N. Sakimura, Ed.
NRI
J. Bradley, Ed.
Independent
B. de Medeiros
Google
M. Jones
Microsoft
E. Jay
MGI1
C. Mortimore
Salesforce
August 11, 2011

OpenID Connect Lite 1.0 - draft 08

Abstract

OpenID Connect 1.0 is a simple identity layer on top of OAuth 2.0 protocol. It allows a web site to verify the identity of the user based on the authentication performed by the server, as well as to obtain basic profile information about the user in an interoperable and RESTful manner.

OpenID Connect Lite is a profile of the full openID Connect 1.0 Specification designed to be easy to read and impliment for Relying Parties. Identity providers should consult the main specification. This profile omits implimentation and security considerations for identity providers.

Implementation

Table of Contents

1. Requirements Notation and Conventions

TOC

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]** .

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.

2. Terminology

The following definitions define additional terminology used in this specification in addition to those defined in OAuth2

Followings are the additional terminology defined in this specification in addition to those defined in **OAuth 2.0** [OAuth.2.0].

Direct Communication
: Direct communication is a Client to Server communication which does not pass through the User-Agent.

ID Token
: A token that contains information about the authentication event. It is a signed token, but can be treated as opaque by the Lite profile. Relying partys wanting to process the token directly should refer to the full openID Connect 1.0 specification.

Should openID be capitalized?

Indirect Communication
: In indirect communication, messages are passed through the User-Agent.

Check Session Endpoint
: A protected resource that, when presented with an access token by the client, returns authentication information about the user represented by that access 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.

Query String Serialization
: In order to serialize the parameters using the query string serialization, the client constructs the string by adding the following parameters to the end-

user authorization endpoint URI query component using the `application/x-www-form-urlencoded` format as defined by **[W3C.REC-html401-19991224]**.

This starts to describe how the string is serialized... not so much what it is.

switch authorization and implict so it matches the presentation in the spec

TOC

Protocol Flows

Authorization requests follow two paths, the authorization code flow and the implicit grant flow. The authorization code flow is suitable for clients that can securely maintain client state between itself and the authorization server whereas the implicit grant flow is suitable for clients that cannot. The openID Connect Lite profile only documents clients using the implicit grant flow. Identity Providers MUST support both flows. Clients wanting to use the code flow and Identity Providers should consult the full openID Connect 1.0 specification.

authorization code

TOC

3.1. openID Connect Scopes

openID Connect uses scopes as defined in 4.2.1 of **OAuth 2.0** [OAuth.2.0].

The supported scopes are:

I find this (and the other optional definitions) confusing. Maybe something like "This scope represents the client's request for default profile information about the authorization user"

I think there should be a paragraph that makes it clear that the client retrieves these attributes by querying the user_info endpoint

openid
: REQUIRED This indicates that the authorization request is using the Connect profile. A id_token MUST be returned for this scope.

profile
: OPTIONAL Default profile information from the user_info endpoint.

email
: OPTIONAL an email address from the user_info endpoint

address
: OPTIONAL an address from the user_info endpoint.

These scopes are additive if a RP wanted the default profile including email and address they would request:

The following is a non-normative example of an Authorization Request URL. Note that the line wraps within the values are for display purpose only:

`scope=openid profile email phone`

This should go above the previous paragraph. And then a full URL example should be added

TOC

3.2. Implicit Flow

consists of

The implicit grant flow ~~follows~~ the following steps:

1. Client Prepares an Authorization Request containing the desired request parameters.
2. Client sends a request to the Authorization Server.
3. Authorization Server Authenticates the End-User.
4. Authorization Server Obtains the End-User Consent/Authorization.
5. Authorization Server Sends the End-User back to the Client with an Access Token and ID Token.

As currently written the redirect to the client only contains an access_token and no id_token.

3.2.1. Client Prepares an Authorization Request

Isn't this the same as "the client does not have a valid access token for the user"?

When the user wishes to access a Protected Resource, and the End-User Authorization has not yet been obtained, the Client prepares an Authorization Request to the authorization endpoint.

The scheme used in the Authorization URL MUST be HTTPS.

This binding further constrains the following request parameters:

response_type
: It MUST include `token`.

If this is the only document I need to read as a relying party, this is confusing because it obviously implies that there is some other document that this one is constraining. Is it sufficient to just list the Required and Optional parameters? Maybe put the "further constraining" statement in the response_type definition?

Other REQUIRED parameters in the

client_id
: The OAuth client identifier.

scope
: It MUST include `openid` as one of the space sparated strings ~~openid~~, optional scope strings of profile, email, and address are also supported.

redirect_uri
: A redirection URI where the response will be sent.

The request MAY contain the following optional parameters:

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

display
: A string value used to convey desired display format. The value are either `none`, `popup`, `touch`, or `mobile`.

prompt
: A space delimited list that can contain `login`, `consent`, and `select_account`. It is used to control the dialogue that is to be shown to the user upon the request.

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

Authorization servers MUST support the use of the HTTP `GET` method as define in **RFC 2616** [RFC2616] and MAY support the `POST` method at the authorization endpoint.

If using the HTTP `GET` method, the parameters are serialized using URI Query String Serialization. If using the HTTP `POST` method, the request parameters are added to the HTTP request entity-body using the `application/x-www-form-urlencoded` format as defined by **[W3C.REC-html401-19991224]**.

The following is a non-normative example of an Authorization Request URL. Note that the line wraps within the values are for display purpose only:

```
https://server.example.com/authorize?
response_type=token
&client_id=s6BhdRkqt3
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&scope=openid profile
&state=af0ifjsldkj
```

3.2.2. Client Sends a Request to the Authorization Server

TOC

Having constructed the URL, the client sends the End-User to the HTTPS End-User Authorization Endpoint using the URL. This MAY happen via HTTPS redirect, hyperlinking, or any other means of directing the User-Agent to the URL through Indirect Communication.

Following is a non-normative example using HTTP redirect. Note: Line wraps are for display purpose only.

This example needs to be updated (show scope)?

```
HTTP/1.1 302 Found
Location: https://server.example.com/authorize?
response_type=token ~~id_token~~
&client_id=s6BhdRkqt3
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&state=af0ifjsldkj
```

3.2.3. Authorization Server Obtains the End-User Consent/Authorization

TOC

The Authorization Server MUST obtain an authorization decision, for the requested scopes. This MAY be done by presenting the user with a dialogue that allows the user to recognize what he is consenting to and obtain his consent or by establishing approval via other means (for example, via previous administrative approval).

The openid scope grants the RP access to only the user identifyer and authenticates the session as beloning to that user.

identifier

If the user dosen't grant access to the openid scope then an error MUST be returned by the Identity provider.

Maybe put 'openid' in courier font or something to distringuish the string as a scope "literal"

All other scopes are optional. It is up to the Identity provider to determine if the authentication should proceed if the user delines to authorize scopes other than openid.

3.2.4. Authorization Server Sends the End-User Back to the Client

TOC

Once the authorization is determined, the Authorization Server returns a positive or negative response.

3.2.4.1. End-User Grants Authorization

TOC

authorization

If the resource owner grants the access request, the authorization server issues an access token and delivers it to the client by adding the following parameters to the query component of the redirection URI using the `application/x-www-form-urlencoded` format:

access_token
: REQUIRED. The Access Token.

I'm confused. Section 3.1 specifies that for a scope of 'openid' an id_token MUST be returned. Is the access_token the id_token? Or do we need to add id_token to this response as well?

state
: REQUIRED if the `state` parameter in the request. Set to the exact value of the `state` parameter received from the client.

The client can then use the Access Token to access protected resources at resource servers.

The following is a non-normative example. Line wraps after the second line is for the display purpose only:

Does there need to be any text describing the use of URL fragments and the importance of the # in the URL?

```
HTTP/1.1 302 Found
Location: https://client.example.com/#
access_token=SlAV32hkKG&
expires_in=3600&
&state=af0ifjsldkj
```

TOC

3.2.4.2. End-User Denies Authorization or Invalid Request

If the user denies the authorization or the user authentication fails, the server MUST return the negative authorization response as defined in 4.2.2.1 of **OAuth 2.0** [OAuth.2.0]. No other parameter SHOULD be returned.

3.3. Check Session Endpoint

Where did this id_token come from? It's referenced in section 3.1 but not returned anywhere.

The Check Session endpoint validates the id_token and returns a text **JSON** [RFC4627] object which contains information about the end user associated with the supplied id-token.

The id_token is used to manage the signon event and user identifier, separatly from access to the user_info endpoint and other OAuth 2.0 protected resources that the user is granting access to. The id_token is scoped to a particular client via the audiance and nonce.

A full explination of how to use the id_token to perform session management can be found in the **OpenID Connect Session Management 1.0** [openID.Session]

TOC

3.3.1. Check Session Request

The Check Session endpoint is an **OAuth 2.0** [OAuth.2.0] protected resource that complies with the **Bearer Token** [OAuth.2.0.Bearer] specification. As such, the access token SHOULD be specified via the HTTP Authorization header.

To request the information about the authentication performed on the user, the following parameter is sent to the Check Session Endpoint.

id_token
: REQUIRED. The access_token obtained from an OpenID Connect authorization request. This token MUST be sent as the access token.

So am I supposed to take the access_token returned in the authorization response and pass it in the HTTP Authorization header as the OAuth2 token when invoking this endpoint? If so, there this call has no parameters. I think I'm confused.

3.3.2. Check Session Response

TOC

The response is a text **JSON** [RFC4627] object using the `application/json` media type with the following parameters.

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** [RFC3339] for details regarding date/times in general and UTC in particular.

I think this forces the AS implementation to hold on to the nonce for the length of the "session". I suppose it could be kept in the access_token

iso29115
: OPTIONAL. (entity authentication assurance): Specifies the X.eaa / **ISO/IEC29115** [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.

issued_to
: OPTIONAL. The OAuth identifier of the client the token was issued to, only present if diffrent from aud.

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

```
Request:

GET /id_token
Host: server.example.com
Authorization: Bearer 7Fjfp0Z.Br1KtDRb.nfVdmIw

Response:

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

{
"iss": "http://server.example.com",
"user_id": "Jane Doe",
"aud": "http://client.example.net",
"exp":1311281970
}
```

We should change the example of "Jane Doe" for the user_id field.

3.3.3. Error Codes

TOC

In addition to the error codes defined in Section 5.2 of **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, malformed, is in an incorrect format, or expired.

So is the id_token really not a "token" in the "classic" sense but just a response from the AS with authentication/session information? To me, this error is really an invalid_access_token as that is all that was provided to the endpoint.

3.3.4. Verification

TOC

3.3.4.1. Request Verification

The authorization server validates the request to ensure all required parameters are present and valid.

TOC

3.3.4.2. Response Verification

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

Is there an expectation that this "ID Token" JSON object will be passed around? I don't think step 3 makes sense in verifying the response from the "check session endpoint"

1. If the authentication request contained a nonce, check that the returned nonce is valid.
2. Verify that the response was intended for the recipient, using the `aud` (audience) contained within the response.
3. If `issued_to` is present then it MUST contain an identifier for a trusted intermediary. If `issued_to` is unknown then the assertion MUST be rejected.
4. Check that the server that responded was really the intended server through a TLS/SSL server certificate check.
5. The Check Session Endpoint has not returned an error for the token being expired or invalid.

4. UserInfo Endpoint

How the client obtains additional tokens is not covered in this specification so I'd leave it out.

TOC

To obtain the additional attributes and tokens, the client makes a GET or POST request to the UserInfo Endpoint.

NOTE: In the Oauth 2.0 implicit flow, possession of an access token for a user_info endpoint is not proof that the user presenitng the token is the subject of the user_info endpoint. These tokens may be long lived and do not contain audiance information to validate that they were issued to a particular RP.

The id_token and Check Session Endpoint MUST be used to validate the identity of a session.

4.1. Requesting UserInfo

TOC

Clients MAY send requests with the following parameters to the UserInfo endpoint to obtain further information about the user. The UserInfo endpoint is an **OAuth 2.0** [OAuth.2.0] protected resource that complies with the **Bearer Token** [OAuth.2.0.Bearer] specification. As such, the access token SHOULD be specified via the HTTP Authorization header.

access_token
: REQUIRED. The access_token obtained from an OpenID Connect authorization request. This parameter MUST only be sent using one method through either HTTP Authorization header or query string.

query string only covers GET, should add text for POST parameters.

schema
: OPTIONAL. The schema in which the data is to be returned. The only predefined value is `openid`. If this parameter is not included, 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.

The following is a non-normative example. Line wraps are for display purpose only:

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer vF9dft4qmT
```

TOC

4.2. Client Receives UserInfo Response

If the requested schema is `openid`, the response MUST return a plain text **JSON** [RFC4627] structure that contains a set of members that are a subset of those defined below. Additional members (not specified below) MAY also be returned.

The members may be represented in multiple languages and scripts. To specify the languages and scripts, **BCP47** [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]** timezone database. For example, `Europe/Paris` or `America/Los_Angeles`.
locale	string	The user's locale, represented as an **RFC 5646** [RFC5646] language tag. This is typically an **ISO 639-1 Alpha-2** [ISO639-1] language code in lowercase and an **ISO 3166-1 Alpha-2** [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** [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** [RFC3339] datetime. For example, `2011-01-03T23:58:42+0000`.

Table 1: Reserved Member Definitions

Following is a non-normative example of such response:

```
{
 "name": "Jane Doe"
 "given_name": "Jane",
 "family_name": "Doe",
 "email": "janedoe@example.com",
 "picture": "http://example.com/janedoe/me.jpg"
}
```

TOC

4.2.1. Error Response

When some error condition arises, the UserInfo endpoint returns the Error Response. In addition to the standard **OAuth 2.0** [OAuth.2.0] errors, the UserInfo endpoint may return the following errors:

unsupported_schema
: The requested schema is unsupported.

5. Discovery and Registration

TOC

Some OpenID Connect installations can use a pre-configured set of OpenID Providers and/or Relying Parties. In those cases, it may not be necessary to support dynamic discovery of information about identities or services or dynamic registration of clients.

However, if installations choose to support unanticipated interactions between Relying Parties and OpenID Providers that do not have pre-configured relationships, they SHOULD accomplish this by implementing the facilities defined in the **OpenID Connect Discovery 1.0** [OpenID.Discovery] and **OpenID Connect Dynamic Client Registration 1.0** [OpenID.Registration] specifications.

6. Security Considerations

TOC

In addition to the Security Considerations in **OAuth 2.0** [OAuth.2.0], followings are the list of attack vectors and remedies that were considered for this specification.

For details of the attack vector, see **[SP800-63]**.

6.1. Assertion Manufacture/Modification

TOC

An assertion is the result of the authentication performed by the server that was provided to the client. The assertion is used to pass information about the user or the authentication process from the server to the client.

1. To mitigate this attack, 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 server MUST be authenticated. In this specification, the assertion is always sent over TLS/SSL protected channel.

6.2. Assertion Disclosure

TOC

This profile is subject to assertion disclosure in the user's browser, if it is compromised. Other OpenID Connect profiles should be used if this threat needs to be mitigated.

6.3. Assertion Redirect

TOC

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

6.4. Assertion Reuse

TOC

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

The use of a nonce in the request is recomended. The response from the Check Session Endpoint contains the nonce sent in the original request. This SHOULD be checked against a list of allready receved ID assertions to check for replays.

TOC

6.5. 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

6.6. Authentication Request Disclosure

Since the authentication request is sent over a protected channel, the disclosure may only happen at the User-Agent where the information is decrypted.

TOC

6.7. Authentication Process Threats

In the category of Authentication Process Threats, the following threats exist:

- Online Guessing
- Phishing
- Pharming
- Eavesdropping
- Replay
- Session Hijacking
- Man-in-the-Middle

The authentication process, per se, as described in **[SP800-63]** is out of scope for this protocol, but care SHOULD be taken to achieve appropriate protection.

TOC

6.8. Implicit Grant Flow Threats

In the implicit grant flow, the access token is returned in the fragment part of the client's redirect_uri through HTTPS, thus it is protected between the Server and the User-Agent, and User-Agent and the Client. The 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 malware.

TOC

6.9. Availability

When the server is down, user is likely to become unable to access the web server. To mitigate this risk, the client SHOULD allow user to associate multiple servers.

7. Privacy Considerations

TOC

The UserInfo response typically contains Personally Identifiable Information. As such, user consent for the release of the information for the specified purpose SHOULD be obtained at or prior to the authorization time in accordance with relevant regulations. The purpose of use is typically registered in association with the `redirect_uri`.

Only necessary UserInfo data should be stored at the client and the client SHOULD associate the received data with the purpose of use statement.

The server SHOULD make the UserInfo access log available to the user so that the user can monitor who accessed his data.

To protect the user from a possible correlation among clients, the use of a Pairwise Pseudonymous Identifier (PPID) as the `user_id` SHOULD be considered.

8. IANA Considerations

TOC

This document makes no request of IANA.

9. Normative References

TOC

[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**," 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.
[OAuth.2.0]	Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, "**OAuth 2.0 Authorization Protocol**," July 2011.
[OAuth.2.0.Bearer]	Jones, M., Ed., Recordon, D., and D. Hardt, "**The OAuth 2.0 Protocol: Bearer Tokens**," July 2011.
[OpenID.Discovery]	Sakimura, N., Bradley, J., Jones, M., and E. Jay, "**OpenID Connect Discovery 1.0**," July 2011.
[OpenID.Registration]	Sakimura, N., Bradley, J., Ed., and M. Jones, "**OpenID Connect Dynamic Client Registration 1.0 - draft 02**," July 2011.
[RFC2119]	**Bradner, S.**, "**Key words for use in RFCs to Indicate Requirement Levels**," BCP 14, RFC 2119, March 1997 (**TXT**, **HTML**, **XML**).
[RFC2616]	**Fielding, R.**, **Gettys, J.**, **Mogul, J.**, **Frystyk, H.**, **Masinter, L.**, **Leach, P.**, and **T. Berners-Lee**, "**Hypertext Transfer Protocol -- HTTP/1.1**," RFC 2616, June 1999 (**TXT**, **PS**, **PDF**, **HTML**, **XML**).
[RFC3339]	**Klyne, G., Ed.** and **C. Newman**, "**Date and Time on the Internet: Timestamps**," RFC 3339, July 2002 (**TXT**, **HTML**, **XML**).
[RFC4627]	Crockford, D., "**The application/json Media Type for JavaScript Object Notation (JSON)**," RFC 4627, July 2006 (**TXT**).
[RFC5646]	Phillips, A. and M. Davis, "**Tags for Identifying Languages**," BCP 47, RFC 5646, September 2009 (**TXT**).
[SP800-63]	National Institute of Standards and Technology, "**NIST SP800-63rev.1: Electronic Authentication Guideline**," NIST SP800-63.
[W3C.REC-html401-19991224]	Hors, A., Jacobs, I., and D. Raggett, "**HTML 4.01 Specification**," World Wide Web Consortium Recommendation REC-html401-19991224, December 1999 (**HTML**).
[openID.Session]	**de Medeiros, B.**, "**OpenID Connect Session Management 1.0 -draft 03**," July 2011.
[zoneinfo]	Public Domain, "**The tz database**," June 2011.

TOC

Appendix A. 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.

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

Casper Biering (cb@peercraft.com), Peercraft

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

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

George Fletcher (gffletch@aol.com), AOL

Edmund Jay (ejay@mgi1.com), MGI1

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

Chuck Mortimore (cmortimore@salesforce.com), Salesforce

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

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

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

TOC

Appendix B. Document History

[[To be removed from the final specification]]

-08

- Added note about IdP needing to read the full spec.
- Reverted back to GET for introspection based on Google feedback.
- changed scopes to openid, profile, address, email. To make tham additive
- Changed introspection to Check Session Endpoint to be consistent with session management.
- Changed validation rules, the Check session endpoint will return an error for expired or invalid tokens, so the client dosen't need to check expirery.
- Added explination of why a id_token is used to verify identity rather than the user_info access token.

-07

- Changed introspection to post
- changed user_info from id to user_id to be consistent with introspection.
- Fixed introspection example to use id_token rather than access token.
- removed asking for id_token in response type

-06

- Only require the `token` flow in Lite. Removed `code` flow.
- Make `id_token` required. The `id_token` is treated as opaque.
- Rearranged sections for readability.
- Dropped the `schema` parameter to the Introspection endpoint, which was

formerly a string with the value `user_id`. This is unnecessary since the `id_token` parameter already can be used to disambiguate the intended uses(s) of the endpoint.
- Dropped the requested audience from the Lite spec, which was formerly the identifier of the target audience of the response. This could be part of the Standard spec, but is an advanced scenario, and so not appropriate for Lite.
- Reference the Discovery and Registration specs, since they're needed for interaction between non-pre-configured parties (so that OpenID Connect installations can be Open).

-05

- Corrected issues raised by Casper Biering.
- Created the OpenID Connect Lite specification.

-04

- Correct issues raised by Pam Dingle and discussed on the mailing list after the 7-Jul-11 working group call.
- Adopted long_names.

-03

- Correct issues raised by Johnny Bufu and discussed on the 7-Jul-11 working group call.

-02

- Consistency and cleanup pass, including removing unused references.

-01

- Initial draft

Authors' Addresses

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

John Bradley (editor)
Independent
Email: jbradley@mac.com

Breno de Medeiros
Google
Email: breno@google.com

Michael B. Jones
Microsoft Corporation
Email: mbj@microsoft.com

Edmund Jay
MGI1
Email: ejay@mgi1.com

Chuck Mortimore
Salesforce
Email: cmortimore@salesforce.com