] for details regarding date/times in general and UTC in particular.

iso29115

OPTIONAL. (entity authentication assurance): Specifies the X.eaa / [ISO/IEC29115 (McCallister, E., “ITU-T Recommendation X.eaa | ISO/IEC 2nd CD 29115 -- Information technology - Security techniques - Entity authentication assurance framework,” .)](#ISO29115) [ISO29115] entity authentication assurance level of the authentication performed.

nonce

OPTIONAL. If the authorization request includes a nonce request value, then this value is REQUIRED and its value is set to the same value as the request value.

JWT ID Tokens MAY be signed or signed and encrypted via [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS] and [JWE (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Encryption,” July 2011.)](#JWE) [JWE] respectively, thereby providing authentication, integrity, non-repudiation and/or confidentiality. ID Tokens in other formats MAY be signed or encrypted with methods suitable for the respective formats.

Clients SHOULD verify and decipher signed or encrypted ID Tokens independently. Clients that do not understand the ID Token format or do not wish to process ID Tokens MAY treat ID Tokens as opaque values and submit them to the [Check ID Endpoint (Check ID Endpoint)](#check_id_ep) for verification and decoding.

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

### 3.1.2.  Authorization Request

Section 4.1.1 and 4.2.1 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0] defines the OAuth Authorization Request parameters. In this specification, the values to the parameters are defined as follows.

response\_type

A space delimited, case sensitive list of string values. Acceptable values include code, token, and id\_token. The value MUST include code for requesting an Authorization Code, token for requesting an Access Token, and id\_token for requesting an ID Token.

scope

A space delimited, case sensitive list of string values. The values specify an additive list of claims that are returned by the UserInfo endpoint. The following values are defined:

openid

REQUIRED. Informs the Authorization Server that the client is making an OpenID request. If the openid scope is not specified, the server SHOULD treat the request as a generic OAuth 2.0 request, and perform no OpenID Connect processing. The openid value also requests that the ID Token associated with the authentication session be returned. If the response\_type includes token, the ID Token is returned in the Authorization Response along with the Access Token. If the response\_type includes code, the ID Token is returned as part of the Token endpoint response.

profile

OPTIONAL. This requests that access to the user's [profile claims (Reserved Member Definitions)](#ClaimTable) excluding the address and email claims at the UserInfo endpoint be granted by the issued Access Token.

address

OPTIONAL. This requests that access to address claim at the UserInfo endpoint be granted by the issued Access Token.

email

OPTIONAL. This requests that access to the email claim at the UserInfo endpoint be granted by the issued Access Token.

Other required OAuth 2.0 parameters in the request include:

client\_id

The client identifier.

redirect\_uri

A redirection URI where the response will be sent.

The following extension parameters are also defined:

nonce

REQUIRED. A random, unique string value used to mitigate replay attacks.

display

OPTIONAL. A string value that specifies how the authorization server displays the authentication page to the user.

none

The authorization server MUST NOT display any authentication or confirmation user interface pages. An error is returned if either the user is not already authenticated or the client does not have pre-configured approval for the requested scopes. This can be used as a method to check for existing authentication and/or approval.

prompt

OPTIONAL. A space delimited, case sensitive list of string values that specifies how the authorization server prompts the user for reauthentication and reapproval. The possible values are:

login

The authorization server MUST prompt the user for reauthentication.

consent

The authorization server MUST prompt the user for reapproval before returning information to the client.

select\_account

The authorization server MUST prompt the user to select a user account if the account has multiple accounts associated with it

This can be used by the client to make sure that the user is still present for the current session or to bring attention to the request. If this parameter is used in conjunction with the display parameter set to "none", an error is returned.

audience

OPTIONAL. The target audience identifier for the ID Token.

request

OPTIONAL. A [JWT (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token,” July 2011.)](#JWT) [JWT] encoded [OpenID Request Object (OpenID Request Object)](#OpenIDReq).

request\_uri

OPTIONAL. An URL that points to an OpenID Request Object. This is used to pass an OpenID Request Object by reference.

The request MAY contain the following optional parameters:

state

An opaque value used to maintain state between the request and the callback.

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

### 3.1.2.1.  OpenID Request Object

The OpenID Request object is used to provide OpenID request parameters that MAY differ from the default ones. Implementing support for the OpenID Request object is OPTIONAL. Supporting it is necessary for implementations that need to request or provide sets of claims other than the default [UserInfo (UserInfo Endpoint)](#userinfo_ep), ID Token, and [Check ID (Check ID Endpoint)](#check_id_ep) claim sets.

The OpenID Request object is a [JWT (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token,” July 2011.)](#JWT) [JWT] that is passed as the value of the "request" parameter in the authorization request. The OpenID Request Object can also be sent by reference. Parameters that affect the information returned from the UserInfo endpoint are in the "userinfo" member. Parameters that affect the information returned in the ID Token are in the "id\_token" member. If the same parameters are available both as query strings and in the OpenID Request Object, the later takes the precedence.

The OpenID Request Object MUST contain all REQUIRED OAuth 2.0 authorization request parameters and MAY contain optional and extension parameters.

The OpenID Request object MAY contain a set of members defined by this specification and MAY contain other members that are not defined by this specification. OpenID Request object members SHOULD be understood by both parties.

The JWT MAY be signed or unsigned. When it is unsigned, it will be indicated by the JWT "signed":"none" convention in the JWT header. If signed, the OpenID Request object SHOULD contain the standard JWT "iss" and "aud" claims.

The members defined by this specification are:

userinfo

OPTIONAL. (UserInfo request): Requests affecting the values to be returned from the UserInfo endpoint. If not present, the UserInfo endpoint behaves in the default manner.

id\_token

OPTIONAL. (ID request): Requests affecting the values to be to be returned in the ID Token. If not present, the default ID Token contents are used. If present, these parameters are used to request additional claims incremental to the [default claims of the ID Token. (ID Token)](#id_token)

An example an OpenID request object before JWT encoding is as follows:

{

"response\_type": "code id\_token",

"client\_id": "s6BhdRkqt3",

"redirect\_uri": "https://client.example.com/cb",

"scope": "openid profile",

"state": "af0ifjsldkj", "userinfo":

{

"claims":

{

"name": null,

"nickname": {"optional": true},

"email": null,

"verified": null,

"picture": {"optional": true},

}

"format": "signed"

}

"id\_token":

{

"claims":

{

"auth\_time": null

}

"max\_age": 86400,

"iso29115": "2"

}

}

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

### 3.1.2.1.1.  "userinfo" member

The structure of the "userinfo" (UserInfo request) member is a JSON object that MAY contain the following members:

claims

OPTIONAL. (Requested Claims): Set of requested claims from the UserInfo endpoint. If not present, the default UserInfo claims held by the OP are returned.

format

OPTIONAL. (Format): The requested format for the UserInfo endpoint information. If not present, the format is an unsigned JSON object.

locale

OPTIONAL. (Locale): The default languages and scripts of the entire claim request, represented as a space-separated list of [BCP47 (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.)](#RFC5646) [RFC5646] language tags.

The "claims" member is a JSON object with a member for each requested claim. The member names are the requested claim names. The member values may be either:

null

This indicates that this claim is being requested in the default manner. In particular, this is a required claim. OR

A JSON Object

This is used to provide additional information about the claim being requested.

The claims MAY be represented in multiple languages and scripts. To specify languages and scripts for the claim request, [BCP47 (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.)](#RFC5646) [RFC5646] language tags delimited by a "#" MUST be added to each requested claim name for which a particular language and script is requested. For example, the claim family\_name#ja-Kana-JP is used for expressing Family Name in Katakana in Japanese, which is commonly used to index and represent the phonetics of the Kanji representation of the same value represented as family\_name#ja-Hani-JP.

All members of the "claims" object are OPTIONAL.

The members of the JSON object value following a claim name defined by this specification are:

optional

If this is an optional claim, this member's value MUST be true, else, if present, its value MUST be false, which indicates that it is a required claim. If it is not present, it is a required claim.

Other members MAY be defined to provide additional information about the requested claim. If the "claims" member is present in the "userinfo" object, the claims requested within it override the default claim set that would otherwise be returned from the UserInfo endpoint.

The "format" member of the "userinfo" object is used to specify the requested format of the UserInfo endpoint contents. Values defined are:

unsigned

- in which case the contents are an unsigned JSON object

signed

- in which case the contents are a signed JWT

encrypted

- in which case the contents are an signed and encrypted JWT

All members of the "userinfo" object are OPTIONAL. Other members MAY be present and if so, SHOULD understood by both parties.

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

### 3.1.2.1.2.  "id\_token" member

The structure and function of the "id\_token" (ID Token request) member of the OpenID Request object is similar to that of the "userinfo" member. It MAY include "claims", "format", "locale". The structure of these members is the same as that for the "userinfo" member. If the "claims" member is present in the "id\_token" object, the claims requested within it modify the default claim set that would otherwise be returned in the ID Token. Unlike for the "userinfo" member, typically these claims will augment, rather than override the default set.

Following claim MAY be requested in the ID Token by specifying it in the "claims" member:

auth\_time

OPTIONAL. (authenticated at): Requests that the "auth\_time" claim be present. The claim value is the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time that the user authentication occurred. (The "auth\_time" claim semantically corresponds to the openid.pape.auth\_time response parameter.)

In addition to the "claims" member, this additional member is defined within the "id\_token" member of the OpenID Request object:

max\_age

OPTIONAL. (max authentication age): Specifies that the user must be actively authenticated if any present authentication is older than the specified number of seconds. (The "max\_age" request parameter corresponds to the OpenID 2.0 openid.pape.max\_auth\_age request parameter.)

iso29115

OPTIONAL. (entity authentication assurance): Specifies the X.eaa / [ISO/IEC29115 (McCallister, E., “ITU-T Recommendation X.eaa | ISO/IEC 2nd CD 29115 -- Information technology - Security techniques - Entity authentication assurance framework,” .)](#ISO29115) [ISO29115] entity authentication assurance level that is requested by the client.

It is anticipated that additional "id\_token" parameters MAY be defined to request that additional properties hold for the authentication - for instance, that certain authentication policies be applied (in the same spirit of the OpenID 2.0 openid.pape.auth\_policies values), or that the authentication conform to the policies defined by a specified trust framework. These parameters MAY be defined by extension specifications.

All members of the "id\_token" object are OPTIONAL. Other members MAY be present and if so, SHOULD understood by both parties

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

### 3.1.3.  Authorization Response

When the response\_type in the request is token, the Authorization Response MUST return the parameters defined in section 4.2.2 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0]. This specification only supports [Bearer Tokens (Jones, M., Hardt, D., and D. Recordon, “OAuth 2.0 Protocol: Bearer Tokens,” September 2011.)](#OAuth.2.0.Bearer) [OAuth.2.0.Bearer]. The OAuth 2.0 response parameter "token\_type" MUST be set to "Bearer".

When the response\_type in the request is code, the Authorization Response MUST return the parameters defined in section 4.1.2 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0].

When the response\_type includes other values, they must be returned as defined by their registration. The id\_token return type is defined in [OAuth 2.0 Multiple Response Type Encoding Practices (de Medeiros, B., Scurtescu, M., and P. Tarjan, “OAuth 2.0 Multiple Response Type Encoding Practices,” October 2011.)](#RESPONSE.TYPES) [RESPONSE.TYPES].

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

### 3.1.4.  Authorization Error Response

If the end-user denies the access request or if the request fails, the authorization server informs the client by returning parameters defined in section 4.1.2.1 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0].

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

### 3.1.4.1.  Error Codes

In addition to the error codes defined in section 4.1.2.1 and 4.2.2.1 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0], this specification defines the following additional error codes:

invalid\_request\_redirect\_uri

The redirect\_uri in the request is missing or malformed.

login\_required

The authorization server requires user authentication.

session\_selection\_required

The user is required to select a session at the authorization server. The user may be authenticated at the authorization server with different associated accounts, but the user did not select a session or no session selection is requested from the user due to the display parameter being set to none.

approval\_required

The authorization server requires user approval.

user\_mismatched

The current logged in user at the authorization server does not match the requested user.

Errors are returned as defined by the registered OAuth 2.0 response\_type.

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

### 3.2.  Token Endpoint

Access Token Request / Responses interact with a Token endpoint. Upon a successful request, it returns an Access Token and ID Token.

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

### 3.2.1.  Access Token Request

The client obtains an access token by authenticating with the authorization server and presenting its access grant (in the form of an authorization code, resource owner credentials, an assertion, or a refresh token).

Clients required to authenticate with the authorization server MUST do so according to section 3.2.1 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0]. OAuth defines an alternative method for clients to authenticate with symmetric client keys through the use of the client\_id and client\_secret parameter in the message request body. This specification extends the client authentication scheme to allow clients to authenticate with asymmetric keys. Asymmetric client authentication allows the client to authenticate with the authorization server without sending its secret key. The asymmetric client authentication parameters are the following:

client\_id

REQUIRED. The identifier of the client.

secret\_type

OPTIONAL. Specifies the client authentication type which determines how the client\_secret parameter is interpreted. It can be one of "basic" or "JWT". The defaults value is "basic". If the value is "basic", the client is performing symmetric key authentication as specified in OAuth 2.0. If the value is "JWT", the client is performing asymmetric key authentication.

client\_secret

REQUIRED. The client secret. If the secret\_type is "basic", the value is the pre-shared secret that was issued to the client during client registration. If the "secret\_type" is "JWT", the value is the [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS] signature of a JSON object containing one of the following claims:

code

REQUIRED. The authorization code that was issued by the authorization server.

refresh\_token

REQUIRED. The refresh token that was issued.

In addition to the client authentication parameters, if this is a Refresh token request, the client MUST send the additional parameters specified in Section 6 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0]. Otherwise the client MUST send the request parameter for the Access Token endpoint as specified in section 4.1.3 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0].

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

### 3.2.2.  Access Token Response

After receiving and verifying a valid and authorized Access Token Request from the client, the Authorization Server returns a Positive Assertion that includes an Access Token and an ID Token. The parameters in the successful response are defined in Section 4.1.4 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0].

This specification further constrains that only [Bearer Tokens (Jones, M., Hardt, D., and D. Recordon, “OAuth 2.0 Protocol: Bearer Tokens,” September 2011.)](#OAuth.2.0.Bearer) [OAuth.2.0.Bearer] are issued at the Token endpoint. The OAuth 2.0 response parameter "token\_type" MUST be set to "Bearer".

In addition to the OAuth 2.0 response parameters, the following parameters MUST be included in the response if the Authorization Request scope parameter contains openid:

id\_token

The ID Token value associated with the authentication session.

Following is a non-normative example:

{

"access\_token": "SlAV32hkKG",

"token\_type": "Bearer",

"refresh\_token": "8xLOxBtZp8",

"expires\_in": 3600,

"id\_token": "eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso"

}

As in the [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0], Clients SHOULD ignore unrecognized response parameters.

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

### 3.2.3.  Token Error Response

If the token request is invalid or unauthorized, the authorization server constructs the error response. The parameters of the Token Error Response are defined as in Section 5.2 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0].

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

### 3.2.3.1.  Error Codes

In addition to the error codes defined in Section 5.2 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0], this specification defines the following error codes.

invalid\_authorization\_code

The authorization code is missing, malformed, or invalid.

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

### 3.3.  UserInfo Endpoint

The UserInfo endpoint is an OAuth 2.0 protected resource that returns a claim object which contains claims about the authenticated user. Claim objects are objects that contain members and members' values which are the individual claims and claims values. A claim object is represented by a JSON object which contains a collection of name and value pairs for the claims.

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

### 3.3.1.  Requests

Clients MAY send requests with the following parameters to the UserInfo endpoint to obtain further information about the user.

access\_token

REQUIRED. The access\_token obtained from an OpenID Connect authorization request. This parameter MUST NOT be sent if the access token is sent in the HTTP Authorization header as described in Section 7.1 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0]. Access tokens sent in the authorization header must be [Bearer tokens (Jones, M., Hardt, D., and D. Recordon, “OAuth 2.0 Protocol: Bearer Tokens,” September 2011.)](#OAuth.2.0.Bearer) [OAuth.2.0.Bearer].

schema

REQUIRED. The schema in which the data is be returned. The default value is openid. If the value of this parameter is omitted, or not openid, the response may be a proprietary schema to support backwards compatibility. A URL MAY be passed to define custom schemes not specified by short names. Custom schema names and responses are out of scope for this specification.

id

This identifier is reserved for backwards compatibility. It MUST be ignored by the endpoint if the openid schema is used.

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

### 3.3.2.  Responses

If the requested schema is "openid", the response MUST return a JSON object that contains the full set or subset of claims that are defined below. Additional claims (not specified below) MAY also be returned.

The members may be represented in multiple languages and scripts. To specify the languages and scripts, [BCP47 (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.)](#RFC5646) [RFC5646] language tags MUST be added to each member names delimited by a #, e.g., familyName#ja-Kana-JP for expressing Family Name in Katakana in Japanese, which is commonly used to index and represent the phonetics of the Kanji representation of the same represented as familyName#ja-Hani-JP.

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| user\_id | string | Identifier for the user at the issuer. |
| name | string | User's full name in displayable form including all name parts, ordered according to user's locale and preferences. |
| given\_name | string | Given name or first name of the user. |
| family\_name | string | Surname or last name of the user. |
| middle\_name | string | Middle name of the user. |
| nickname | string | Casual name of the user that may or may not be the same as the given\_name. For instance, a nickname value of Mike might be returned alongside a given\_name value of Michael. |
| profile | string | URL of user's profile page. |
| picture | string | URL of the user's profile picture. |
| website | string | URL of user's web page or blog. |
| email | string | The user's preferred e-mail address. |
| verified | boolean | True if the user's e-mail address has been verified; otherwise false. |
| gender | string | The user's gender: female or male. |
| birthday | string | The user's birthday, represented as a date string in MM/DD/YYYY format. The year MAY be 0000, indicating that it is omitted. |
| zoneinfo | string | String from zoneinfo [[zoneinfo] (Public Domain, “The tz database,” June 2011.)](#zoneinfo) timezone database. For example, Europe/Paris or America/Los\_Angeles. |
| locale | string | The user's locale, represented as an [RFC 5646 (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.)](#RFC5646) [RFC5646] language tag. This is typically an [ISO 639-1 Alpha-2 (International Organization for Standardization, “ISO 639-1:2002. Codes for the representation of names of languages -- Part 1: Alpha-2 code,” 2002.)](#ISO639-1) [ISO639‑1] language code in lowercase and an [ISO 3166-1 Alpha-2 (International Organization for Standardization, “ISO 3166-1:1997. Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes,” 1997.)](#ISO3166-1) [ISO3166‑1] country code in uppercase, separated by a dash. For example, en-US or fr-CA. As a compatibility note, some implementations have used an underscore as the separator rather than a dash, for example, en\_US; Implementations MAY choose to accept this locale syntax as well. |
| phone\_number | string | The user's preferred telephone number. For example, +1 (425) 555-1212 or +56 (2) 687 2400. |
| address | JSON object | The user's preferred address. The value of the address member is a [JSON (Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” July 2006.)](#RFC4627) [RFC4627] structure containing some or all of these string-valued fields: formatted, street\_address, locality, region, postal\_code, and country. The street\_address field MAY contain multiple lines if the address representation requires it. Implementations MAY return only a subset of the fields of an address, depending upon the information available and the user's privacy preferences. For example, the country and region might be returned without returning more fine-grained address information. |
| updated\_time | string | Time the user's information was last updated, represented as a [RFC 3339 (Klyne, G., Ed. and C. Newman, “Date and Time on the Internet: Timestamps,” July 2002.)](#RFC3339) [RFC3339] datetime. For example, 2011-01-03T23:58:42+0000. |

|  |
| --- |
| **Table 1: Reserved Member Definitions** |

For privacy reasons, OpenID providers may elect to not provide values for some schema elements as part of the "openid" scope.

The UserInfo endpoint MUST return claims in JSON format unless a request for a different format is made by the client in the authorization request. The UserInfo endpoint MAY return claims in JWT format which can be signed or encrypted via [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS] and [JWE (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Encryption,” July 2011.)](#JWE) [JWE] respectively. [The OpenID Request Object (OpenID Request Object)](#OpenIDReq) describes how to request a different format. The UserInfo endpoint MUST return a content-type header to indicate which format is being returned. The following are accepted content types:

|  |  |
| --- | --- |
| **Content-Type** | **Format Returned** |
| application/json | plain text JSON object |
| application/jwt | A JWT |

The following is a non-normative normal claims responses:

{

"name": "Jane Doe"

"given\_name": "Jane",

"family\_name": "Doe",

"email": "janedoe@example.com",

"picture": "http://example.com/janedoe/me.jpg"

}

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

### 3.3.3.  Errors

In addition to the error codes defined in section 2.4.1 of [OAuth 2.0 Bearer Tokens (Jones, M., Hardt, D., and D. Recordon, “OAuth 2.0 Protocol: Bearer Tokens,” September 2011.)](#OAuth.2.0.Bearer) [OAuth.2.0.Bearer], this specification defines the following additional error codes:

invalid\_schema

The requested schema is invalid or unsupported.

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

### 3.3.4.  Claim Types

The UserInfo endpoint MAY return a claim object containing the following three types of claims:

Normal Claims

Claims that are directly asserted by the OpenID Provider.

Aggregated Claims

Claims that are asserted by a claims provider other than the OpenID Provider but are returned by OpenID Provider.

Distributed Claims

Claims that are asserted by a claims provider other than the OpenID Provider but are returned as references by the OpenID Provider.

The UserInfo endpoint MUST support normal claims.

Aggregated and distributed claims support is OPTIONAL.

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

### 3.3.4.1.  Normal Claims

Normal claims are represented as members in a claim object. The claim name is the member name and the claim value is the member value.

The following is a non-normative normal claims responses:

{

"name": "Jane Doe"

"given\_name": "Jane",

"family\_name": "Doe",

"email": "janedoe@example.com",

"picture": "http://example.com/janedoe/me.jpg"

}

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

### 3.3.4.2.  Aggregated and Distributed Claims

Aggregated and distributed claims are represented within the "\_claim\_names" and "\_claim\_sources" members of a claim object. Both "\_claim\_names" and "\_claim\_sources" members are claim objects.

\_claim\_names

This is a JSON object whose member names are the claims names for the aggregated and distributed claims. The member values are references to the member names in the "\_claim\_sources" member of the claim object from which the actual value can be retrieved.

\_claim\_sources

This is a JSON object whose member names are referenced by the member values of the "\_claim\_names" member of the claim object. The member values contain sets of aggregated claims or reference locations for distributed claims. The member values can have one of the following formats depending on whether it's providing aggregated or distributed claims:

Aggregated Claims

A JSON object which MUST contain the "JWT" member whose value is a [JWT (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token,” July 2011.)](#JWT) [JWT] which MUST contain all the claims in the "\_claim\_names" object which references the corresponding "\_claim\_sources" member. Other members MAY be present if they are understood by both parties.

JWT

REQUIRED. JWT Value

Distributed Claims

A JSON object which contains the following members and values:

endpoint

REQUIRED. The value is the OAuth 2.0 resource endpoint from which the associated claim can be retrieved. The endpoint URL MUST return the claim as a JWT.

access\_token

OPTIONAL. Access token enabling retrieval of the claims from the endpoint URL by using the [OAuth 2.0 Bearer (Jones, M., Hardt, D., and D. Recordon, “OAuth 2.0 Protocol: Bearer Tokens,” September 2011.)](#OAuth.2.0.Bearer) [OAuth.2.0.Bearer] scheme. Claims SHOULD be requested using the Authorization request header field and claims sources MUST support this method. If the access token is not available, clients MAY need to retrieve the access token out of band or use an a priori access token that was negotiated between the claim source and client, or the claim source MAY reauthenticate the user and/or reauthorize the client.

Other members MAY be present, if understood by both parties

The following is a non-normative response with aggregated claims:

Claims Provider A contains the following claims for Jane Doe :

{

"address": "1234 Hollywood Blvd., Los Angeles, CA 90210",

"phone\_number": "+1 (310) 123-4567"

}

Claims Provider A signs the JSON claims, resulting in a signed JWT:

jwt\_header.jwt\_part2.jwt\_part3

Authorization Server returns Jane Doe's aggregated claims from Claims Provider A :

{

"name": "Jane Doe",

"given\_name": "Jane",

"family\_name": "Doe",

"birthday": "01/01/2001",

"eye\_color": "blue",

"email": "janedoe@example.com",

"\_claim\_names": {

"address": "src1",

"phone\_number": "src1",

},

"\_claim\_sources": {

"src1": {"JWT": "jwt\_header.jwt\_part2.jwt\_part3"},

}

}

The following is a non-normative response with distributed claims:

Claims Provider A (Jane Doe's Bank) contains the following claims for Jane Doe :

{

"shipping\_address": "1234 Hollywood Blvd., Los Angeles, CA 90210",

"payment\_info": "Some\_Card 1234 5678 90123 4562"

"phone\_number": "+1 (310) 123-4567"

}

A Claims Provider B (Credit Agency) contains the following claims for Jane Doe :

{

"credit\_score": "650"

}

Authorization Server returns Jane Doe's claims along with the distributed claims from

Claims Provider A and B by sending the access tokens and URL locations where the claims

may be retrieved.

{

"name": "Jane Doe",

"given\_name": "Jane",

"family\_name": "Doe",

"email": "janedoe@example.com",

"birthday": "01/01/2001",

"eye\_color": "blue",

"\_claim\_names": {

"payment\_info": "src1",

"shipping\_address": "src1",

"credit\_score": "src2"

},

"\_claim\_sources": {

"src1": {"endpoint": "https://bank.example.com/claimsource"},

"src2": {"endpoint": "https://creditagency.example.com/claimshere", "access\_token": "ksj3n283dke"}

}

}

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

### 3.4.  Check ID Endpoint

The Check ID endpoint validates ID Tokens and returns a [JSON (Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” July 2006.)](#RFC4627) [RFC4627] object which contains information about the end user and authentication event associated with the supplied ID Token.

This endpoint can be used by clients that are not able to or do not wish to handle ID Tokens. In such cases, clients MAY treat the ID Token as an opaque value, and use the Check ID endpoint to retrieve and examine the claims associated with the token.

A full explanation of how to use an ID Token to perform session management can be found in the [OpenID Connect Session Management 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Session Management 1.0,” September 2011.)](#OpenID.Session) [OpenID.Session].

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

### 3.4.1.  Check ID Request

To request the information about the authentication performed on the user, the following parameters are sent to the Check ID endpoint:

id\_token

REQUIRED. The ID Token obtained from an OpenID Connect authorization request.

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

### 3.4.2.  Check ID Response

The response is a [JSON (Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” July 2006.)](#RFC4627) [RFC4627] object containing the [ID Token (ID Token)](#id_token) claims.

Other claims MAY be returned by specifying the desired ID Token claims to be returned in an [OpenID Request Object (OpenID Request Object)](#OpenIDReq) when making an authorization request. The Check ID endpoint MUST return claims in JSON format unless a request for a different format is made by the client in the authorization request. The Check ID endpoint MAY return claims in JWT format which can be [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS] signed or [JWE (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Encryption,” July 2011.)](#JWE) [JWE] encrypted. [The OpenID Request Object (OpenID Request Object)](#OpenIDReq) describes how to request a different format. The Check ID endpoint MUST return a content-type header to indicate which format is being returned. The following are accepted content types:

|  |  |
| --- | --- |
| **Content-Type** | **Format Returned** |
| application/json | plain text JSON object |
| application/jwt | A JWT |

The following is a non-normative example of a request to the Check ID endpoint:

POST /id\_token HTTP/1.1

Host: server.example.com

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

id\_token=eyJ0eXAiOiJKV1QiL

HTTP/1.1 200 OK

Content-Type: application/json

{

"iss": "http://server.example.com",

"user\_id": "248289761001",

"aud": "http://client.example.com",

"exp": 1311281970

}

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

### 3.4.3.  Error Codes

In addition to the error codes defined in Section 5.2 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” September 2011.)](#OAuth.2.0) [OAuth.2.0], this specification defines the following error codes.

invalid\_id\_token

The ID Token is not valid for the requested resource, is malformed, is in an incorrect format, or is expired.

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

### 3.5.  Session Management Endpoints

The Session Management endpoints provide endpoints to manage and synchronize authentication sessions at the authorization server and clients. The endpoints are specified in the [OpenID Connect Session Management (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Session Management 1.0,” September 2011.)](#OpenID.Session) [OpenID.Session] specification.

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

### 4.  Serializations

Each message can be serialized either in query parameter serialization or JSON serialization unless it was explicitly stated in the previous sections.

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

### 4.1.  Query String Serialization

In order to serialize the parameters using the query string serialization, the client constructs the string by adding the parameters and values to the query component using the application/x-www-form-urlencoded format as defined by [[W3C.REC‑html401‑19991224] (Raggett, D., Hors, A., and I. Jacobs, “HTML 4.01 Specification,” December 1999.)](#W3C.REC-html401-19991224).

Following is a non-normative example of such serialization:

GET /authorize?scope=openid&response\_type=code

&client\_id=s6BhdRkqt3

&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1

Host: server.example.com

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

### 4.2.  JSON Serialization

The parameters are serialized into a JSON structure by adding each parameter at the highest structure level. Parameter names and string values are included as JSON strings. Numerical values are included as JSON numbers. Each parameter may have JSON Structure as its value.

Following is a non-normative example of such serialization:

{

"access\_token":"SlAV32hkKG",

"expires\_in":3600,

"refresh\_token":"8xLOxBtZp8"

}

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

### 5.  Signatures and Encryption

Depending on the transport through which the messages are sent, the integrity of the message may not be guaranteed and the originator of the message may not be authenticated. To mitigate these risks, OpenID Connect messages MAY utilize [JSON Web Signatures (JWS) (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS] to sign the content.

To achieve message confidentiality, OpenID Connect messages MAY use [JSON Web Encryption (JWE) (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Encryption,” July 2011.)](#JWE) [JWE] to encrypt the content.

When the message is both signed and encrypted, it MUST be signed first then encrypted.

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

### 5.1.  Signing

HMAC-SHA256 Signatures

When using HMAC-SHA256 Signatures, alg claim of the JWS header MUST be set to HS256. The client\_secret MUST be used as the signature key.

RSA and ECDSA Signatures

When using RSA-SHA256 or ECDSA-SHA256 Signatures, alg MUST be set to RS256 and ES256 respectively. As the key, either x5u or x5t registered/obtained MUST be used to retrieve the relevant key. If there were multiple keys in jku, kid MUST be specified in JWS header. If there were multiple certificates in x5u, then x5t MUST be specified in JWS header. The key usage of the respective keys MUST include signature.

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

### 5.2.  Encryption

Using client\_secret

[tbd] To use client\_secret to encrypt the payload, Keywrap MUST be used.

Asymmetric Encryption

[tbd] . As the key, either x5u or x5t registered/obtained MUST be used to retrieve the relevant key. If there were multiple keys in jku, kid MUST be specified in JWS header. If there were multiple certificates in x5u, then x5t MUST be specified in JWS header. The key usage of the respective keys MUST include signature.

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

### 6.  Verification

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

### 6.1.  Authorization Request Verification

If the request is signed, the authorization server MUST validate the signature according to Section 5 of [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS].

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

### 6.2.  Authorization Response Verification

To verify the validity of the Authorization Response, the client MUST do the following:

1. If the response was signed, the Client SHOULD verify the token signature as the first step.
2. Check that current time is within the validity period contained within the response.
3. Check that the OP that responded was really the intended OP through a TLS/SSL server certificate check.

If the client does not directly verify the signature, it MUST make a request to the Check ID Endpoint to validate the token.

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

### 6.3.  Token Request Verification

If the request is signed, the authorization server MUST validate the signature according to Section 5 of [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS].

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

### 6.4.  Token Response Verification

To verify the validity of the Token response, the client MUST do the following:

1. If the response is signed, the Client SHOULD validate the signature according to Section 5 of [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS].
2. Check that current time is within the validity period contained within the response.
3. Check that the OP that responded was really the intended OP through a TLS/SSL server certificate check.

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

### 6.5.  UserInfo Request Verification

If the request is signed, the authorization server MUST validate the signature according to Section 5 of [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS].

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

### 6.6.  UserInfo Response Verification

To verify the validity of the UserInfo response, the client MUST do the following:

1. If the response was signed, the Client SHOULD validate the signature according to Section 5 of [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS].
2. Check that the OP that responded was really the intended OP through a TLS/SSL server certificate check.

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

### 6.7.  Check ID Request Verification

If the request is signed, the authorization server MUST validate the signature according to Section 5 of [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS].

The authorization server also MUST validate the request to ensure all required parameters are present and valid.

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

### 6.8.  Check ID Response Verification

To verify the validity of the Response, the client MUST do the following:

1. Check that current time is within the validity period of the exp contained within the response.
2. Verify that the response was intended for the recipient, using the aud (audience) contained within the response.
3. Verify that iss is a trusted issuer of the response.
4. If nonce is present, verify that it is the same value as the one that was sent in the authorization request.
5. Check that the server that responded was really the intended server through a TLS/SSL server certificate check.

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

### 7.  Related Specifications

These related OpenID Connect specifications MAY OPTIONALLY be used in combination with this specification to provide additional functionality:

* [OpenID Connect Discovery 1.0 (Sakimura, N., Bradley, J., Jones, M., and E. Jay, “OpenID Connect Discovery 1.0,” September 2011.)](#OpenID.Discovery) [OpenID.Discovery] - Dynamic discovery for user and authorization server endpoints and information
* [OpenID Connect Dynamic Client Registration 1.0 (Sakimura, N., Bradley, J., Ed., and M. Jones, “OpenID Connect Dynamic Client Registration 1.0,” September 2011.)](#OpenID.Registration) [OpenID.Registration] - Dynamic registration of OpenID Connect clients with OpenID Providers
* [OpenID Connect Session Management 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Session Management 1.0,” September 2011.)](#OpenID.Session) [OpenID.Session] - Session management for OpenID Connect sessions
* [OpenID Connect Standard 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Standard 1.0,” October 2011.)](#OpenID.Standard) [OpenID.Standard] - Protocol binding for the full set of OpenID Connect Messages
* [OpenID Connect Basic Client 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Basic Client 1.0,” October 2011.)](#OpenID.Basic) [OpenID.Basic] - Protocol binding for a subset of the OpenID Connect Messages which is intended for use by basic relying parties.

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

### 8.  Security Considerations

This specification references the security considerations defined in [OAuth 2.0 Security Considerations (Lodderstedt, T., Ed., McGloin, M., and P. Hunt, “OAuth 2.0 Threat Model and Security Considerations,” July 2011.)](#OAuth.2.0.SC) [OAuth.2.0.SC].

Followings are the list of attack vectors and remedies that were considered for this specification.

For details of the attack vector, see [[SP800‑63] (National Institute of Standards and Technology, “NIST SP800-63rev.1: Electronic Authentication Guideline,” .)](#SP800-63).

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

### 8.1.  Assertion Manufacture/Modification

To mitigate this attack, there are two ways to mitigate it.

1. The assertion may be digitally signed by the OP. The Relying Party SHOULD check the digital signature to verify that it was issued by a legitimate OP.
2. The assertion may be sent over a protected channel such as TLS/SSL. In order to protect the integrity of assertions from malicious attack, the OP MUST be authenticated. In this specification, the assertion is always sent over TLS/SSL protected channel.

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

### 8.2.  Assertion Disclosure

The Assertion disclosure can be mitigated in the following two ways.

1. Assertion is sent over TLS/SSL protected channel, where RP is authenticated by "client\_id" and "client\_secret".
2. Signed Assertion is encrypted by the RP's public key.

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

### 8.3.  Assertion Repudiation

To mitigate this threat, the assertion may be digitally signed by the OP using a key that supports non-repudiation. The RP SHOULD check the digital signature to verify that it was issued by a legitimate OP.

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

### 8.4.  Assertion Redirect

To mitigate this threat, the assertion includes the identity of the RP for whom it was generated as "client\_id". The RP verifies that incoming assertions include its identity as the recipient of the assertion.

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

### 8.5.  Assertion Reuse

The assertion includes a timestamp and a short lifetime of validity. The Relying Party checks the timestamp and lifetime values to ensure that the assertion is currently valid.

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

### 8.6.  Secondary Authenticator Manufacture

Due to the large entropy requirement of the Artifact ("code") and short life nature of its validity, the success probability of this attack is extremely low.

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

### 8.7.  Secondary Authenticator Capture

Secondary authenticator (="code") is transmitted only through HTTPS, thus it is protected between the OP and the User-Agent, and User-Agent and the RP.

Only the place it can be captured is the User-Agent where the TLS session is terminated, and is possible if the User-Agent is infested by malwares. However, it renders no usefulness as long as the profile in use either RP authentication or assertion encryption.

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

### 8.8.  Assertion Substitution

Responses to assertion requests is bound to the corresponding requests by message order in HTTP, as both assertions and requests are protected by TLS that can detect and disallow malicious reordering of packets.

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

### 8.9.  Authentication Request Disclosure

If the authentication request is POSTed directly through a protected channel, it is not possible to disclose the authentication request.

If the Request File is encrypted by the OP's public key, the authentication request will not be disclosed unless OP's private key gets compromised or the encryption algorithm becomes vulnerable.

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

### 8.10.  Timing Attack

Timing attacks can be used to reduce the effective key length of the signature if the time required to return the response in case of a signature error and a correct signature differs. Care should be taken in the implementation to avoid this attack.

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

### 8.11.  Authentication Process Threats

In the category of Authentication Process Threats, following threats exists.

* Online guessing
* Phishing
* Pharming
* Eavesdropping
* Replay
* Session hijack
* Man-in-the-middle

Authentication process per se as described in NIST SP800-63-rev1 is out of scope for this protocol, but care SHOULD be taken to achieve appropriate protection.

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

### 9.  IANA Considerations

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

### 9.1.  OAuth Parameters Registry

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

### 9.1.1.  Scope Parameters

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Parameter name: openid, profile, email, address, PPID
* Parameter usage location: The end-user authorization endpoint request, the end-user authorization endpoint response, the token endpoint request, the token endpoint response, and the WWW-Authenticate header field.
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.1.2.  Authorization Request Parameter (display)

The following is the parameter registration request for the Authorization Request in this specification:

* Parameter name: display
* Parameter usage location: Authorization Request
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.1.3.  Authorization Request Parameter (prompt)

The following is the parameter registration request for the Authorization Request in this specification:

* Parameter name: prompt
* Parameter usage location: Authorization Request
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.1.4.  Authorization Request Parameter (nonce)

The following is the parameter registration request for the Authorization Request in this specification:

* Parameter name: nonce
* Parameter usage location: Authorization Request
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.1.5.  Authorization Request Parameter (audience)

The following is the parameter registration request for the Authorization Request in this specification:

* Parameter name: audience
* Parameter usage location: Authorization Request
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.1.6.  Authorization Request Parameter (request)

The following is the parameter registration request for the Authorization Request in this specification:

* Parameter name: request
* Parameter usage location: Authorization Request
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.1.7.  Authorization Request Parameter (request\_uri)

The following is the parameter registration request for the Authorization Request in this specification:

* Parameter name: request\_uri
* Parameter usage location: Authorization Request
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.1.8.  ID Token Response Parameters

The following is the parameter registration request for the ID Token Response in this specification:

* Parameter name: id\_token
* Parameter usage location: Authorization Response, Access Token Response
* Change controller: IETF
* Specification document(s): [[ this document ]]
* Related information: None

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

### 9.2.  OAuth Extensions Error Registry

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

### 9.2.1.  Authorization endpoint error (invalid\_request\_redirect\_uri)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: invalid\_request\_redirect\_uri
* Error usage location: Authorization endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 9.2.2.  Authorization endpoint error (login\_required)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: login\_required
* Error usage location: Authorization endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 9.2.3.  Authorization endpoint error (session\_selection\_required)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: session\_selection\_required
* Error usage location: Authorization endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 9.2.4.  Authorization endpoint error (approval\_required)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: approval\_required
* Error usage location: Authorization endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 9.2.5.  Authorization endpoint error (user\_mismatched)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: user\_mismatched
* Error usage location: Authorization endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 9.2.6.  Token endpoint error (invalid\_authorization\_code)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: invalid\_authorization\_code
* Error usage location: Token endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 9.2.7.  UserInfo endpoint error (invalid\_schema)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: invalid\_schema
* Error usage location: UserInfo endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 9.2.8.  Check ID endpoint error (invalid\_id\_token)

The following is the parameter value registration request for the "scope" parameter as defined in this specification:

* Error name: invalid\_schema
* Error usage location: UserInfo endpoint
* Related protocol extension:
* Change controller: IETF
* Specification document(s): [[this document ]]

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

### 10.  References

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

### 10.1. Normative References

|  |  |
| --- | --- |
| **[ISO29115]** | McCallister, E., “ITU-T Recommendation X.eaa | ISO/IEC 2nd CD 29115 -- Information technology - Security techniques - Entity authentication assurance framework,” ISO/IEC 29115. |
| **[ISO3166-1]** | International Organization for Standardization, “[ISO 3166-1:1997. Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes](http://www.w3.org/WAI/ER/IG/ert/iso639.htm),” 1997. |
| **[ISO639-1]** | International Organization for Standardization, “ISO 639-1:2002. Codes for the representation of names of languages -- Part 1: Alpha-2 code,” 2002. |
| **[JWE]** | Jones, M., Bradley, J., and N. Sakimura, “[JSON Web Encryption](http://tools.ietf.org/html/draft-jones-json-web-encryption),” July 2011. |
| **[JWS]** | Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “[JSON Web Signatures](http://tools.ietf.org/html/draft-jones-json-web-signature),” April 2011. |
| **[JWT]** | Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “[JSON Web Token](http://tools.ietf.org/html/draft-jones-json-web-token),” July 2011. |
| **[OAuth.2.0]** | Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “[OAuth 2.0 Authorization Protocol](http://tools.ietf.org/html/draft-ietf-oauth-v2),” September 2011. |
| **[OAuth.2.0.Bearer]** | Jones, M., Hardt, D., and D. Recordon, “[OAuth 2.0 Protocol: Bearer Tokens](http://tools.ietf.org/html/draft-ietf-oauth-v2-bearer),” September 2011. |
| **[OAuth.2.0.SC]** | Lodderstedt, T., Ed., McGloin, M., and P. Hunt, “[OAuth 2.0 Threat Model and Security Considerations](http://tools.ietf.org/html/draft-ietf-oauth-v2-threatmodel),” July 2011. |
| **[OpenID.Basic]** | Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “[OpenID Connect Basic Client 1.0](http://openid.net/specs/openid-connect-basic-1_0.html),” October 2011. |
| **[OpenID.Discovery]** | Sakimura, N., Bradley, J., Jones, M., and E. Jay, “[OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0.html),” September 2011. |
| **[OpenID.Registration]** | Sakimura, N., Bradley, J., Ed., and M. Jones, “[OpenID Connect Dynamic Client Registration 1.0](http://openid.net/specs/openid-connect-registration-1_0.html),” September 2011. |
| **[OpenID.Session]** | Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “[OpenID Connect Session Management 1.0](http://openid.net/specs/openid-connect-session-1_0.html),” September 2011. |
| **[OpenID.Standard]** | Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “[OpenID Connect Standard 1.0](http://openid.net/specs/openid-connect-standard-1_0.html),” October 2011. |
| **[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)). |
| **[RFC3339]** | [Klyne, G., Ed.](mailto:GK@ACM.ORG) and [C. Newman](mailto:chris.newman@sun.com), “[Date and Time on the Internet: Timestamps](http://tools.ietf.org/html/rfc3339),” RFC 3339, July 2002 ([TXT](http://www.rfc-editor.org/rfc/rfc3339.txt), [HTML](http://xml.resource.org/public/rfc/html/rfc3339.html), [XML](http://xml.resource.org/public/rfc/xml/rfc3339.xml)). |
| **[RFC4627]** | Crockford, D., “[The application/json Media Type for JavaScript Object Notation (JSON)](http://tools.ietf.org/html/rfc4627),” RFC 4627, July 2006 ([TXT](http://www.rfc-editor.org/rfc/rfc4627.txt)). |
| **[RFC5646]** | Phillips, A. and M. Davis, “[Tags for Identifying Languages](http://tools.ietf.org/html/rfc5646),” BCP 47, RFC 5646, September 2009 ([TXT](http://www.rfc-editor.org/rfc/rfc5646.txt)). |
| **[SP800-63]** | National Institute of Standards and Technology, “[NIST SP800-63rev.1: Electronic Authentication Guideline](http://csrc.nist.gov/publications/drafts/800-63-rev1/SP800-63-Rev1_Dec2008.pdf),” NIST SP800-63. |
| **[W3C.REC-html401-19991224]** | Raggett, D., Hors, A., and I. Jacobs, “[HTML 4.01 Specification](http://www.w3.org/TR/1999/REC-html401-19991224),” World Wide Web Consortium Recommendation REC-html401-19991224, December 1999 ([HTML](http://www.w3.org/TR/1999/REC-html401-19991224)). |
| **[zoneinfo]** | Public Domain, “[The tz database](http://www.twinsun.com/tz/tz-link.htm),” June 2011. |

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

### 10.2. Informative References

|  |  |
| --- | --- |
| **[OpenID.2.0]** | specs@openid.net, “OpenID Authentication 2.0,” 2007 ([TXT](http://www.openid.net/specs/openid-authentication-2_0.txt), [HTML](http://www.openid.net/specs/openid-authentication-2_0.html)). |
| **[RESPONSE.TYPES]** | de Medeiros, B., [Scurtescu, M.](mailto:mscurtescu@google.com), and [P. Tarjan](mailto:pt@fb.com), “[OAuth 2.0 Multiple Response Type Encoding Practices](http://openid.net/specs/oauth-v2-multiple-response-types-1_0.html),” October 2011. |

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

### Appendix A.  Acknowledgements

As a successor version of OpenID, this specification heavily relies on [OpenID Authentication 2.0 (specs@openid.net, “OpenID Authentication 2.0,” 2007.)](#OpenID.2.0) [OpenID.2.0]. Please refer to Appendix C of OpenID Authentication 2.0 for the full list of the contributors for that specification.

This specification is largely compliant with OAuth 2.0 draft 20. Please refer to the OAuth 2.0 specification for the list of contributors.

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

Anthony Nadalin (tonynad@microsoft.com), Microsoft

Axel Nennker (axel.nennker@telekom.de), Deutsche Telekom

Casper Biering (cb@peercraft.com), Peercraft

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

Chuck Mortimore (cmortimore@salesforce.com), Salesforce.com

David Recordon (dr@fb.com), Facebook

George Fletcher (george.fletcher@corp.aol.com), AOL

Hideki Nara (hideki.nara@gmail.com), Takt Communications

John Bradley (jbradely@mac.com), Protiviti Government Services

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

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

Paul Tarjan (pt@fb.com), Facebook

Ryo Itou (ritou@yahoo-corp.jp), Yahoo! Japan

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

### Appendix B.  Document History

[[ To be removed from the final specification ]]

-06

* Changed section 3.1.4.1 to say the errors are returned as defined by the response type not always as query paramaters. per ticket #174.
* Bumped version + date.
* Fixed section 3.3.3 to refer to errors in Bearer Token.
* Fixed 3.1.3 to ref the other response types ticket #173.
* Included reference to multiple response types.
* Fixed 3.1.2.1 to indicate default claims in id\_token.
* Fixed section 3.2.2 to reference the access token response from the token endpoint 4.1.4.
* Fixed section 3.2.1 to include refresh tokens.
* Fixed section 3.1.1 to be clear on JWT being the token format per ticket #171.

-05

* Changed check\_session to check\_id.
* schema=openid now required when requesting UserInfo.
* Removed issued\_to, since not well defined.
* Removed display values popup, touch, and mobile, since not well defined.

-04

* Changes associated with renaming "Lite" to "Basic Client" and replacing "Core" and "Framework" with "Messages" and "Standard".
* Numerous cleanups, including updating references.

-03

* Added secret\_type to the Token endpoint.
* Minor edits to the samples.

-02

* Incorporates feedback from Nat Sakimura.

-01

* First Draft that incorporates the merge of the Core and Framework specs.

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

### Authors' Addresses

|  |  |
| --- | --- |
|  | Nat Sakimura (editor) |
|  | Nomura Research Institute, Ltd. |
| **Email:** | [n-sakimura@nri.co.jp](mailto:n-sakimura@nri.co.jp) |
|  |  |
|  | David Recordon |
|  | Facebook |
| **Email:** | [dr@fb.com](mailto:dr@fb.com) |
|  |  |
|  | John Bradley |
|  | Protiviti Government Services |
| **Email:** | [jbradley@mac.com](mailto:jbradley@mac.com) |
|  |  |
|  | Breno de Medeiros |
|  | Google Inc. |
| **Email:** | [breno@google.com](mailto:breno@google.com) |
|  |  |
|  | Michael B. Jones |
|  | Microsoft Corporation |
| **Email:** | [mbj@microsoft.com](mailto:mbj@microsoft.com) |
|  |  |
|  | Edmund Jay |
|  | MGI1 |
| **Email:** | [ejay@mgi1.com](mailto:ejay@mgi1.com) |