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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Draft | N. Sakimura | |  | NRI | |  | J. Bradley | |  | Ping Identity | |  | M. Jones | |  | Microsoft | |  | B. de Medeiros | |  | Google | |  | C. Mortimore | |  | Salesforce | |  | E. Jay | |  | Illumila | |  | June 27, 2013 | |

# OpenID Connect Implicit Client Profile 1.0 - draft 11

### Abstract

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

OpenID Connect Implicit Client Profile 1.0 is a profile of the OpenID Connect Messages 1.0 and OpenID Connect Standard 1.0 specifications that is designed to be easy to read and implement for basic Web-based Relying Parties using the OAuth implicit grant type. This specification intentionally duplicates content from the Messages and Standard specifications to provide a self-contained implementation profile for basic Web-based Relying Parties using the OAuth implicit grant type.

OpenID Providers and non-Web-based applications should instead consult the Messages and Standard specifications.

### Table of Contents

[**1.**](#Introduction)  Introduction  
    [**1.1.**](#rnc)  Requirements Notation and Conventions  
    [**1.2.**](#terminology)  Terminology  
[**2.**](#ProtocolElements)  Protocol Elements  
    [**2.1.**](#ImplicitFlow)  Implicit Flow  
        [**2.1.1.**](#AuthorizationRequest)  Client Prepares Authorization Request  
            [**2.1.1.1.**](#RequestParameters)  Request Parameters  
        [**2.1.2.**](#implicit_req)  Client Sends Request to Authorization Server  
        [**2.1.3.**](#Authenticates)  Authorization Server Authenticates End-User  
        [**2.1.4.**](#Consent)  Authorization Server Obtains End-User Consent/Authorization  
        [**2.1.5.**](#implicit_res)  Authorization Server Sends End-User Back to Client  
            [**2.1.5.1.**](#implicit_ok)  End-User Grants Authorization  
            [**2.1.5.2.**](#implicit_authz_error)  End-User Denies Authorization or Invalid Request  
            [**2.1.5.3.**](#implicit_callback)  Example Redirect URI Response  
    [**2.2.**](#id_token)  ID Token  
        [**2.2.1.**](#id.token.validation)  ID Token Validation  
        [**2.2.2.**](#access.token.validation)  Access Token Validation  
    [**2.3.**](#userinfo)  UserInfo Endpoint  
        [**2.3.1.**](#UserInfoRequest)  UserInfo Request  
        [**2.3.2.**](#UserInfoResponse)  UserInfo Response  
        [**2.3.3.**](#UserInfoErrorResponse)  UserInfo Error Response  
    [**2.4.**](#scopes)  Scope Values  
    [**2.5.**](#StandardClaims)  Standard Claims  
        [**2.5.1.**](#address_claim)  Address Claim  
        [**2.5.2.**](#ClaimsLanguagesAndScripts)  Claims Languages and Scripts  
        [**2.5.3.**](#claim.stability)  Claim Stability and Uniqueness  
[**3.**](#self_issued)  Self-Issued OpenID Provider  
    [**3.1.**](#self_issued.discovery)  Self-Issued OpenID Provider Discovery  
    [**3.2.**](#self_issued.registration)  Self-Issued OpenID Provider Registration  
        [**3.2.1.**](#SelfIssuedRegistrationRequest)  Providing Additional Registration Information  
    [**3.3.**](#self_issued.request)  Self-Issued OpenID Provider Request  
    [**3.4.**](#self_issued.response)  Self-Issued OpenID Provider Response  
    [**3.5.**](#self_issued.validation)  Self-Issued ID Token Validation  
[**4.**](#Serializations)  Serializations  
    [**4.1.**](#qss)  Query String Serialization  
    [**4.2.**](#form_serialization)  Form Serialization  
[**5.**](#stringops)  String Operations  
[**6.**](#tls)  TLS Version  
[**7.**](#ImplementationConsiderations)  Implementation Considerations  
    [**7.1.**](#disco_reg)  Discovery and Registration  
[**8.**](#security_considerations)  Security Considerations  
    [**8.1.**](#TLS_requirements)  TLS Requirements  
[**9.**](#privacy_considerations)  Privacy Considerations  
    [**9.1.**](#PII)  Personally Identifiable Information  
    [**9.2.**](#AccessMonitoring)  Data Access Monitoring  
    [**9.3.**](#Correlation)  Correlation  
[**10.**](#IANA)  IANA Considerations  
[**11.**](#rfc.references1)  References  
    [**11.1.**](#rfc.references1)  Normative References  
    [**11.2.**](#rfc.references2)  Informative References  
[**Appendix A.**](#Acknowledgements)  Acknowledgements  
[**Appendix B.**](#Notices)  Notices  
[**Appendix C.**](#History)  Document History  
[**§**](#rfc.authors)  Authors' Addresses

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

### 1.  Introduction

OpenID Connect Implicit Client Profile 1.0 is a profile of the [OpenID Connect Messages 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Messages 1.0,” June 2013.)](#OpenID.Messages) [OpenID.Messages] and [OpenID Connect Standard 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Standard 1.0,” June 2013.)](#OpenID.Standard) [OpenID.Standard] specifications that is designed to be easy to read and implement for basic Web-based Relying Parties using the OAuth implicit grant type. This specification intentionally duplicates content from the Messages and Standard specifications to provide a self-contained implementation profile for basic Web-based Relying Parties using the OAuth implicit grant type.

See the [OpenID Connect Basic Client Profile 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, “OpenID Connect Basic Client Profile 1.0,” June 2013.)](#OpenID.Basic) [OpenID.Basic] specification for a related profile for basic Web-based Relying Parties using the OAuth authorization\_code grant type. OpenID Providers and non-Web-based applications should instead consult the Messages and Standard specifications. This profile omits implementation and security considerations for OpenID Providers and non-Web-based applications.

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

### 1.1.  Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.)](#RFC2119).

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

All uses of [JSON Web Signature (JWS) (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](#JWS) [JWS] data structures in this specification utilize the JWS Compact Serialization; the JWS JSON Serialization is not used.

When the RFC 2119 language applies to the behavior of OpenID Providers, it is in this specification for explanatory value to help Client implementers understand the expected behavior of OpenID Providers.

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

### 1.2.  Terminology

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

This specification also defines the following terms:

Claim

Piece of information asserted about an Entity.

Claims Provider

Server that can return Claims about an Entity.

End-User

Human Resource Owner.

Entity

Something that has a separate and distinct existence and that can be identified in a context. An End-User is one example of an Entity.

ID Token

[JSON Web Token (JWT) (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Token (JWT),” May 2013.)](#JWT) [JWT] that contains Claims about the authentication event. It MAY contain other Claims.

Issuer

Entity that issues a set of Claims.

Issuer Identifier

URL using the https scheme that acts as a verifiable identifier for an Issuer.

OpenID Provider (OP)

OAuth 2.0 Authorization Server that is capable of providing Claims to a Relying Party.

Pairwise Pseudonymous Identifier (PPID)

Identifier that identifies the Entity to a Relying Party that cannot be correlated with the Entity's PPID at another Relying Party.

Personally Identifiable Information (PII)

Information that (a) can be used to identify the natural person to whom such information relates, or (b) is or might be directly or indirectly linked to a natural person to whom such information relates.

Relying Party (RP)

OAuth 2.0 Client application requiring Claims from an OpenID Provider.

Self-Issued OpenID Provider

Personal OpenID Provider that issues self-signed ID Tokens.

UserInfo Endpoint

Protected resource that, when presented with an Access Token by the Client, returns authorized information about the End-User represented by the corresponding Authorization Grant.

Validation

Process intended to establish the soundness or correctness of a construct.

Verification

Process intended to test or prove the truth or accuracy of a fact or value.

Voluntary Claim

Claim specified by the Client as being useful but not Essential for the specific task requested by the End-User.

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

### 2.  Protocol Elements

Authorization Requests can follow one of two paths; the Implicit Flow or the Authorization Code Flow. The Authorization Code Flow is suitable for Clients that can securely maintain a Client Secret between themselves and the Authorization Server whereas, the Implicit Flow is suitable for Clients that cannot.

This specification only provides information that is sufficient for basic Clients using the Implicit Flow.

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

### 2.1.  Implicit Flow

The Implicit Flow consists of 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.

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

### 2.1.1.  Client Prepares Authorization Request

When the Client 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.

Communication with the Authorization Endpoint MUST utilize TLS. See [Section 8.1 (TLS Requirements)](#TLS_requirements) for more information on using TLS.

Clients MAY construct the request using the HTTP GET or the HTTP POST method as defined 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].

If using the HTTP GET method, the parameters are serialized using the Query String Serialization, per [Section 4.1 (Query String Serialization)](#qss). 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] (Hors, A., Raggett, D., and I. Jacobs, “HTML 4.01 Specification,” December 1999.)](#W3C.REC-html401-19991224).

The following is a non-normative example of an Authorization Request URL (with line wraps within values for display purposes only):

https://server.example.com/authorize?

response\_type=id\_token%20token

&client\_id=s6BhdRkqt3

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

&scope=openid%20profile

&state=af0ifjsldkj

&nonce=n-0S6\_WzA2Mj

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

### 2.1.1.1.  Request Parameters

This profile of OpenID Connect uses the following OAuth 2.0 request parameters:

response\_type

REQUIRED. This value MUST include id\_token and token, as a space delimited list. This requests that both an Access Token and an ID Token be returned in the URL fragment component of the response, as specified in [OAuth 2.0 Multiple Response Type Encoding Practices (de Medeiros, B., Ed., Scurtescu, M., and P. Tarjan, “OAuth 2.0 Multiple Response Type Encoding Practices,” June 2013.)](#OAuth.Responses) [OAuth.Responses].

client\_id

REQUIRED. OAuth 2.0 Client Identifier.

scope

REQUIRED. Space delimited, case sensitive list of ASCII OAuth 2.0 scope values. OpenID Connect requests MUST contain the openid scope value. OPTIONAL scope values of profile, email, address, phone, and offline\_access are also defined. See [Section 2.4 (Scope Values)](#scopes) for more about the scope values defined by this specification.

redirect\_uri

REQUIRED. Redirection URI to which the response will be sent. This MUST be pre-registered with the OpenID Provider. This URI MUST exactly match one of the redirect\_uris registered for the Client, with the matching performed as described in Section 6.2.1 of [[RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.)](#RFC3986) (Simple String Comparison). If the Client uses the OAuth implicit grant type, the redirection URI MUST NOT use the http scheme unless the Client is a native application, in which case it MAY use the http: scheme with localhost as the hostname.

state

RECOMMENDED. Opaque value used to maintain state between the request and the callback. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation is done by cryptographically binding the value of this parameter with the browser cookie.

This specification also defines the following request parameters:

nonce

REQUIRED. String value used to associate a Client session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authorization Request to the ID Token. Sufficient entropy MUST be present in the nonce values used to prevent attackers from guessing values. One method to achieve this is to store a random value as a signed session cookie, and pass the value in the nonce parameter. In that case, the nonce in the returned ID Token can be compared to the signed session cookie to detect ID Token replay by third parties.

display

OPTIONAL. ASCII string value that specifies how the Authorization Server displays the authentication and consent user interface pages to the End-User. The defined values are:

page

The Authorization Server SHOULD display authentication and consent UI consistent with a full User-Agent page view. If the display parameter is not specified this is the default display mode.

popup

The Authorization Server SHOULD display authentication and consent UI consistent with a popup User-Agent window. The popup User-Agent window SHOULD be 450 pixels wide and 500 pixels tall.

touch

The Authorization Server SHOULD display authentication and consent UI consistent with a device that leverages a touch interface. The Authorization Server MAY attempt to detect the touch device and further customize the interface.

wap

The Authorization Server SHOULD display authentication and consent UI consistent with a "feature phone" type display.

prompt

OPTIONAL. Space delimited, case sensitive list of ASCII string values that specifies whether the Authorization Server prompts the End-User for reauthentication and consent. The defined values are:

none

The Authorization Server MUST NOT display any authentication or consent user interface pages. An error is returned if the End-User is not already authenticated or the Client does not have pre-configured consent for the requested Claims or does not fulfill other conditions for processing. This can be used as a method to check for existing authentication and/or consent.

login

The Authorization Server SHOULD prompt the End-User for reauthentication. If it cannot prompt the End-User, it MUST return an error.

consent

The Authorization Server SHOULD prompt the End-User for consent before returning information to the Client.

select\_account

The Authorization Server SHOULD prompt the End-User to select a user account. This allows an End-User who has multiple accounts at the Authorization Server to select amongst the multiple accounts that they might have current sessions for. If it cannot prompt the End-User, it MUST return an error.

The prompt parameter can be used by the Client to make sure that the End-User is still present for the current session or to bring attention to the request. If this parameter contains none with any other value, an error is returned.

max\_age

OPTIONAL. Maximum Authentication Age. Specifies the allowable elapsed time in seconds since the last time the End-User was actively authenticated. If the elapsed time is greater than this value, the OP MUST attempt to actively re-authenticate the End-User. When max\_age is used, the ID Token returned MUST include an auth\_time Claim Value.

ui\_locales

OPTIONAL. End-User's preferred languages and scripts for the user interface, represented as a space-separated list of [BCP47 (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.)](#RFC5646) [RFC5646] language tag values, ordered by preference. For instance, the value "fr-CA fr en" represents a preference for French as spoken in Canada, then French (without a region designation), followed by English (without a region designation). An error SHOULD NOT result if some or all of the requested locales are not supported by the OpenID Provider.

claims\_locales

OPTIONAL. End-User's preferred languages and scripts for Claims being returned, represented as a space-separated list of [BCP47 (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.)](#RFC5646) [RFC5646] language tag values, ordered by preference. An error SHOULD NOT result if some or all of the requested locales are not supported by the OpenID Provider.

id\_token\_hint

OPTIONAL. Previously issued ID Token passed to the Authorization Server as a hint about the End-User's current or past authenticated session with the Client. This SHOULD be present when prompt=none is used. If the End-User identified by the ID Token is logged in or is logged in by the request, then the Authorization Server returns a positive response; otherwise, it SHOULD return a negative response. The Authorization Server need not be listed as an audience of the ID Token when it is used as an id\_token\_hint value.

login\_hint

OPTIONAL. Hint to the Authorization Server about the login identifier the End-User might use to log in (if necessary). This hint can be used by an RP if it first asks the End-User for their e-mail address (or other identifier) and then wants to pass that value as a hint to the discovered authorization service. It is RECOMMENDED that the hint value match the value used for discovery. This value MAY also be a phone number in the format specified for the phone\_number Claim. The use of this parameter is left to the OP's discretion.

acr\_values

OPTIONAL. Requested Authentication Context Class Reference values. Space-separated string that specifies the acr values that the Authorization Server is being requested to use for processing this authentication request, with the values appearing in order of preference. The Authentication Context Class satisfied by the authentication performed is returned as the acr Claim Value, as specified in [Section 2.2 (ID Token)](#id_token). The acr Claim is requested as a Voluntary Claim by this parameter.

registration

OPTIONAL. This parameter is used by the Client to provide information about itself to a Self-Issued OP that would normally be provided to an OP during Dynamic Client Registration, as specified in [Section 3.2.1 (Providing Additional Registration Information)](#SelfIssuedRegistrationRequest). The registration parameter SHOULD NOT be used when the OP is not a Self-Issued OP.

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

### 2.1.2.  Client Sends Request to Authorization Server

Having constructed the Authorization Request, the Client sends it to the Authorization Endpoint using HTTPS.

Following is a non-normative example using HTTP redirect (with line wraps within values for display purposes only):

HTTP/1.1 302 Found

Location: https://server.example.com/authorize?

response\_type=id\_token%20token

&client\_id=s6BhdRkqt3

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

&scope=openid%20profile

&state=af0ifjsldkj

&nonce=n-0S6\_WzA2Mj

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

### 2.1.3.  Authorization Server Authenticates End-User

The Authorization Server logs in the End-User or verifies whether he is logged in, depending upon the request parameter values used. If interaction with the End-User occurs over an HTTP channel, it MUST use TLS, as per [Section 8.1 (TLS Requirements)](#TLS_requirements). The exact authentication methods used are out of scope for this specification.

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

### 2.1.4.  Authorization Server Obtains End-User Consent/Authorization

The Authorization Server obtains an authorization decision for the requested Claims. This can done by presenting the End-User with a dialogue that allows the End-User to recognize what he is consenting to and obtain his consent or by establishing consent via other means (for example, via previous administrative consent).

The openid scope value declares that this OAuth 2.0 request is an OpenID Connect request. Use of all other scope values is OPTIONAL.

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

### 2.1.5.  Authorization Server Sends End-User Back to Client

Once the authorization is determined, the Authorization Server returns a successful response or an error response.

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

### 2.1.5.1.  End-User Grants 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 fragment component of the redirection URI using the application/x-www-form-urlencoded format as defined in Section 4.2.2 of [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749] and [OAuth 2.0 Multiple Response Type Encoding Practices (de Medeiros, B., Ed., Scurtescu, M., and P. Tarjan, “OAuth 2.0 Multiple Response Type Encoding Practices,” June 2013.)](#OAuth.Responses) [OAuth.Responses].

In the Implicit Flow, the entire response is returned in the fragment component of the redirection URI, as defined in 4.2.2 of [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749].

access\_token

REQUIRED. Access Token for the UserInfo Endpoint.

token\_type

REQUIRED. OAuth 2.0 Token Type value. The value MUST be Bearer or another token\_type value that the Client has negotiated with the Authorization Server. Clients implementing this profile MUST support the [OAuth 2.0 Bearer Token Usage (Jones, M. and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” October 2012.)](#RFC6750) [RFC6750] specification. This profile only describes the use of bearer tokens.

id\_token

REQUIRED. ID Token.

state

OAuth 2.0 state value. REQUIRED if the state parameter is present in the Client Authorization Request. Clients MUST verify that the state value is equal to the value of state parameter in the Authorization Request.

expires\_in

OPTIONAL. Expiration time of the Access Token in seconds since the response was generated.

The Client can then use the Access Token to access protected resources at Resource Servers.

The following is a non-normative example (with line wraps for the display purposes only):

HTTP/1.1 302 Found

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

access\_token=SlAV32hkKG

&token\_type=bearer

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

&expires\_in=3600

&state=af0ifjsldkj

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

### 2.1.5.2.  End-User Denies Authorization or Invalid Request

If the End-User denies the authorization or the End-User authentication fails, the Authorization Server MUST return the error Authorization Response as defined in 4.2.2.1 of [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749]. No other parameters SHOULD be returned.

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

### 2.1.5.3.  Example Redirect URI Response

The Client MUST provide a way for the User-Agent to parse the fragment encoded response and post it to the Web Server Client for validation.

The following is an example of a JavaScript file that a Client might host at its redirect\_uri. This is loaded by the redirect from the Authorization Server. The fragment component is parsed and then sent by POST to a URI that will validate the information received.

Following is a non-normative example of a Redirect URI response:

GET /cb HTTP/1.1

Host: client.example.org

HTTP/1.1 200 OK

Content-Type: text/html

<script type="text/javascript">

// First, parse the query string

var params = {}, postBody = location.hash.substring(1),

regex = /([^&=]+)=([^&]\*)/g, m;

while (m = regex.exec(postBody)) {

params[decodeURIComponent(m[1])] = decodeURIComponent(m[2]);

}

// And send the token over to the server

var req = new XMLHttpRequest();

// using POST so query isn't logged

req.open('POST', 'https://' + window.location.host +

'/catch\_response', true);

req.setRequestHeader('Content-Type',

'application/x-www-form-urlencoded');

req.onreadystatechange = function (e) {

if (req.readyState == 4) {

if (req.status == 200) {

// If the response from the POST is 200 OK, perform a redirect

window.location = 'https://'

+ window.location.host + '/redirect\_after\_login'

}

// if the OAuth response is invalid, generate an error message

else if (req.status == 400) {

alert('There was an error processing the token')

} else {

alert('Something other than 200 was returned')

}

}

};

req.send(postBody);

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

### 2.2.  ID Token

The ID Token is a security token that contains Claims about the authentication event and other requested Claims. The ID Token is represented as a [JSON Web Token (JWT) (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Token (JWT),” May 2013.)](#JWT) [JWT].

The ID Token is used to manage the authentication event and user identifier and is scoped to a particular Client via the aud (audience) and nonce Claims.

The following Claims are used within the ID Token:

iss

REQUIRED. Issuer Identifier for the Issuer of the response. The iss value is a case sensitive string.

sub

REQUIRED. Subject identifier. A locally unique and never reassigned identifier within the Issuer for the End-User, which is intended to be consumed by the Client, e.g., 24400320 or AItOawmwtWwcT0k51BayewNvutrJUqsvl6qs7A4. It MUST NOT exceed 255 ASCII characters in length. The sub value is a case sensitive string.

aud

REQUIRED. Audience(s) that this ID Token is intended for. It MUST contain the OAuth 2.0 client\_id of the Relying Party as an audience value. It MAY also contain identifiers for other audiences. In the general case, the aud value is an array of case sensitive strings. In the special case when there is one audience, the aud value MAY be a single case sensitive string.

exp

REQUIRED. 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 time is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the 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. The exp value is a number.

iat

REQUIRED. Time at which the JWT was issued. The time is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. The iat value is a number.

auth\_time

OPTIONAL or REQUIRED. Time when the End-User authentication occurred. The time is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. When a max\_age request is made then this Claim is REQUIRED. The auth\_time value is a number.

nonce

REQUIRED. String value used to associate a Client session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authorization Request to the ID Token. Clients MUST verify that the nonce Claim Value is equal to the value of the nonce parameter sent in the Authorization Request. If present in the Authorization Request, Authorization Servers MUST include a nonce Claim in the ID Token with the Claim Value being the nonce value sent in the Authorization Request. Use of the nonce is REQUIRED when using the implicit flow. The nonce value is a case sensitive string.

at\_hash

REQUIRED. Access Token hash value. If the ID Token is issued with an access\_token in an implicit flow, this is REQUIRED, which is the case for this profile. Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the access\_token value, where the hash algorithm used is the hash algorithm used in the alg parameter of the ID Token's [JWS (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](#JWS) [JWS] header. For instance, if the alg is RS256, hash the access\_token value with SHA-256, then take the left-most 128 bits and base64url encode them. The at\_hash value is a case sensitive string.

acr

OPTIONAL. Authentication Context Class Reference. String specifying an Authentication Context Class Reference value that identifies the Authentication Context Class that the authentication performed satisfied. The value "0" indicates the End-User authentication did not meet the requirements of [ISO/IEC 29115 (International Organization for Standardization, “ISO/IEC 29115:2013 -- Information technology - Security techniques - Entity authentication assurance framework,” March 2013.)](#ISO29115) [ISO29115] level 1. Authentication using a long-lived browser cookie, for instance, is one example where the use of "level 0" is appropriate. Authentications with level 0 SHOULD never be used to authorize access to any resource of any monetary value. An absolute URI or a [registered name (Johansson, L., “An IANA Registry for Level of Assurance (LoA) Profiles,” August 2012.)](#RFC6711) [RFC6711] SHOULD be used as the acr value; registered names MUST NOT be used with a different meaning than that which is registered. Parties using this claim will need to agree upon the meanings of the values used, which may be context-specific. The acr value is a case sensitive string.

amr

OPTIONAL. Authentication Methods References. JSON array of strings that are identifiers for authentication methods used in the authentication. For instance, values might indicate that both password and OTP authentication methods were used. The definition of particular values to be used in the amr Claim is beyond the scope of this specification. Parties using this claim will need to agree upon the meanings of the values used, which may be context-specific. The amr value is an array of case sensitive strings.

azp

OPTIONAL or REQUIRED. Authorized Party - the party to which the ID Token was issued. If present, it MUST contain the OAuth 2.0 client\_id of the party that will be using it. This Claim is only REQUIRED when the party requesting the ID Token is not the same as the sole audience of the ID Token. It MAY be included even when the Authorized Party is the same as the sole audience. The azp value is a case sensitive string containing a StringOrURI value.

sub\_jwk

NOT RECOMMENDED or REQUIRED. Public key value used to check the signature of an ID Token issued by a Self-Issued OpenID Provider, as specified in [Section 3 (Self-Issued OpenID Provider)](#self_issued). The key is a bare key in JWK [[JWK] (Jones, M., “JSON Web Key (JWK),” May 2013.)](#JWK) format (not an X.509 certificate value). Use of the sub\_jwk Claim is REQUIRED when the OP is a Self-Issued OP and is NOT RECOMMENDED when the OP is not Self-Issued. The iss value is a JSON object.

ID Tokens MAY contain other Claims. Any Claims used that are not understood MUST be ignored.

ID Tokens MUST be signed using [JWS (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](#JWS) [JWS]. Clients MUST validate the ID Token per [Section 2.2.1 (ID Token Validation)](#id.token.validation).

ID Tokens SHOULD NOT use the JWS or JWE x5u, x5c, jku, or jwk header parameter fields. Instead, key values and key references used for ID Tokens are communicated in advance using Discovery and Registration parameters.

The following is a non-normative example of a base64url decoded ID Token:

{

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

"sub": "24400320",

"aud": "s6BhdRkqt3",

"nonce": "n-0S6\_WzA2Mj",

"exp": 1311281970,

"iat": 1311280970,

"at\_hash": "MTIzNDU2Nzg5MDEyMzQ1Ng"

}

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

### 2.2.1.  ID Token Validation

If any of the validation procedures defined in this specification fail, any operations requiring the information that failed to correctly validate MUST be aborted and the information that failed to validate MUST NOT be used.

To validate the ID Token in the Authorization Response, the Client MUST do the following:

1. The Client MUST validate that the aud (audience) Claim contains its client\_id value registered at the Issuer identified by the iss (issuer) Claim as an audience. The ID Token MUST be rejected if the ID Token does not list the Client as a valid audience, or if it contains additional audiences not trusted by the Client.
2. If the ID Token contains multiple audiences, the Client SHOULD verify that an azp Claim is present.
3. If an azp (authorized party) Claim is present, the Client SHOULD verify and that its client\_id is the Claim value.
4. The Client MUST validate the signature of the ID Token according to [JWS (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](" \l "JWS) [JWS] using the algorithm specified in the alg parameter of the JWT header.
5. The alg value SHOULD be RS256. Validation of tokens using other signing algorithms is described in the [OpenID Connect Messages 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Messages 1.0,” June 2013.)](" \l "OpenID.Messages) [OpenID.Messages] specification.
6. The Client MUST use the signing key provided in Discovery by the Issuer. The Issuer MUST exactly match the value of the iss (issuer) Claim.
7. The current time MUST be less than the value of the exp Claim (possibly allowing for some small leeway to account for clock skew).
8. The iat Claim can be used to reject tokens that were issued too far away from the current time, limiting the amount of time that nonces need to be stored to prevent attacks. The acceptable range is Client specific.
9. The value of the nonce Claim MUST be checked to verify that it is the same value as the one that was sent in the Authorization Request. The Client SHOULD check the nonce value for replay attacks. The precise method for detecting replay attacks is Client specific.
10. If the acr Claim was requested, the Client SHOULD check that the asserted Claim Value is appropriate. The meaning and processing of acr Claim Values is out of scope for this specification.
11. When a max\_age request is made, the Client SHOULD check the auth\_time Claim value and request re-authentication if it determines too much time has elapsed since the last End-User authentication.

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

### 2.2.2.  Access Token Validation

To validate an Access Token issued with an ID Token in the implicit flow, the Client SHOULD do the following:

1. Hash the octets of the ASCII representation of the access\_token with the hash algorithm specified in [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](" \l "JWA) [JWA] for the alg parameter in the ID Token's [JWS (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](" \l "JWS) [JWS] header.
2. Take the left-most half of the hash and base64url encode it.
3. The value of at\_hash in the ID Token MUST match the value produced in the previous step if at\_hash is present in the ID Token.

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

### 2.3.  UserInfo Endpoint

The UserInfo Endpoint is an OAuth 2.0 Protected Resource that returns Claims about the authenticated End-User. The location of the UserInfo Endpoint MUST be a URL using the https scheme, which MAY contain port, path, and query parameter components. The returned Claims are represented by a JSON object that contains a collection of name and value pairs for the Claims.

Communication with the UserInfo Endpoint MUST utilize TLS. See [Section 8.1 (TLS Requirements)](#TLS_requirements) for more information on using TLS.

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

### 2.3.1.  UserInfo Request

Clients send requests to the UserInfo Endpoint to obtain Claims about the End-User. The UserInfo Endpoint is an [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749] Protected Resource that complies with the [OAuth 2.0 Bearer Token Usage (Jones, M. and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” October 2012.)](#RFC6750) [RFC6750] specification. The Access Token SHOULD be sent using the Authorization header field. The following parameters are defined for use in UserInfo Requests:

access\_token

REQUIRED. Access Token obtained from an OpenID Connect Authorization Request. This parameter MUST only be sent using one method using either the Authorization header field or a form-encoded POST body parameter.

The following is a non-normative example of a UserInfo Request:

GET /userinfo HTTP/1.1

Host: server.example.com

Authorization: Bearer SlAV32hkKG

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

### 2.3.2.  UserInfo Response

The UserInfo Claims MUST be returned as the members of a JSON object. The response body SHOULD be encoded using UTF-8. The Claims defined in [Section 2.5 (Standard Claims)](#StandardClaims) can be returned, as can additional Claims not specified there.

If a Claim is not returned, that Claim Name SHOULD be omitted from the JSON object representing the Claims; it SHOULD NOT be present with a null or empty string value.

The sub (subject) Claim MUST always be returned in the UserInfo Response.

NOTE: The UserInfo Endpoint response is not guaranteed to be about the End-User identified by the sub (subject) element of the ID Token. The sub Claim in the UserInfo Endpoint response MUST be verified to exactly match the sub Claim in the ID Token before using additional UserInfo Endpoint Claims.

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

### 2.3.3.  UserInfo Error Response

When an error condition occurs, the UserInfo Endpoint returns an Error Response as defined in Section 3 of [OAuth 2.0 Bearer Token Usage (Jones, M. and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” October 2012.)](#RFC6750) [RFC6750].

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

### 2.4.  Scope Values

OpenID Connect Clients use scope values as defined in 3.3 of [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749] to specify what access privileges are being requested for Access Tokens. The scopes associated with Access Tokens determine what resources will be available when they are used to access OAuth 2.0 protected endpoints. For OpenID Connect, scopes can be used to request that specific sets of information be made available as Claim Values. This specification describes only the scope values used by OpenID Connect.

OpenID Connect allows additional scope values to be defined and used. Scope values used that are not understood by an implementation SHOULD be ignored.

Claims requested by the following scopes are treated by Authorization Servers as Voluntary Claims.

OpenID Connect defines the following scope values:

openid

REQUIRED. Informs the Authorization Server that the Client is making an OpenID Connect request. If the openid scope value is not present, the behavior is entirely unspecified.

profile

OPTIONAL. This scope value requests access to the End-User's default profile Claims, which are: name, family\_name, given\_name, middle\_name, nickname, preferred\_username, profile, picture, website, gender, birthdate, zoneinfo, locale, and updated\_at.

email

OPTIONAL. This scope value requests access to the email and email\_verified Claims.

address

OPTIONAL. This scope value requests access to the address Claim.

phone

OPTIONAL. This scope value requests access to the phone\_number and phone\_number\_verified Claims.

offline\_access

This scope value MUST NOT be used with the OpenID Connect Implicit Client Profile 1.0. See the [OpenID Connect Basic Client Profile 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, “OpenID Connect Basic Client Profile 1.0,” June 2013.)](#OpenID.Basic) [OpenID.Basic] for its usage in that profile.

Multiple scope values MAY be used by creating a space delimited, case sensitive list of ASCII scope values.

The Claims requested by the profile, email, address, and phone scope values are returned from the UserInfo Endpoint, as described in [Section 2.3.2 (UserInfo Response)](#UserInfoResponse).

In some cases, the End-User will be given the option to have the OpenID Provider decline to provide some or all information requested by Clients. To minimize the amount of information that the End-User is being asked to disclose, a Client can elect to only request a subset of the information available from the UserInfo Endpoint.

The following is a non-normative example of a scope Request.

scope=openid profile email phone

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

### 2.5.  Standard Claims

This profile defines a set of standard Claims. They are returned by normal OPs in the UserInfo Response and are returned by Self-Issued OPs in the ID Token.

|  |  |  |
| --- | --- | --- |
| **Member** | **Type** | **Description** |
| sub | string | Subject - Identifier for the End-User at the Issuer. |
| name | string | End-User's full name in displayable form including all name parts, possibly including titles and suffixes, ordered according to the End-User's locale and preferences. |
| given\_name | string | Given name(s) or first name(s) of the End-User. Note that in some cultures, people can have multiple given names; all can be present, with the names being separated by space characters. |
| family\_name | string | Surname(s) or last name(s) of the End-User. Note that in some cultures, people can have multiple family names or no family name; all can be present, with the names being separated by space characters. |
| middle\_name | string | Middle name(s) of the End-User. Note that in some cultures, people can have multiple middle names; all can be present, with the names being separated by space characters. Also note that in some cultures, middle names are not used. |
| nickname | string | Casual name of the End-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. |
| preferred\_username | string | Shorthand name that the End-User wishes to be referred to at the RP, such as janedoe or j.doe. This value MAY be any valid JSON string including special characters such as @, /, or whitespace. This value MUST NOT be relied upon to be unique by the RP. (See [Section 2.5.3 (Claim Stability and Uniqueness)](#claim.stability).) |
| profile | string | URL of the End-User's profile page. The contents of this Web page SHOULD be about the End-User. |
| picture | string | URL of the End-User's profile picture. This URL MUST refer to an image file (for example, a PNG, JPEG, or GIF image file), rather than to a Web page containing an image. Note that this URL SHOULD specifically reference a profile photo of the End-User suitable for displaying when describing the End-User, rather than an arbitrary photo taken by the End-User. |
| website | string | URL of the End-User's Web page or blog. This Web page SHOULD contain information published by the End-User or an organization that the End-User is affiliated with. |
| email | string | End-User's preferred e-mail address. Its value MUST conform to the [RFC 5322 (Resnick, P., Ed., “Internet Message Format,” October 2008.)](#RFC5322) [RFC5322] addr-spec syntax. This value MUST NOT be relied upon to be unique by the RP, as discussed in [Section 2.5.3 (Claim Stability and Uniqueness)](#claim.stability). |
| email\_verified | boolean | True if the End-User's e-mail address has been verified; otherwise false. When this Claim Value is true, this means that the OP took affirmative steps to ensure that this e-mail address was controlled by the End-User at the time the verification was performed. The means by which an e-mail address is verified is context-specific, and dependent upon the trust framework or contractual agreements within which the parties are operating. |
| gender | string | End-User's gender. Values defined by this specification are female and male. Other values MAY be used when neither of the defined values are applicable. |
| birthdate | string | End-User's birthday, represented as an [ISO 8601:2004 (International Organization for Standardization, “ISO 8601:2004. Data elements and interchange formats - Information interchange - Representation of dates and times,” 2004.)](#ISO8601-2004) [ISO8601‑2004] YYYY-MM-DD format. The year MAY be 0000, indicating that it is omitted. To represent only the year, YYYY format is allowed. Note that depending on the underlying platform's date related function, providing just year can result in varying month and day, so the implementers need to take this factor into account to correctly process the dates. |
| zoneinfo | string | String from zoneinfo [[zoneinfo] (Public Domain, “The tz database,” June 2011.)](#zoneinfo) time zone database representing the End-User's time zone. For example, Europe/Paris or America/Los\_Angeles. |
| locale | string | End-User's locale, represented as a [BCP47 (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 | End-User's preferred telephone number. [E.164 (International Telecommunication Union, “E.164: The international public telecommunication numbering plan,” 2010.)](#E.164) [E.164] is RECOMMENDED as the format of this Claim, for example, +1 (425) 555-1212 or +56 (2) 687 2400. If the phone number contains an extension, it is RECOMMENDED that the extension be represented using the [RFC 3966 (Schulzrinne, H., “The tel URI for Telephone Numbers,” December 2004.)](#RFC3966) [RFC3966] extension syntax, for example, +1 (604) 555-1234;ext=5678. |
| phone\_number\_verified | boolean | True if the End-User's phone number has been verified; otherwise false. When this Claim Value is true, this means that the OP took affirmative steps to ensure that this phone number was controlled by the End-User at the time the verification was performed. The means by which a phone number is verified is context-specific, and dependent upon the trust framework or contractual agreements within which the parties are operating. When true, the phone\_number Claim MUST be in E.164 format and any extensions MUST be represented in RFC 3966 format. |
| address | JSON object | End-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 the members defined in [Section 2.5.1 (Address Claim)](#address_claim). |
| updated\_at | number | Time the End-User's information was last updated. The time is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time. |

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

Following is a non-normative example of such a response:

{

"sub": "248289761001",

"name": "Jane Doe",

"given\_name": "Jane",

"family\_name": "Doe",

"preferred\_username": "j.doe",

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

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

}

The UserInfo Endpoint MUST return Claims in JSON format unless a different format was specified during Registration [[OpenID.Registration] (Sakimura, N., Bradley, J., and M. Jones, “OpenID Connect Dynamic Client Registration 1.0,” June 2013.)](#OpenID.Registration). 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 | JSON Web Token (JWT) |

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

### 2.5.1.  Address Claim

The Address Claim represents a physical mailing address. Implementations MAY return only a subset of the fields of an address, depending upon the information available and the End-User's privacy preferences. For example, the country and region might be returned without returning more fine-grained address information.

Implementations MAY return just the full address as a single string in the formatted sub-field, or they MAY return just the individual component fields using the other sub-fields, or they MAY return both. If both variants are returned, they SHOULD be describing the same address, with the formatted address indicating how the component fields are combined.

formatted

Full mailing address, formatted for display or use on a mailing label. This field MAY contain multiple lines, separated by newline characters.

street\_address

Full street address component, which MAY include house number, street name, Post Office Box, and multi-line extended street address information. This field MAY contain multiple lines, separated by newline characters.

locality

City or locality component.

region

State, province, prefecture or region component.

postal\_code

Zip code or postal code component.

country

Country name component.

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

### 2.5.2.  Claims Languages and Scripts

Human-readable Claim Values and Claim Values that reference human-readable values 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 are added to member names, delimited by a # character. For example, family\_name#ja-Kana-JP expresses the 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 family\_name#ja-Hani-JP. As another example, both website and website#de Claim Values might be returned, referencing a Web site in an unspecified language and a Web site in German.

Since Claim Names are case sensitive, it is strongly RECOMMENDED that language tag values used in Claim Names be spelled using the character case with which they are registered in the [IANA Language Subtag Registry (Internet Assigned Numbers Authority (IANA), “Language Subtag Registry,” 2005.)](#IANA.Language) [IANA.Language]. In particular, normally language names are spelled with lowercase characters, region names are spelled with uppercase characters, and scripts are spelled with mixed case characters. However, since BCP47 language tag values are case insensitive, implementations SHOULD interpret the language tag values supplied in a case insensitive manner.

Per the recommendations in BCP47, language tag values for Claims SHOULD only be as specific as necessary. For instance, using fr might be sufficient in many contexts, rather than fr-CA or fr-FR. Where possible, OPs SHOULD try to match requested Claim locales with Claims it has. For instance, if the Client asks for a Claim with a de (German) language tag and the OP has a value tagged with de-CH (Swiss German) and no generic German value, it would be appropriate for the OP to return the Swiss German value to the Client. (This intentionally moves as much of the complexity of language tag matching to the OP as possible, to simplify Clients.)

A claims\_locales request can be used to specify the preferred languages and scripts to use for the returned Claims.

When the OP determines, either through the claims\_locales parameter, or by other means, that the End-User and Client are requesting Claims in only one set of languages and scripts, it is RECOMMENDED that OPs return Claims without language tags when they employ this language and script. It is also RECOMMENDED that Clients be written in a manner that they can handle and utilize Claims using language tags.

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

### 2.5.3.  Claim Stability and Uniqueness

The sub (subject) and iss (issuer) Claims are the only Claims that a Client can rely upon as a stable identifier for the End-User, since the sub Claim MUST be locally unique and never reassigned within the Issuer for a particular End-User, as described in [Section 2.2 (ID Token)](#id_token). Therefore, the only guaranteed unique identifier for a given End-User is the combination of the iss Claim and the sub Claim.

All other Claims carry no such guarantees across different issuers in terms of stability over time or uniqueness across users, and Issuers are permitted to apply local restrictions and policies. For instance, an Issuer MAY re-use an email Claim value across different End-Users at different points in time, and the claimed email address for a given End-User MAY change over time. Therefore, other Claims such as email, phone\_number, and preferred\_username and MUST NOT be used as unique identifiers for the End-User.

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

### 3.  Self-Issued OpenID Provider

OpenID Connect supports Self-Issued OpenID Providers - personal OPs that issue self-signed ID Tokens. Self-Issued OPs use the special Issuer Identifier https://self-issued.me.

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

### 3.1.  Self-Issued OpenID Provider Discovery

If the input identifier for the discovery process contains the domain self-issued.me, dynamic discovery is not performed. Instead, then the following static configuration values are used:

{

"authorization\_endpoint":

"openid:",

"issuer":

"https://self-issued.me",

"scopes\_supported":

["openid", "profile", "email", "address", "phone"],

"response\_types\_supported":

["id\_token"],

"subject\_types\_supported":

["pairwise"],

"id\_token\_signing\_alg\_values\_supported":

["RS256"],

"request\_object\_signing\_alg\_values\_supported":

["none", "RS256"]

}

Note: The OpenID Foundation may consider hosting a site https://self-issued.me/ that returns the above static configuration file so that the Client would not need any special treatment for discovery of the Self-Issued OP.

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

### 3.2.  Self-Issued OpenID Provider Registration

When using a Self-Issued OP, the Client is deemed to have registered with the OP and obtained following Client Registration Response.

client\_id

redirect\_uri value of the Client.

client\_secret\_expires\_at

0

Note: The OpenID Foundation may consider hosting the (stateless) endpoint https://self-issued.me/registration/1.0/ that returns the response above so that the Client would not need to perform any special processing for registration of a Self-Issued OP.

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

### 3.2.1.  Providing Additional Registration Information

The registration request parameter can be used by the Client to provide information about itself to a Self-Issued OP that would normally be provided to an OP during Dynamic Client Registration. The value is a JSON object containing name/value pairs defined in Section 2.1 of the [OpenID Connect Dynamic Client Registration 1.0 (Sakimura, N., Bradley, J., and M. Jones, “OpenID Connect Dynamic Client Registration 1.0,” June 2013.)](#OpenID.Registration) [OpenID.Registration] specification. None of this information is REQUIRED by Self-Issued OPs, so the use of this parameter is OPTIONAL.

The registration parameter value is represented in an OAuth 2.0 request as UTF-8 encoded JSON (which ends up being form-urlencoded when passed as an OAuth parameter).

The Registration parameters that would typically be used in requests to Self-Issued OPs are policy\_uri, tos\_uri, and logo\_uri. If the Client uses more than one redirection URI, the redirect\_uris parameter would be used to register them.

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

### 3.3.  Self-Issued OpenID Provider Request

The Client sends the Authorization Request to the Authorization Endpoint with the following parameters:

response\_type

REQUIRED. Constant string value id\_token.

client\_id

REQUIRED. Client ID value for the Client, which in this case contains the redirect\_uri value of the Client. Since the Client's redirect\_uri URI value is communicated as the Client ID, a redirect\_uri parameter is NOT REQUIRED to also be included in the request.

scope

REQUIRED. scope parameter value, as defined in [Section 2.4 (Scope Values)](#scopes).

id\_token\_hint

OPTIONAL. Previously issued ID Token passed to the Authorization Server as a hint about the End-User's current or past authenticated session with the Client. This SHOULD be present when prompt=none is used. If the End-User identified by the ID Token is logged in or is logged in by the request, then the Authorization Server returns a positive response; otherwise, it SHOULD return a negative response.

registration

OPTIONAL. This parameter is used by the Client to provide information about itself to a Self-Issued OP that would normally be provided to an OP during Dynamic Client Registration, as specified in [Section 3.2.1 (Providing Additional Registration Information)](#SelfIssuedRegistrationRequest). The registration parameter SHOULD NOT be used when the OP is not a Self-Issued OP.

Other parameters MAY be sent. Note that all Claims are returned in the ID Token.

The entire URL MUST NOT exceed 2048 ASCII characters.

Following is a non-normative example (with line wraps within values for display purposes only):

HTTP/1.1 302 Found

Location: openid://

?response\_type=id\_token

&client\_id=https%3A%2F%2Fclient.example.org%2Fcb

&scope=openid%20profile

&state=af0ifjsldkj

&nonce=n-0S6\_WzA2Mj

registration=&%7B%22logo\_uri%22%3A%22https%3A%2F%2F

client.example.org%2Flogo.png%22%7D

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

### 3.4.  Self-Issued OpenID Provider Response

The self-issued OpenID Provider response is the same as the normal implicit flow response with the following refinements. Since it is an implicit flow response, the response parameters will be returned in the URL fragment component.

1. The iss (issuer) Claim Value is https://self-issued.me.
2. A sub\_jwk Claim is present, with its value being the public key value used to check the signature of the ID Token.
3. The sub (subject) Claim value is the base64url encoded SHA-256 hash of the concatenation of the octets of the UTF-8 representations of the base64url encoded key values in the sub\_jwk Claim. When the kty value is RSA, the key values n and e are concatenated in that order. When the kty value is EC, the key values crv, x, and y are concatenated in that order.
4. No Access Token is returned for accessing a UserInfo Endpoint, so all Claims returned MUST be in the ID Token.

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

### 3.5.  Self-Issued ID Token Validation

To validate the ID Token in the Authorization or Token Endpoint Response, the Client MUST do the following:

1. The Client MUST validate that the value of the iss (issuer) Claim is https://self-isued.me. If iss contains a different value, the ID Token is not Self-Issued, and instead it MUST be validated according to [Section 2.2.1 (ID Token Validation)](" \l "id.token.validation).
2. The Client MUST validate that the aud (audience) Claim contains the value of the redirect\_uri that the Client sent in the authentication request as an audience.
3. The Client MUST validate the signature of the ID Token according to [JWS (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](" \l "JWS) [JWS] using the algorithm specified in the alg parameter of the JWT header [[JWT] (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Token (JWT),” May 2013.)](" \l "JWT), using the key in the sub\_jwk Claim; the key is a bare key in JWK [[JWK] (Jones, M., “JSON Web Key (JWK),” May 2013.)](" \l "JWK) format (not an X.509 certificate value).
4. The alg value SHOULD be the default of RS256. It MAY also be ES256.
5. The Client MUST validate that the sub (subject) Claim value is the base64url encoded SHA-256 hash of the concatenation of the octets of the UTF-8 representations of the base64url encoded key values in the sub\_jwk Claim. When the kty value is RSA, the key values n and e are concatenated in that order. When the kty value is EC, the key values crv, x, and y are concatenated in that order.
6. The current time MUST be less than the value of the exp Claim (possibly allowing for some small leeway to account for clock skew).
7. The iat Claim can be used to reject tokens that were issued too far away from the current time, limiting the amount of time that nonces need to be stored to prevent attacks. The acceptable range is Client specific.
8. If a nonce value was sent in the Authorization Request, a nonce Claim MUST be present and its value of the checked to verify that it is the same value as the one that was sent in the Authorization Request. The Client SHOULD check the nonce value for replay attacks. The precise method for detecting replay attacks is Client specific.

The following is a non-normative example of a base64url decoded Self-Issued ID Token (with line wraps within values for display purposes only):

{

"iss": "https://self-issued.me",

"sub": "wBy8QvHbPzUnL0x63h13QqvUYcOur1X0cbQpPVRqX5k",

"aud": "https://client.example.org/cb",

"nonce": "n-0S6\_WzA2Mj",

"exp": 1311281970,

"iat": 1311280970,

"sub\_jwk": {

"kty":"RSA",

"n": "0vx7agoebGcQSuuPiLJXZptN9nndrQmbXEps2aiAFbWhM78LhWx

4cbbfAAtVT86zwu1RK7aPFFxuhDR1L6tSoc\_BJECPebWKRXjBZCiFV4n3oknjhMs

tn64tZ\_2W-5JsGY4Hc5n9yBXArwl93lqt7\_RN5w6Cf0h4QyQ5v-65YGjQR0\_FDW2

QvzqY368QQMicAtaSqzs8KJZgnYb9c7d0zgdAZHzu6qMQvRL5hajrn1n91CbOpbI

SD08qNLyrdkt-bFTWhAI4vMQFh6WeZu0fM4lFd2NcRwr3XPksINHaQ-G\_xBniIqb

w0Ls1jF44-csFCur-kEgU8awapJzKnqDKgw",

"e":"AQAB"

}

}

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

### 4.  Serializations

A request message MAY be serialized using one of the following methods:

1. Query String Serialization
2. Form Serialization

|  |
| --- |
| [**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] (Hors, A., Raggett, D., and I. Jacobs, “HTML 4.01 Specification,” December 1999.)](#W3C.REC-html401-19991224). Query String Serialization is typically used in HTTP GET requests. The same serialization method is also used when adding parameters to the fragment component of a URL.

Following is a non-normative example of this serialization (with line wraps within values for display purposes only):

GET /authorize?scope=openid

&response\_type=id\_token%20token

&client\_id=s6BhdRkqt3

&redirect\_uri=https%3A%2F%2Fclient.example.org%2Fcb HTTP/1.1

Host: server.example.com

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

### 4.2.  Form Serialization

Parameters and their values are Form Serialized by adding the parameter names and values to the entity body of the HTTP request using the application/x-www-form-urlencoded format as defined by [[W3C.REC‑html401‑19991224] (Hors, A., Raggett, D., and I. Jacobs, “HTML 4.01 Specification,” December 1999.)](#W3C.REC-html401-19991224). Form Serialization is typically used in HTTP POST requests.

Following is a non-normative example of this serialization (with line wraps within values for display purposes only):

POST /authorize HTTP/1.1

Host: server.example.com

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

scope=openid

&response\_type=id\_token%20token

&client\_id=s6BhdRkqt3

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

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

### 5.  String Operations

Processing some OpenID Connect messages requires comparing values in the messages to known values. For example, the Claim Names returned by the UserInfo Endpoint might be compared to specific Claim Names such as sub. Comparing Unicode strings, however, has significant security implications.

Therefore, comparisons between JSON strings and other Unicode strings MUST be performed as specified below:

1. Remove any JSON applied escaping to produce an array of Unicode code points.
2. [Unicode Normalization (Davis, M., Whistler, K., and M. Dürst, “Unicode Normalization Forms,” 09 2009.)](" \l "USA15) [USA15] MUST NOT be applied at any point to either the JSON string or to the string it is to be compared against.
3. Comparisons between the two strings MUST be performed as a Unicode code point to code point equality comparison.

In several places, this specification uses space delimited lists of strings. In all such cases, only the ASCII space character (0x20) MAY be used for this purpose.

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

### 6.  TLS Version

Whenever Transport Layer Security (TLS) is used by this specification, the appropriate version (or versions) of TLS will vary over time, based on the widespread deployment and known security vulnerabilities. At the time of this writing, TLS version 1.2 [[RFC5246] (Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” August 2008.)](#RFC5246) is the most recent version, but has a very limited deployment base and might not be readily available for implementation. TLS version 1.0 [[RFC2246] (Dierks, T. and C. Allen, “The TLS Protocol Version 1.0,” January 1999.)](#RFC2246) is the most widely deployed version and will provide the broadest interoperability.

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

### 7.  Implementation Considerations

This specification defines features used by Relying Parties using the OAuth implicit grant type. These Relying Parties MUST implement the features that are listed in this specification as being "REQUIRED" or are described with a "MUST".

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

### 7.1.  Discovery and Registration

Some OpenID Connect installations can use a pre-configured set of OpenID Providers and/or Relying Parties. In those cases, it might 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 (Sakimura, N., Bradley, J., Jones, M., and E. Jay, “OpenID Connect Discovery 1.0,” June 2013.)](#OpenID.Discovery) [OpenID.Discovery] and [OpenID Connect Dynamic Client Registration 1.0 (Sakimura, N., Bradley, J., and M. Jones, “OpenID Connect Dynamic Client Registration 1.0,” June 2013.)](#OpenID.Registration) [OpenID.Registration] specifications.

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

### 8.  Security Considerations

For security considerations other than those listed below, refer to the [OpenID Connect Messages 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Messages 1.0,” June 2013.)](#OpenID.Messages) [OpenID.Messages] and [OpenID Connect Standard 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Standard 1.0,” June 2013.)](#OpenID.Standard) [OpenID.Standard] specifications.

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

### 8.1.  TLS Requirements

Implementations MUST support TLS. Which version(s) ought to be implemented will vary over time, and depend on the widespread deployment and known security vulnerabilities at the time of implementation. At the time of this writing, TLS version 1.2 [[RFC5246] (Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” August 2008.)](#RFC5246) is the most recent version, but has very limited actual deployment, and might not be readily available in implementation toolkits. TLS version 1.0 [[RFC2246] (Dierks, T. and C. Allen, “The TLS Protocol Version 1.0,” January 1999.)](#RFC2246) is the most widely deployed version, and will give the broadest interoperability.

To protect against information disclosure and tampering, confidentiality protection MUST be applied using TLS with a ciphersuite that provides confidentiality and integrity protection.

Whenever TLS is used, a TLS server certificate check MUST be performed, per [RFC 6125 (Saint-Andre, P. and J. Hodges, “Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS),” March 2011.)](#RFC6125) [RFC6125].

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

### 9.  Privacy Considerations

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

### 9.1.  Personally Identifiable Information

The UserInfo Response typically contains Personally Identifiable Information (PII). As such, End-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\_uris.

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

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

### 9.2.  Data Access Monitoring

The Resource Server SHOULD make the UserInfo access log available to the End-User so that the End-User can monitor who accessed his data.

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

### 9.3.  Correlation

To protect the End-User from a possible correlation among Clients, the use of a Pairwise Pseudonymous Identifier (PPID) as the sub (subject) SHOULD be considered.

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

### 10.  IANA Considerations

This document makes no requests of IANA.

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

### 11.  References

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

### 11.1. Normative References

|  |  |
| --- | --- |
| **[E.164]** | International Telecommunication Union, “[E.164: The international public telecommunication numbering plan](http://www.itu.int/rec/T-REC-E.164-201011-I/en),” 2010. |
| **[IANA.Language]** | Internet Assigned Numbers Authority (IANA), “[Language Subtag Registry](http://www.iana.org/assignments/language-subtag-registry),” 2005. |
| **[ISO29115]** | International Organization for Standardization, “[ISO/IEC 29115:2013 -- Information technology - Security techniques - Entity authentication assurance framework](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45138),” ISO/IEC 29115, March 2013. |
| **[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. |
| **[ISO8601-2004]** | International Organization for Standardization, “ISO 8601:2004. Data elements and interchange formats - Information interchange - Representation of dates and times,” 2004. |
| **[JWA]** | Jones, M., “[JSON Web Algorithms (JWA)](http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms),” draft-ietf-jose-json-web-algorithms (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms-11)). |
| **[JWK]** | Jones, M., “[JSON Web Key (JWK)](http://tools.ietf.org/html/draft-ietf-jose-json-web-key),” draft-ietf-jose-json-web-key (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-jose-json-web-key-11)). |
| **[JWS]** | Jones, M., Bradley, J., and N. Sakimura, “[JSON Web Signature (JWS)](http://tools.ietf.org/html/draft-ietf-jose-json-web-signature),” draft-ietf-jose-json-web-signature (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-jose-json-web-signature-11)). |
| **[JWT]** | Jones, M., Bradley, J., and N. Sakimura, “[JSON Web Token (JWT)](http://tools.ietf.org/html/draft-ietf-oauth-json-web-token),” draft-ietf-oauth-json-web-token (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-08)). |
| **[OAuth.Responses]** | de Medeiros, B., Ed., Scurtescu, M., and P. Tarjan, “[OAuth 2.0 Multiple Response Type Encoding Practices](http://openid.net/specs/oauth-v2-multiple-response-types-1_0-08.html),” June 2013. |
| **[OpenID.Discovery]** | Sakimura, N., Bradley, J., Jones, M., and E. Jay, “[OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0-17.html),” June 2013. |
| **[OpenID.Messages]** | Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “[OpenID Connect Messages 1.0](http://openid.net/specs/openid-connect-messages-1_0-20.html),” June 2013. |
| **[OpenID.Registration]** | Sakimura, N., Bradley, J., and M. Jones, “[OpenID Connect Dynamic Client Registration 1.0](http://openid.net/specs/openid-connect-registration-1_0-19.html),” June 2013. |
| **[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_0-21.html),” June 2013. |
| **[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)). |
| **[RFC2246]** | [Dierks, T.](mailto:tdierks@certicom.com) and [C. Allen](mailto:callen@certicom.com), “[The TLS Protocol Version 1.0](http://tools.ietf.org/html/rfc2246),” RFC 2246, January 1999 ([TXT](http://www.rfc-editor.org/rfc/rfc2246.txt)). |
| **[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)). |
| **[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)). |
| **[RFC3966]** | Schulzrinne, H., “[The tel URI for Telephone Numbers](http://tools.ietf.org/html/rfc3966),” RFC 3966, December 2004 ([TXT](http://www.rfc-editor.org/rfc/rfc3966.txt)). |
| **[RFC3986]** | [Berners-Lee, T.](mailto:timbl@w3.org), [Fielding, R.](mailto:fielding@gbiv.com), and [L. Masinter](mailto:LMM@acm.org), “[Uniform Resource Identifier (URI): Generic Syntax](http://tools.ietf.org/html/rfc3986),” STD 66, RFC 3986, January 2005 ([TXT](http://www.rfc-editor.org/rfc/rfc3986.txt), [HTML](http://xml.resource.org/public/rfc/html/rfc3986.html), [XML](http://xml.resource.org/public/rfc/xml/rfc3986.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)). |
| **[RFC5246]** | Dierks, T. and E. Rescorla, “[The Transport Layer Security (TLS) Protocol Version 1.2](http://tools.ietf.org/html/rfc5246),” RFC 5246, August 2008 ([TXT](http://www.rfc-editor.org/rfc/rfc5246.txt)). |
| **[RFC5322]** | [Resnick, P., Ed.](mailto:presnick@qualcomm.com), “[Internet Message Format](http://tools.ietf.org/html/rfc5322),” RFC 5322, October 2008 ([TXT](http://www.rfc-editor.org/rfc/rfc5322.txt), [HTML](http://xml.resource.org/public/rfc/html/rfc5322.html), [XML](http://xml.resource.org/public/rfc/xml/rfc5322.xml)). |
| **[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)). |
| **[RFC6125]** | Saint-Andre, P. and J. Hodges, “[Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)](http://tools.ietf.org/html/rfc6125),” RFC 6125, March 2011 ([TXT](http://www.rfc-editor.org/rfc/rfc6125.txt)). |
| **[RFC6711]** | Johansson, L., “[An IANA Registry for Level of Assurance (LoA) Profiles](http://tools.ietf.org/html/rfc6711),” RFC 6711, August 2012 ([TXT](http://www.rfc-editor.org/rfc/rfc6711.txt)). |
| **[RFC6749]** | Hardt, D., “[The OAuth 2.0 Authorization Framework](http://tools.ietf.org/html/rfc6749),” RFC 6749, October 2012 ([TXT](http://www.rfc-editor.org/rfc/rfc6749.txt)). |
| **[RFC6750]** | Jones, M. and D. Hardt, “[The OAuth 2.0 Authorization Framework: Bearer Token Usage](http://tools.ietf.org/html/rfc6750),” RFC 6750, October 2012 ([TXT](http://www.rfc-editor.org/rfc/rfc6750.txt)). |
| **[USA15]** | [Davis, M.](mailto:markdavis@google.com), [Whistler, K.](mailto:ken@unicode.org), and M. Dürst, “Unicode Normalization Forms,” Unicode Standard Annex 15, 09 2009. |
| **[W3C.REC-html401-19991224]** | Hors, A., Raggett, D., 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) |

### 11.2. Informative References

|  |  |
| --- | --- |
| **[OpenID.Basic]** | Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, “[OpenID Connect Basic Client Profile 1.0](http://openid.net/specs/openid-connect-basic-1_0-28.html),” June 2013. |

|  |
| --- |
| [**TOC**](#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.

Naveen Agarwal (naa@google.com), Google

Casper Biering (cb@peercraft.com), Peercraft

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

Tim Bray (tbray@textuality.com), Google

Johnny Bufu (jbufu@janrain.com), Janrain

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

Pamela Dingle (pdingle@pingidentity.com), Ping Identity

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

Roland Hedberg (roland.hedberg@adm.umu.se), University of Umea

Ryo Ito (ryo.ito@mixi.co.jp), mixi, Inc.

Edmund Jay (ejay@mgi1.com), Illumila

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

Torsten Lodderstedt (t.lodderstedt@telekom.de), Deutsche Telekom

Nov Matake (nov@matake.jp), Independent

Chuck Mortimore (cmortimore@salesforce.com), Salesforce

Anthony Nadalin (tonynad@microsoft.com), Microsoft

Hideki Nara (hdknr@ic-tact.co.jp), Tact Communications

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

David Recordon (dr@fb.com), Facebook

Justin Richer (jricher@mitre.org), Mitre

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

Luke Shepard (lshepard@fb.com), Facebook

Andreas Akre Solberg (andreas.solberg@uninett.no), UNINET

Paul Tarjan (pt@fb.com), Facebook

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

### Appendix B.  Notices

Copyright (c) 2013 The OpenID Foundation.

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

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

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

### Appendix C.  Document History

[[ To be removed from the final specification ]]

-11

* Fixed #847 - Corrected type of updated\_at to number.
* Stated that redirect\_uri matches must be exact, with matching performed as described in Section 6.2.1 of RFC 3986 (Simple String Comparison).
* Fixed #854 - Clarified that the acr\_values values are in order of preference and that acr\_values requests the acr Claim as a Voluntary Claim.

-10

* Stated that sufficient entropy must be present in nonce values to prevent attackers from guessing values.
* Stated that the Authorization Server need not be listed as an audience of the ID Token when it is used as an id\_token\_hint value.
* Restricted the meaning of the azp (authorized party) Claim to simply be the single party to which the ID Token was issued.
* Changed from using the term "byte" to either "octet" or "character".
* Stated that the JWS Compact Serialization is always used for JWS data structures.

-09

* Fixed #825 - Replaced updated\_time, which used the RFC 3339 textual time format, with updated\_at, using the numeric time format used by iat, etc.
* Fixed #829 - Stated that additional scope values can be defined and used and that scope values that are not understood should be ignored.
* Fixed #831 - Stated that JWS and JWE header parameters used to communicate key values and key references should not be used in ID Tokens, since these are communicated in advance using Discovery and Registration parameters.
* Fixed #712 and #830 - Clarified the azp description and made azp multi-valued, like aud.

-08

* Fixed #802 - Clarified recommendations and responsibilities for producing and consuming Claims with and without language tags.
* Fixed #797 - Clarified the intended semantics of e-mail verification and that the precise verification rules are context-specific.
* Fixed #806 - Added phone\_number\_verified Claim.
* Fixed #800 - Specified that phone number extensions are to be represented using RFC 3966 extension syntax.
* Fixed #795 - Specified that e-mail addresses must conform to the RFC 5322 addr-spec syntax.
* Fixed #808 - Specified that phone numbers may be used as login\_hint values.
* Fixed #801 - Removed schema and id parameters to UserInfo Endpoint. Also fixed related issue #791 - Removed invalid\_schema error.
* Fixed #793, #796, and #799 - Allow name Claims to contain multiple space-separated names.
* Fixed #794 - Required picture to refer to an image file that is a picture of the End-User.
* Fixed #811 - Specify that language tag components should be spelled using the character cases registered in the IANA Language Subtag Registry.
* Fixed #812 - Clarified that language tag values used need not be unnecessarily specific.
* Fixed #816 - Changed "must understand" language to "MUST be ignored if not understood".

-07

* Fixed #711 - Awkward phrase "The following Claims are REQUIRED and OPTIONAL".
* Fixed #712 - "azp" definition clarification.
* Fixed #713 - Explicitly require "sub" claim to be returned from UserInfo endpoint.
* Fixed #716 - Client/server 2119 blurriness.
* Fixed #722 - Text on "id\_token\_hint" needs to be clarified.
* Fixed #718 - Text on re-encrypting should be clearer.
* Fixed #732 - Capitalize name of "Bearer" authentication scheme.
* Fixed #738 - Behavior when "openid" scope is omitted.
* Added Security Considerations section about TLS version requirements and usage.
* Removed language about clients that do not support TLS. Also removed language about supporting other transport-layer mechanisms with equivalent security to TLS.
* State that when any validations fail, any operations requiring the information that failed to correctly validate MUST be aborted and the information that failed to validate MUST NOT be used.
* Added id\_token\_hint parameter to Implicit, since it SHOULD be present when prompt=none is used.
* Removed vestigial encryption text that had been copied over from Standard, since encryption is not part of the Implicit profile.
* Fixed #742 - Added new ui\_locales parameter.
* Fixed #743 - Added claims\_locales parameter.
* Fixed #744 - Added max\_age parameter.
* Fixed #765 - Added new acr\_values parameter.
* Fixed #597 - Changed representation of omitted year in birthdate from 9999 to 0000.
* Fixed #739 - Added registration parameter and members of registration object.
* Deleted top-level policy\_url parameter from the Self-Issued case, since it is already a registration parameter member.
* Fixed #778 - Added registration parameter to Self-Issued request example.
* Fixed #782 - Changed uses of "\_url" in identifiers to "\_uri".
* Consolidated the x509\_encryption\_uri and jwk\_encryption\_uri parameters into a combined jwk\_uri parameter (per other changes related to #703).
* Fixed #786 - Changed the name of jwk\_uri to jwks\_uri.
* Clarified when the http scheme can and can not be used in redirect\_uri values.
* Stated that the azp Claim is only needed when the party requesting the ID Token is different than the audience of the ID Token.
* Use legal acr values in examples.
* Fixed #789 - Added amr (authentication methods references) Claim.

-06

* Fixed #637 removed requirement for hash of at\_token and code to be SHA2 in Section 2.1.2.1 and Section 5.2.
* Added Section 2.5 Access Token Validation.
* Fixed #620 - Update Section 2.2.5.1 to allow for other token types, but make bearer mandatory to support for implicit clients.
* Fixed #657 - Update Section 3.3 to say that the sub is sent as the kid if the id\_token is encrypted in the request.
* Added Implementation Considerations section.
* Fixed #698 - Inconsistent use of articles.
* Updated Scopes description.
* Added auth\_time definition to ID Token schema.
* Fixed #655 - Specify UTF-8 as encoding scheme whenever necessary.
* Renamed the user\_jwk Claim to sub\_jwk, paralleling the change from user\_id to sub.
* Defined the sub\_jwk claim.
* Clarified that the offline\_access scope value MUST NOT be used with the Implicit Client Profile.
* To remove ambiguity in the self-issued sub computation, changed the text "the concatenation of the key values" to "the concatenation of the bytes of the UTF-8 representations of the base64url encoded key values".
* Tracked JWK parameter name changes alg -> kty, mod -> n, exp -> e.

-05

* Fixed #687 - Inconsistency between user\_id and prn claims. The fix changed these names: user\_id -> sub, user\_id\_types\_supported -> subject\_types\_supported, user\_id\_type -> subject\_type, and prn -> sub.
* Fixed #689 - Track JWT change that allows JWTs to have multiple audiences.
* Fixed #660 - Clarified that returning the sub value from the UserInfo endpoint is mandatory.
* Fixed #636 - ID Token authorized party claim.
* Fixed #689 - Add caution about multiple audiences.
* Fixed #694 - Add login\_hint

-04

* Make it clear that nonce is REQUIRED for implicit
* RE #607 add example decoded id\_token for non self-issued.
* Fixed #614 - Discovery - 3.2 Distinguishing between signature and integrity parameters for HMAC algorithms. This fix tracks the parameter changes made to the JWE spec in draft-ietf-jose-json-web-encryption-06. It deletes the parameters {userinfo,id\_token}\_encrypted\_response\_int. It replaces the parameters {userinfo,id\_token,request\_object,token\_endpoint}\_algs\_supported with {userinfo,id\_token,request\_object,token\_endpoint}\_signing\_alg\_values\_supported and {userinfo,id\_token,request\_object,token\_endpoint}\_encryption\_{alg,enc}\_values\_supported.
* Fixed #666 - JWS signature validation vs. verification.
* Fixed #682 - Change remaining uses of "birthday" to "birthdate".
* Referenced OAuth 2.0 RFCs -- RFC 6749 and RFC 6750.

-03

* Defined means of using a self-issued OP

-02

* Added preferred\_username claim under profile scope
* Added ID Token section to describe required claims
* Added section on claim stability

-01

* Removed claims\_in\_id\_token scope value, per decision on June 15, 2012 special working group call

-00

* Initial version, based upon Basic Client specification version -17
* Renamed from Basic Client to Implicit Client, per issue #567
* Changed verified to email\_verified, per issue #564
* Removed Check ID Endpoint and added ID token signature verification text, per issue #570
* Changed client.example.com to client.example.org, per issue #251
* Added claims\_in\_id\_token scope definition to Basic and Implicit, per issue #594
* Use standards track version of JSON Web Token spec (draft-ietf-oauth-json-web-token)

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

### Authors' Addresses

|  |  |
| --- | --- |
|  | Nat Sakimura |
|  | Nomura Research Institute, Ltd. |
| **Email:** | [n-sakimura@nri.co.jp](mailto:n-sakimura@nri.co.jp) |
|  |  |
|  | John Bradley |
|  | Ping Identity |
| **Email:** | [ve7jtb@ve7jtb.com](mailto:ve7jtb@ve7jtb.com) |
|  |  |
|  | Michael B. Jones |
|  | Microsoft |
| **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 |
|  | Illumila |
| **Email:** | [ejay@mgi1.com](mailto:ejay@mgi1.com) |