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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Draft | N. Sakimura | |  | NRI | |  | J. Bradley | |  | Protiviti | |  | M. Jones | |  | Microsoft | |  | B. de Medeiros | |  | Google | |  | C. Mortimore | |  | Salesforce | |  | E. Jay | |  | MGI1 | |  | September 2, 2011 | |

# OpenID Connect Session Management 1.0 - draft 04

### Abstract

OpenID Connect is an identity protocol 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 document describes how to manage sessions for OpenID Connect.

### 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].

### Table of Contents

[**1.**](#anchor1)  Introduction  
[**2.**](#terminology)  Terminology  
[**3.**](#anchor2)  Session Management  
    [**3.1.**](#anchor3)  Creating Sessions  
        [**3.1.1.**](#IDToken)  ID Token  
        [**3.1.2.**](#auth_req)  Authorization Request  
        [**3.1.3.**](#anchor4)  Token Endpoint  
        [**3.1.4.**](#anchor5)  Implicit (User-Agent) Flow  
        [**3.1.5.**](#anchor7)  Authorization Code (Web Server) Flow  
        [**3.1.6.**](#anchor11)  4th Party Native Applications  
    [**3.2.**](#anchor13)  Session Management Endpoints  
        [**3.2.1.**](#anchor14)  Refresh Session  
        [**3.2.2.**](#anchor15)  Check Session  
        [**3.2.3.**](#anchor16)  End Session  
    [**3.3.**](#SessionSync)  Session Synchronization  
[**4.**](#IANA)  IANA Considerations  
[**5.**](#Security)  Security Considerations  
[**6.**](#rfc.references1)  Normative References  
[**Appendix A.**](#Acknowledgements)  Acknowledgements  
[**Appendix B.**](#anchor18)  Document History  
[**§**](#rfc.authors)  Authors' Addresses

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

### 1.  Introduction

Sessions are used to keep track of information and interactions for users across multiple pages. This creates a sense of continuity, customization, and a more pleasant experience for the users. Visitors to an OpenID relying party site accessing protected resources will be asked for authentication and authorization. Upon user authorization, the user will be granted access to the requested resources. The site may perform other background tasks for the user using the same authenticated session. This allows the user to have a simplified experience without being asked for authorization each time and may even allow the user to go off-line while the tasks are being performed. This specification describes how OpenID Connect sessions can be created, used, and terminated.

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

### 2.  Terminology

In addition to the terminology defined in the [OpenID Connect Messages 1.0 (Sakimura, N., Recordon, D., Bradley, J., de Medeiros, B., Jones, M., and E. Jay, “OpenID Connect Messages 1.0,” September 2011.)](#OpenID.Messages) [OpenID.Messages] specification, the following terms are defined:

Client

An application obtaining authorization and making protected resource requests.

Client Identifier

A unique identifier that the client uses to identify itself to the OP.

Identifier

An Identifier is either an "http" or "https" URI, (commonly referred to as a "URL" within this document), or an account URI. This document defines various kinds of Identifiers, designed for use in different contexts.

Base64url

Base 64 Encoding [[RFC3548] (Josefsson, S., “The Base16, Base32, and Base64 Data Encodings,” July 2003.)](#RFC3548) with URL and Filename Safe Alphabet without padding.

Client Servlet

A web application that is also an OAuth 2.0 client.

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

### 3.  Session Management

OpenID Connect supports life-cycle session management and synchronization for third parties that support authentication with the authorization server. In addition, session management for fourth parties such as desktop, native or mobile applications that utilize authorization server credentials at fourth party web sites are also supported.

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

### 3.1.  Creating Sessions

To create a session, the client sends an authorization request to the authorization server with id\_token as one of the response\_type values.

If the response\_type includes the value code, then an [ID token (ID Token)](#IDToken) is returned in the response of the Token Endpoint when the Access Token is retrieved.

If the response\_type includes the value token, then an ID token is returned as a fragment parameter in the redirect\_uri specified in the request.

In either case, an ID Token will also be returned along with the access token when submitting a refresh token to the token access endpoint if the initial authorization request included id\_token in the response\_type parameter.

The ID Token serves as a key to an authenticated user session and contains claims for the user.

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

### 3.1.1.  ID Token

This specification defines an ID Token as a signed [JWT (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token,” July 2011.)](#JWT) [JWT] that minimally contains the following claims:

issuer

REQUIRED. The unique identifier of the issuer of the claims

client\_id

REQUIRED. The unique identifier of the client.

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.

audience

REQUIRED. The [JWT (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token,” July 2011.)](#JWT) [JWT] aud (audience) claim.

exp

REQUIRED. The [JWT (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Token,” July 2011.)](#JWT) [JWT] exp (expiration time) claim.

pape

OPTIONAL. (TBD) If we want this token to be short, we probably want to define a shorter equivalent of PAPE.

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.

The ID Token MAY contain other claims. The ID Token can be used to access session information from an authenticated session or to pass a session to other applications.

|  |
| --- |
| [**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,” July 2011.)](#OAuth.2.0) [OAuth.2.0] defines 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 (Pending OAuth 2.0 changes). The value MUST include id\_token for requesting an ID Token from a session.

In addition, this specification defines the following extension parameters.

nonce

OPTIONAL. A random, unique string. The nonce value is returned in the ID token.

id\_token\_audience

OPTIONAL. The identifier of the target audience for an ID token.

Following is a non-normative example when they are sent in the query parameters serialization:

GET /authorize?scope=openid&response\_type=token%20id\_token

&client\_id=s6BhdRkqt3

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

Host: server.example.com

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

### 3.1.3.  Token Endpoint

The Token Endpoint MUST return an ID Token if id\_token is included in the response\_type parameter of the authorization request.

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

### 3.1.3.1.  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. The parameters in the successful response are defined in Section 4.2.2 of [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” July 2011.)](#OAuth.2.0) [OAuth.2.0] .

In addition, this specification defines the following additional return parameters:

id\_token

REQUIRED if it was requested in the authorization request. It may be a [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS] of the [ID token (ID Token)](#IDToken).

Following is a non-normative example:

{

"access\_token": "SlAV32hkKG",

"token\_type": "JWT",

"refresh\_token": "8xLOxBtZp8",

"expires\_in": 3600,

"id\_token":"jwt\_header.jwt\_part2.jwt\_part3"

}

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

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

### 3.1.4.  Implicit (User-Agent) Flow

User-agents can use the OAuth implicit grant flow by including token and id\_token in the response\_type of the authorization request to get an ID Token.

1. The user-agent makes an authorization request to the authorization endpoint.
2. The authorization server authenticates the user
3. The authorization server returns an access and ID token.
4. The user-agent and client servlet can then use the session management endpoints by presenting the ID token to the endpoints.

+----------------------------------+

+----------+ | |

| | | +----------------------+ |

| | | | Authorization | |

| | | | server | |

|user-agent| | +----------------------+ |

| | | | +---------------+ | |

| |>----(1)-------------|------|-->| Authorization | | |

| |<----(3)-------------|------|--<| Endpoint (2) | | |

+----------+ | | +---------------+ | |

^ +----------|------|-->| Check\_Session | | |

| | +--------|------|--<| EndPoint | | |

| | | | | +---------------+ | |

v | | | +----------------------+ |

+----------+ (4)| | | |

| | | | | |

|client |>---------+ | | +----------------------+ |

|servlet |<-----------+ | | | |

| | | | UserInfo Endpoint | |

| | | | | |

| |>--------------------|----->| | |

| |<--------------------|-----<| | |

| | | | | |

| | | | | |

+----------+ | +----------------------+ |

| |

| |

+----------------------------------+

+-----------------------------+

| |

| Authorization |

| Server |

+-------------+ | |

| | | +--------------------+ |

| User-Agent | | | Refresh Session | |

| | (4) | | Endpoint | |

| |>-------------|---->| | |

| |<-------------|----<| | |

| | | | | |

| | | +--------------------+ |

| | (4) | | End Session | |

| |>-------------|---->| Endpoint | |

| |<-------------|----<| | |

| | | | | |

| | | +--------------------+ |

+-------------+ +-----------------------------+

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

### 3.1.4.1.  Implicit Flow Request

The authorization request parameter values are constrained as follows.

response\_type

A space delimited, case sensitive list of string values (Pending OAuth 2.0 changes). The value MUST include token and id\_token and to request an access and ID Token from the session.

Following is a non-normative example of a request using query parameters serialization:

GET /authorize?scope=openid&response\_type=token%20id\_token

&client\_id=s6BhdRkqt3

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

Host: server.example.com

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

### 3.1.4.2.  Implicit Flow Response

When the response\_type in the request includes 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,” July 2011.)](#OAuth.2.0) [OAuth.2.0] in the query fragment of the response.

In addition, when response\_type includes id\_token, an ID Token MUST be returned in the query fragment of the response.

Following is a non-normative example of a response:

HTTP/1.1 302 Found

Location: https://client.example.com/cb#access\_token=i1WsRn1uB1&token\_type=JWT&id\_token=jwt\_header.jwt\_part2.jwt\_part3

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

### 3.1.5.  Authorization Code (Web Server) Flow

Web server clients can use the OAuth authorization code flow by including code and id\_token in the response\_type parameter of the authorization request to obtain an ID token. The ID Token is returned along with the access token after the client submits the authorization code to the access token endpoint.

1. The user-agent makes an authorization request to the authorization endpoint.
2. The authorization server authenticates the user
3. The authorization server returns an authorization code.
4. The client sends authorization code to the token access endpoint.
5. The authorization server verifies the authorization token and returns an access and ID token
6. The user-agent and client servlet can then use the session management endpoints by presenting the ID token to the endpoints.

+----------------------------------+

+----------+ | |

| | | +----------------------+ |

| | | | Authorization | |

| | | | server | |

|user-agent| | +----------------------+ |

| | | | +---------------+ | |

| |>-----(1)------------|------|-->| Authorization | | |

| |<-----(3)------------|------|--<| Endpoint (2) | | |

+----------+ | | +---------------+ | |

^ +----------|------|-->| Access Token | | |

| | +--------|------|--<| EndPoint | | |

| | | | | +---------------+ | |

v | | | | | Session | | |

+----------+ | | | | | Management | | |

| | | | | | | Endpoints | + |

|client |>-----(4)-+ | | | +---------------+ | |

|servlet |<-----(5)---+ | +----------------------+ |

| | | |

| | | +----------------------+ |

| | | | | |

| | | | UserInfo Endpoint | |

| |>--------------------|----->| | |

| |<--------------------|-----<| | |

+----------+ | +----------------------+ |

| |

| |

+----------------------------------+

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

### 3.1.5.1.  Authorization Code Flow Request

The authorization request parameter values are constrained as follows.

response\_type

A space delimited, case sensitive list of string values (Pending OAuth 2.0 changes). The value MUST include code and id\_token and to request an access and ID Token from the session.

Following is a non-normative example of a request using query parameters serialization:

GET /authorize?scope=openid&response\_type=code%20id\_token

&client\_id=s6BhdRkqt3

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

Host: server.example.com

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

### 3.1.5.2.  Authorization Code Flow Response

When the response\_type in the request includes 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,” July 2011.)](#OAuth.2.0) [OAuth.2.0] in the query of the response.

In addition, when response\_type includes id\_token, the ID token is retrieved from the token access endpoint.

Following is a non-normative example of a response:

HTTP/1.1 302 Found

Location: https://client.example.com/cb?code=i1WsRn1uB1

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

### 3.1.5.3.  Token Access Request

The client uses the authorization code to make a request to the token access endpoint to retrieve an access and ID token.

The request format is defined in section 4.1.3 of the [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” July 2011.)](#OAuth.2.0) [OAuth.2.0] specification.

Following is a non-normative example of a token access endpoint request:

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

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

grant\_type=authorization\_code&client\_id=s6BhdRkqt3&

code=i1WsRn1uB1&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb

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

### 3.1.5.4.  Token Access Response

The access and ID token is returned in the response.

The response format is defined in section 4.1.4 of the [OAuth 2.0 (Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, “OAuth 2.0 Authorization Protocol,” July 2011.)](#OAuth.2.0) [OAuth.2.0] specification.

Following is a non-normative example of a token access endpoint response:

HTTP/1.1 200 OK

Content-Type: application/json

Cache-Control: no-store

{

"access\_token":"SlAV32hkKG",

"token\_type":"JWT",

"expires\_in":3600,

"refresh\_token":"8xLOxBtZp8",

"id\_token":"jwt\_header.jwt\_part2.jwt\_part3"

}

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

### 3.1.6.  4th Party Native Applications

Fourth party native applications involve four parties: 1) the user, 2) the native (desktop) application, 3) the authorization server, and 4) the client servlet web application. The native application uses protected resources from a client servlet but it integrates with authentication services from the authorization server directly. The native application directs the user to perform authentication at the authorization server to obtain access and ID tokens. The tokens can then be used to access protected resources at the web servlet client. The process of obtaining an ID Token for the native application is very similar to that of using the code authorization (web server) flow method. However, the target audience of the ID Token is not the native application, but that of the client servlet. The client needs to indicate the target audience for the ID Token by setting the id\_token\_audience parameter in the authorization request to that of the identifier of the client servlet.

+-----------------------------+

+----------------+ | |

| | | Authorization |

| Native App | | Server |

| | | |

| | | +--------------------+ |

| |>------------------|----->| Authorization | |

| |<------------------|-----<| Endpoint | |

| | | | | |

| | | | | |

| | | +--------------------+ |

| | | | Access Token | |

| |>------------------|----->| Endpoint | |

| |<------------------|-----<| | |

| | | | | |

| | | +--------------------+ |

| |>------------------|----->| Session Mgmt | |

| |<------------------|-----<| Endpoints | |

| | | | | |

+----------------+ | | | |

^ | | | |

| | +--------------------+ |

v | |

+----------------+ | |

| Client | +-----------------------------+

| Servlet |

| |

+----------------+

When accessing protected resources at the client servlet, the native application sends the ID Token as an Auth HTTP header in the request. The client servlet can check the validity of the ID Token by verifying the cryptographic information or by sending the token to the Check Session Endpoint [OpenID Connect Messages 1.0 (Sakimura, N., Recordon, D., Bradley, J., de Medeiros, B., Jones, M., and E. Jay, “OpenID Connect Messages 1.0,” September 2011.)](#OpenID.Messages) [OpenID.Messages].

GET /resource1

Auth: jwt\_header.jwt\_part2.jwt\_part3

Host: servlet.example.com

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

### 3.1.6.1.  Browser Load

Some native applications may wish to start an authenticated browser session for the same user. The native application starts a browser with the location of the client servlet and passing an ID Token as a query parameter. The client servlet immediately initiates a request to the refresh session endpoint with the ID Token. The user may need to reauthenticate at the authorization server. The client servlet then gets an ID Token that is [session synchronized (Session Synchronization)](#SessionSync) with the authorization server.

+--------------------------+

+------------+ +-----------+ | |

| | | | | Authorization Server |

| Native App |>---->|User-Agent | | |

| | | | | +------------------+ |

| | | |>------|--->| Session Refresh | |

| | | |<------|---<| Endpoint | |

+------------+ +-----------+ | | | |

^ ^ | +------------------+ |

| | | |

v v | |

+--------------------------------+ | |

| | | |

| Client Servlet | | |

| | | |

+--------------------------------+ +--------------------------+

GET

/refesh\_token?state=bar&redirect\_uri=https://foo.com/oauth2callback&id\_token=$id\_token // never uses immediate mode here, to allow login

Host: www.example.com

Response:

HTTP/1.1 302 Found

Location: https://foo.com/oauth2callback?state=bar&session=$new\_id\_token

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

### 3.2.  Session Management Endpoints

To manage a session, the client sends a request to the session management endpoints at the authorization server. The session management endpoints at the authorization server are:

Refresh Session

Refreshes an expired ID Token

Check Session

Get a plain text JSON structure from a ID Token

End Session

Ends a session

Authorization servers MUST support the use of the HTTP "GET" method as define in [RFC 2616 (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.)](#RFC2616) [RFC2616] and MAY support the "POST" method for all session management endpoints.

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

### 3.2.1.  Refresh Session

To refresh an ID Token that has expired, the client sends a request to the Refresh Session endpoint with an ID Token. A new ID Token is returned.

Request Parameters:

id\_token

REQUIRED. A previously issued ID Token from an authorization request. The ID Token MAY be expired.

redirect\_uri

REQUIRED. An absolute URI to which the authorization server will redirect the user-agent to with the new ID Token.

state

REQUIRED. An opaque value used by the Client to maintain state between the request and callback. If provided, the Authorization Server MUST include this value when redirecting the user-agent back to the Client. Clients are strongly advised to use this variable to relate the request and response.

Response:

The authorization server returns the following parameters in the query of the redirect\_uri URI specified in the request:

id\_token

REQUIRED.A new ID Token.

state

REQUIRED. The value of the state parameter in the request.

In a typical HTTP binding, an HTTP 302 is returned to redirect the user-agent to the specified redirect\_uri in the request with a new ID Token in the query fragment.

The following is a non-normative session refresh request:

Request:

GET /op/refresh\_session?id\_token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6

ImNsaWVudC5leGFtcGxlLmNvbSJ9.eyJpc3N1ZXIiOiJodHRwOlwvXC9zZXJ2ZXIuZXhhbXBs

ZS5jb20iLCJjbGllbnRfaWQiOiJjbGllbnQuZXhhbXBsZS5jb20iLCJhdWRpZW5jZSI6ImNsa

WVudC5leGFtcGxlLmNvbSIsImlkIjoidXNlcl8yMzQyMzQiLCJleHAiOjEzMDM4NTI4ODB9.a

JwagC6501Da-zK-X8Az9B-Y625aSEfxVuBpFEDjOxQ

&state=bar&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fidtoken\_cb

Host: server.example.com

Response:

HTTP/1.1 302 OK

Location: http://client.example.com/idtoken\_cb#id\_token=eyJ0eXAiOiJKV1QiLCJh

bGciOiJIUzI1NiIsImtpZCI6ImNsaWVudC5leGFtcGxlLmNvbSJ9.eyJpc3N1ZXIiOiJodHRwO

lwvXC9zZXJ2ZXIuZXhhbXBsZS5jb20iLCJjbGllbnRfaWQiOiJjbGllbnQuZXhhbXBsZS5jb20

iLCJhdWRpZW5jZSI6ImNsaWVudC5leGFtcGxlLmNvbSIsImlkIjoidXNlcl8yMzQyMzQiLCJle

HAiOjEzMDM4NTI4ODB9.aJwagC6501Da-zK-X8Az9B-Y625aSEfxVuBpFEDjOxQ&state=bar&

expires\_in=3600

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

### 3.2.2.  Check Session

For clients that are not capable of dealing with JWS signed ID Tokens, they can send the ID Token to the Check Session endpoint. It will validate the ID Token and return a plain text JSON structure of the ID Token.

Request Parameters:

id\_token

REQUIRED. A previously issued ID Token

Response:

The response body is a plain text JSON structure of the claims in the ID token.

If the ID token is a [JWS (Jones, M., Balfanz, D., Bradley, J., Goland, Y., Panzer, J., Sakimura, N., and P. Tarjan, “JSON Web Signatures,” April 2011.)](#JWS) [JWS], then it is the base64url decoded payload of the signed ID Token. In a typical HTTP binding, the response is a HTTP 200 response code with the content-type header set to "application/json".

The following is a non-normative example of a check session request:

Request:

POST /op/check\_session?id\_token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6

ImNsaWVudC5leGFtcGxlLmNvbSJ9.eyJpc3N1ZXIiOiJodHRwOlwvXC9zZXJ2ZXIuZXhhbXBs

ZS5jb20iLCJjbGllbnRfaWQiOiJjbGllbnQuZXhhbXBsZS5jb20iLCJhdWRpZW5jZSI6ImNsa

WVudC5leGFtcGxlLmNvbSIsImlkIjoidXNlcl8yMzQyMzQiLCJleHAiOjEzMDM4NTI4ODB9.a

JwagC6501Da-zK-X8Az9B-Y625aSEfxVuBpFEDjOxQ

Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

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

"client\_id","http://client.example.com",

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

"user\_id":"user\_328723",

"exp":1303852880

}

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

### 3.2.3.  End Session

To end the session, the client sends an ID Token to the End Session endpoint. Upon receiving the request, the authorization server performs the logout flow for the user and then redirects the user-agent to the specified redirect\_uri.

Request Parameters:

id\_token

REQUIRED. A previously issued ID Token

state

REQUIRED. An opaque value used by the Client to maintain state between the request and callback. If provided, the Authorization Server MUST include this value when redirecting the user-agent back to the Client. Clients are strongly advised to use this variable to relate the request and response.

redirect\_uri

REQUIRED. An absolute URI to which the authorization server will redirect the user-agent.

Response:

The authorization server returns the following parameters in the query of the redirect\_uri URI specified in the request:

state

REQUIRED. The value of the state parameter in the request.

In HTTP binding, the response is a HTTP 302 redirect response to the redirect\_uri specified in the request.

The following is a non-normative session refresh request:

Request:

GET /op/end\_session?id\_token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImtpZCI6

ImNsaWVudC5leGFtcGxlLmNvbSJ9.eyJpc3N1ZXIiOiJodHRwOlwvXC9zZXJ2ZXIuZXhhbX

BsZS5jb20iLCJjbGllbnRfaWQiOiJjbGllbnQuZXhhbXBsZS5jb20iLCJhdWRpZW5jZSI6I

mNsaWVudC5leGFtcGxlLmNvbSIsImlkIjoidXNlcl8yMzQyMzQiLCJleHAiOjEzMDM4NTI4

ODB9.aJwagC6501Da-zK-X8Az9B-Y625aSEfxVuBpFEDjOxQ

&state=bar

&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fendtoken\_cb

Host: server.example.com

...

Authorization server performs logout flow

...

Response:

HTTP/1.1 302 OK

Location: http://client.example.com/endtoken\_cb?state=bar

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

### 3.3.  Session Synchronization

An ID Token is usually bound to a user's sign in session at the authorization server, but in some cases, such as offline access by a web server or native application, it may not be. ID Tokens obtained in the following scenarios are bound to a user's signed-in state at the authorization server:

* Redeeming a code for an access and ID Token by way of indirect communication through the browser
* Obtaining an access and ID Token in the authorization response through the browser
* Obtaining an ID Token at the refresh session endpoint by submitting a previously issued ID Token

ID Tokens obtained in the above manner are session synchronized.

If an ID Token is obtained by submitting a refresh token at the access token endpoint, then the resulting ID Token is not bound to a user's sign in state at the authorization server. The client may be in offline mode or the user has logged out from the authorization server. If a session bound ID Token is desired, the client should obtain a new ID Token by sending a request to the refresh session endpoint.

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

### 4.  IANA Considerations

This document makes no request of IANA.

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

### 5.  Security Considerations

TBD

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

### 6. Normative References

|  |  |
| --- | --- |
| **[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),” July 2011. |
| **[OpenID.Messages]** | Sakimura, N., Recordon, D., Bradley, J., de Medeiros, B., Jones, M., and E. Jay, “[OpenID Connect Messages 1.0](http://openid.net/specs/openid-connect-messages-1_0.html),” September 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)). |
| **[RFC2616]** | [Fielding, R.](mailto:fielding@ics.uci.edu), [Gettys, J.](mailto:jg@w3.org), [Mogul, J.](mailto:mogul@wrl.dec.com), [Frystyk, H.](mailto:frystyk@w3.org), [Masinter, L.](mailto:masinter@parc.xerox.com), [Leach, P.](mailto:paulle@microsoft.com), and [T. Berners-Lee](mailto:timbl@w3.org), “[Hypertext Transfer Protocol -- HTTP/1.1](http://tools.ietf.org/html/rfc2616),” RFC 2616, June 1999 ([TXT](http://www.rfc-editor.org/rfc/rfc2616.txt), [PS](http://www.rfc-editor.org/rfc/rfc2616.ps), [PDF](http://www.rfc-editor.org/rfc/rfc2616.pdf), [HTML](http://xml.resource.org/public/rfc/html/rfc2616.html), [XML](http://xml.resource.org/public/rfc/xml/rfc2616.xml)). |
| **[RFC3548]** | Josefsson, S., “[The Base16, Base32, and Base64 Data Encodings](http://tools.ietf.org/html/rfc3548),” RFC 3548, July 2003 ([TXT](http://www.rfc-editor.org/rfc/rfc3548.txt)). |

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

### Appendix A.  Acknowledgements

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

### Appendix B.  Document History

[[ To be removed from the final specification ]]

-04

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

-03

* Corrected examples.

-02

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

-01

* Consistency and cleanup pass, including removing unused references.

-00

* Split from core when all optional features were removed.

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

### Authors' Addresses

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