Workgroup: connect
Internet-Draft: openid-connect-self-issued-v2-1_0
Published: 8 July 2021
Intended Status: Standards Track
Authors: K. Yasuda *Microsoft*
M. Jones *Microsoft*
T. Looker *Mattr*

Self-Issued OpenID Provider V2, draft 02

Abstract

This specification defines an extension of OpenID Connect to allow End-users to use OpenID Providers (OPs) that they control. Using Self-Issued OPs, End-users can authenticate themselves and present claims directly to the Relying Parties (RPs) without relying on a third-party Identity Provider.

Table of Contents

1. Introduction

Self-Issued OpenID Provider (Self-Issued OP) extends OpenID Connect to allow End-users to use OpenID Providers (OPs) that they control. Using Self-Issued OPs, End-users can authenticate themselves and present claims directly to the Relying Parties (RPs) without relying on a third-party Identity Provider.

The term Self-Issued comes from the fact that the End-users issue self-signed ID Tokens to prove validity of the identifiers and claims. This is a trust model different from that of the rest of OpenID Connect where OP is run by the third party who issues ID Tokens on behalf of the End-user upon End-user's consent. Therefore, Self-Issued OP comes with several limitations that are solved by introducing new mechanisms which require certain trade-offs.

First limitation is RP cannot be expected to be able to pre-establish trust relationships with every Self-Issued OP because there could be as many Self-Issued OPs as there are End-users. Usage of cryptographically verifiable identifiers is defined in this specification as a mechanism for RPs to trust SIOP without having to pre-establish a trust relationship between RP and OP as in a basic protocol of OpenID Connect.

Second limitation is Self-Issued OP cannot be trusted to assert all the claims about the End-user, because it is hosted by the End-user. Usage of cryptographically verifiable claims is defined in this specification as a mechanism for Self-Issued OP to present claims about the End-user asserted by the Claims Providers other than Self-Issued OP.

This specification defines how End-user provides ID Token and claims about the End-user to the RP using Self-Issued OP that is deployed on a device. This leads to another limitation that Self-Issued OPs cannot be expected to be able to host end-points.

Specifications for the few additional parameters and for the values of some parameters are defined in this section. Self-Issued OpenID Provider is an extension to OpenID Connect 1.0, and aspects not defined in this section must follow OpenID Connect 1.0.

Note: This specification replaces Self-Issued OpenID Connect Provider DID Profile v0.1 and was written as a working item of a liaison between Decentralized Identity Foundation and OpenID Foundation.

2. Use-cases

- Sudden or planned IdP unavailability IdPs can become unable to operate because they are destroyed as the result of a natural disaster such as hurricanes, tsunamis and earthquakes or as the result of a planned business decision. Using Self-Issued OP would enable End-users to authenticate themselves even when IdPs are temporarily or permanently unfunctionable.

- Authentication at the edge As more and more number of services and goods become digital and get connected to the Internet, the need for beneficiaries of those services and owners of those goods to be able to authenticate themselves before enjoying these services. Using Self-Issued OP would allow such authentications to happen at the edge, on End-user's device to achieve required efficiency and speed.

- Sharing credentials from several issuers in one transaction When End-users apply to open a banking account online, in most countries they are required to submit scanned versions of the required documents. These documents are usualy issued by different authorities, and hard to be verified in a digital form. When credentials are expressed as verified claims, using Self-Issued OP documents from various issuers can be shared in one transaction while allowing receiver to digitally verfy the validity of the shared document.

- Portability of the identities among providers As End-users sign up to use new online services, they have to create a new account and enter basic information about themselves once again. They could using one of the large Identity Providers, but than they need to remember which one they used for which service. This

experience could be improved using Self-Issued OP, if End-users can ask services to use identifier of their choice and present claims about themselves to provide their basic information to the services.

3. Scope

This document is scoped for a deployment model where Self-Issued OP is deployed on an End-user's device.

In scope:

- Discovery of Self-Issued OP

How an application on the End-user's edge device that is used to run Self-Issued OpenID Provider gets invoked upon receiving a Self-Issued OP request.

- Negotiation of supported metadata between RP and Self-Issued OP

How RP and Self-Issued OP negotiate supported metadata that necessary to process request and response such as supported signing algorithms, cryptographically verifiable identifiers, and credential formats. Negotiation is initiated by the RP and is included in the authorization request. The process is distinct from Dynamic Client Registration in OpenID Connect, since neither RP nor Self-Issued OP are expected to store information about the metadata supported by the counterparty. Negotiation establishes an ad hoc trust and is performed during every transaction even when RP and Self-Issued OP in question have transacted before.

- Usage of cryptographically verifiable identifiers as a way for RPs to identify the Authenticated user

Cryptographically verifiable identifiers include information about the key material used to sign the request and/or response. This way an entity receiving the request or response can verify whether the identifier is controlled by the other entity. First mechanism defined is the usage of jwk thumbprint, which is base64url encoded representation of the thumbprint of the key in the `sub_jwk` claim. Second mechanism defined is the usage of Decentralized Identifiers (DID). DID is a string that is used to obtain a DID document that contains information associated with the subject identified by a DID, including key material. Indirection layer between DID and DID Document allows controller of a DID to modify key material used to prove control over the identifier. DID Document is recorded on a system or network of some kind that can be a database of any kind including distributed ledgers and cloud storage.

- Usage of cryptographically verifiable claims

Mechanism for Self-Issued OPs to present claims using additional credential formats that enable the Holder to prove posession over the claims using cryptographic means. Additional credential formats include Verifiabel Presentation defined in [VC-DATA-MODEL].

Out of Scope:

- Claims issuance flow that defines how Self-Issued OP requests and receives claims from a Claims Provider acting in RP's capacity. Self-Issued OP can present those claims to the RP in Self-Issued OP response defined in this document.

4. Terms and definitions

Common terms in this document come from four primary sources: DID-CORE, VC-DATA, RFC6749 and OpenID-Core. In the case where a term has a definition that differs, the definition below is authoritative.

5. Abbreviations

- Self-Issued OP: Self-Issued OpenID Provider
- RP: Relying Party
- OP: OpenID Provider

6. Protocol Flow

Self-Issued OpenID Provider Request is an OpenID Connect Authentication Request that results in a Holder providing ID Token to the Relying Party through the Self-Issued OP. ID Token MAY include attested claims about the Holder.

!---

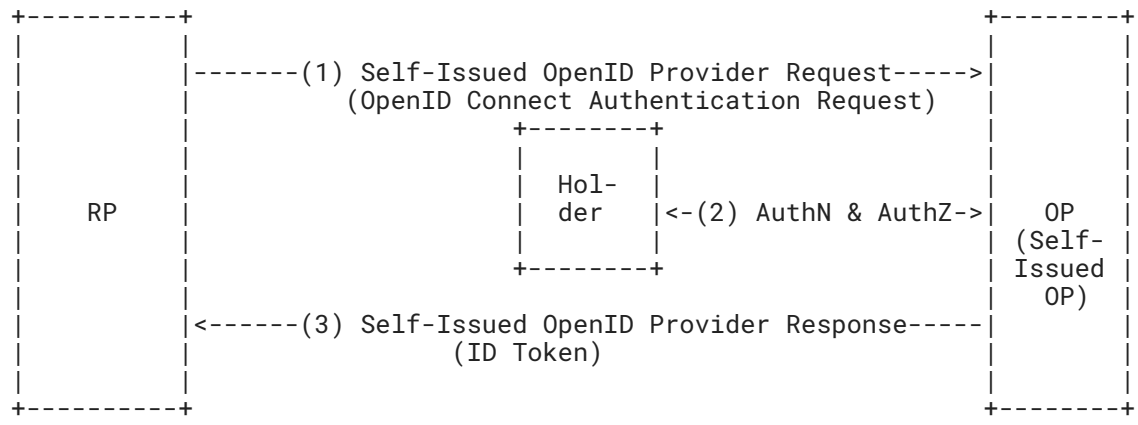

Figure 1: Self-Issued Op Protocol Flow

7. Discovery and Negotiation

8. Self-Issued OpenID Provider Discovery

Self-Issued OP MUST associate a custom schema `openid://` with itself. Relying Party MUST call `openid://` when sending a request to a Self-Issued OP.

Note: Custom schema is a mechanism offered by Mobile Operating System providers. If an application developer registers custom schema with the application, that application will be invoked when a request containing custom schema is received by the device.

Note: When more than one Self-issued OP with the same custom schema has been installed on one device, there could be confusion over which Self-Issued OP gets invoked.

9. Relying Party Registration

Relying Party must communicate which metadata parameters it supports. If Self-Issued OP and RP mutually support a compatible set of parameters, Self-Issued OP flow continues. If they do not, Self-Issued OP returns an error. Metadata parameters should preferrably be sent by reference as a URI, but when RP cannot host a webserver, they can be sent by value.

OpenID Connect defines the following negotiation parameters to enable Relying Party to provide information about itself to a Self-Issued OP that would normally be provided to an OP during Dynamic Client Registration:

`registration` OPTIONAL. This parameter enables RP Registration Metadata to be passed in a single, self-contained parameter. The value is a JSON object containing RP Registration Metadata values.

`registration_uri` OPTIONAL. This parameter enables RP Registration Metadata to be passed by reference, rather than by value. The request_uri value is a URL using the https scheme referencing a resource containing RP Negotiation Metadata values.

RP MUST use either of there parameters, but if one of these parameters is used, the other MUST NOT be used in the same request.

RP Negotiation metadata values are defined in Section 4.3 and Section 2.1 of the OpenID Connect Dynamic RP Registration 1.0 [OpenID.Registration] specification.

If Self-Issued OP supports the same parameters, Self-Issued OpenID Provider flow continues, if Self-Issued OP does not support, it returns an error.

If no error is returned, the RP must proceed as if it had obtained the following Client Registration Response:

- client_id
  - `redirect_uri` value of the Client.
- client*secret*expires_at
  - 0

Metadata parameters should preferably be sent by reference as a URI using `registration_uri` parameter, but when RP cannot host a webserver, metadata parameters should be sent by value using `registration` parameter.

`registration` and `registration_uri` parameters SHOULD NOT be used when the OP is not a Self-Issued OP.

9.1. Passing Relying Party Registration Metadata by Value

The `registration` SIOP Request parameter enables RP Registration Metadata to be passed in a single, self-contained parameter.

The registration parameter value is represented in an OAuth 2.0 request as a UTF-8 encoded JSON object (which ends up being form-urlencoded when passed as an OAuth parameter). When used in a Request Object value, per Section 6.1, the JSON object is used as the value of the registration member.

9.2. Passing Relying Party Registration Metadata by Reference

The `registration_uri` SIOP Request parameter enables RP Registration Metadata to be passed by reference.

This parameter is used identically to the request parameter, other than that the Relying Party registration metadata value is retrieved from the resource at the specified URL, rather than passed by value.

The contents of the resource referenced by the URL MUST be a RP Registration Metadata Object. The scheme used in the `registration_uri` value MUST be https. The request_uri value MUST be reachable by the Self-Issued OP, and SHOULD be reachable by the RP.

9.3. Relying Party Registration Metadata Values

This extension defines the following RP Registration Metadata values, used by the RP to provide information about itself to the Self-Issued OP:

- authorization_endpoint
  - REQUIRED. MUST include `openid:`, could also include additional custom schema.
- issuer
  - REQUIRED. MUST be `https://self-issued.me/v2`
- response*types*supported
  - REQUIRED. MUST be `id_token`
- scopes_supported
  - REQUIRED. A JSON array of strings representing supported scopes. Valid values include `openid`, `profile`, `email`, `address`, and phone.
- subject*types*supported
  - REQUIRED. A JSON array of strings representing supported subject types. Valid values include `pairwise` and `public`.
- subject*identifier*types_supported
  - REQUIRED. A JSON array of strings representing supported subject identifier types. Valid values include `jkt` and concrete did methods supported. DID methods supported must take the value of `Method Name` in Chapter 9 of did-spec-registries, such as `did:peer`:
- did*methods*supported
  - OPTIONAL. A JSON array of strings representing supported DID methods. Valid values include DID method names expressed following [DID] specification, for example `did:web`. RP can indicate support for any DID method by omitting `did_methods_supported`, While including `did` in ṣubject*identifier*types_supported'.
- credential*formats*supported
  - REQUIRED. A JSON array of strings representing supported credential formats. Valid values include `jwt`, `jwt_vc`, `jwt_vp`, `ldp_vc`, and `ldp_vp`.
- id*token*signing*alg*values_supported
  - REQUIRED. ID token signing alg values supported. Valid values include `RS256`, `ES256`, `ES256K`, and `EdDSA`.

- request*object*signing*alg*values_supported
  - REQUIRED. Request object signing alg values supported. Valid values include `none`, `RS256`, `ES256`, `ES256K`, and `EdDSA`.

Other registration parameters defined in [OpenID.Registration] could be used. Examples are explanatory parameters such as policy*uri, tos*uri, and logo*uri. If the RP uses more than one Redirection URI, the redirect*uris parameter would be used to register them. Finally, if the RP is requesting encrypted responses, it would typically use the jwks*uri, id*token*encrypted*response*alg and id*token*encrypted*response_enc parameters.

Registration parameter may include decentralized identifier of the RP.

The following is a non-normative example of RP Registration Metadata Values supported by Self-Issued OP:

```
{
 "authorization_endpoint":
  "openid:",
 "issuer":
  "https://self-issued.me/v2",
 "response_types_supported":
  ["id_token"],
 "scopes_supported":
  ["openid", "profile", "email", "address", "phone"],
 "subject_types_supported":
  ["pairwise"],
 "subject_identifier_types_supported":
  ["did:web:", "did:ion:"],
  "credential_formats_supported":
  ["jwt","jwt_vp"],
  "id_token_signing_alg_values_supported":
  ["ES256", "ES256K"],
 "request_object_signing_alg_values_supported":
  ["ES256", "ES256K"]
}
```

9.3.1. Sub Types

A sub type is used by Self-Issued OP to advertise which types of identifiers are supported for the `sub` claim. Two types are defined by this specification:

- jkt
  - JWK Thumbprint Subject sub type. When this subject sub type is used, the `sub` claim value MUST be the base64url encoded representation of the thumbprint of the key in the `sub_jwk` claim. [RFC7638]
- did
  - Decentralized sub type. When this sub type is used, the `sub` value MUST be a DID defined in [DID-CORE].

NOTE: Consider adding a subject type for OpenID Connect Federation entity statements.

9.4. Relying Party Registration Metadata Error Response

This extension defines the following error codes that MUST be returned when Self-Issued OP does not support all of the Relying Party Registration metadata values received from the Relying Party in the registration parameter:

- did*methods*not_supported

    - The Self-Issued OP does not support all of the DID methods included in `did_methods_supported` parameter.
- subject*identifier*types*not*supported
    - The Self-Issued OP does not support all of the subject identifier types included in `subject_identifier_types_supported` parameter.
- credential*formats*not_supported
    - The Self-Issued OP does not support all of the credential formats included in `credential_formats_supported` parameter.
- value*not*supported
    - The Self-Issued OP does not support more than one of the RP Registration Metadata values defined in Section 4.3. When not supported metadata values are DID methods, subject identifier types, or credential formats, more specific error message must be used.
- invalid*registration*uri
    - The registration_uri in the Self-Issued OpenID Provider request returns an error or contains invalid data.
- invalid*registration*object
    - The registration parameter contains an invalid RP Registration Metadata Object.

Error response must be made in the same manner as defined in Section 3.1.2.6.

10. Identifier Portability and Verifiable Presentation Support

11. Self-Issued OpenID Provider Request

The RP sends the Authentication Request to the Authorization Endpoint with the following parameters:

- scope
    - REQUIRED. `scope` parameter value, as specified in Section 3.1.2.
- 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 RP.
- redirect_uri
    - REQUIRED. MUST equal to `client_id` value. MUST be included for compatibility reasons.
- id*token*hint
    - OPTIONAL. id*token*hint parameter value, as specified in Section 3.1.2. If the ID Token is encrypted to the Self-Issued OP, the sub (subject) of the signed ID Token MUST be sent as the kid (Key ID) of the JWE.

- claims
  - OPTIONAL. claims parameter value, as specified in Section 5.5.
- registration
  - OPTIONAL. This parameter is used by the RP to provide information about itself to a Self-Issued OP that would normally be provided to an OP during Dynamic RP Registration, as specified in Section 2.2.1.
- registration_uri
  - OPTIONAL. This parameter is used by the RP to provide information about itself to a Self-Issued OP that would normally be provided to an OP during Dynamic RP Registration, as specified in Section 2.2.2.
- request
  - OPTIONAL. Request Object value, as specified in Section 6.1. The Request Object MAY be encrypted to the Self-Issued OP by the RP. In this case, the sub (subject) of a previously issued ID Token for this RP MUST be sent as the kid (Key ID) of the JWE.
- request_uri
  - OPTIONAL. URL where Request Object value can be retrieved from, as specified in Section 6.2.

When `request` or `reques_uri` parameters are NOT present, `registration` or `registration_uri` parameters MUST be present in the request. When `request` or `reques_uri` parameters are present, `registration` or `registration_uri` parameters MUST be included in either of those parameters.

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

The entire URL MUST NOT exceed 2048 ASCII characters.

The following is a non-normative example HTTP 302 redirect response by the RP, which triggers the User Agent to make an Authentication Request to the Self-Issued OP (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
  &redirect_uri=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
```

12. Self-Issued OpenID Provider Response

Self-Issued OpenID Provider Response is returned when Self-Issued OP supports all of the Relying Party Registration metadata values received from the Relying Party in the registration parameter. If even one of the Relying Party Registration Metadata Values is not supported, Self-Issued OP MUST return an error according to Section 4.4.

This extension defines the following claims to be included in the ID token for use in Self-Issued OpenID Provider Responses:

- sub
  - REQUIRED. Subject identifier value, represented by a URI. When sub type is `jkt`, the value is the base64url encoded representation of the thumbprint of the key in the `sub_jwk` Claim. When sub type is `did`, the value is a decentralized identifier. The thumbprint value is computed as the SHA-256 hash of the octets of the UTF-8 representation of a JWK constructed containing only the REQUIRED members to represent the key, with the member names sorted into lexicographic order, and with no white space or line breaks. For instance, when the kty value is RSA, the member names e, kty, and n are the ones present in the constructed JWK used in the thumbprint computation and appear in that order; when the kty value is EC, the member names crv, kty, x, and y are present in that order. Note that this thumbprint calculation is the same as that defined in the JWK Thumbprint [RFC7638] specification.
- sub_jwk
  - REQUIRED. a secure binding between the subject of the verifiable credential and the subject identifier (and related keys) of the holder who creates the presentation. When subr type is `jkt`, the key is a bare key in JWK [JWK] format (not an X.509 certificate value). When sub type is `did`, sub*jwk MUST contain a kid that is a DID URL referring to the verification method in the Self-Issued OP's DID Document that can be used to verify the JWS of the id*token directly or indirectly. The sub_jwk value is a JSON object. Use of the `sub_jwk` Claim is NOT RECOMMENDED when the OP is not Self-Issued.
- vp
  - OPTIONAL. A JSON object, that represents a JWT verifiable presentation, following W3C Verifiable Credentials Specification [VC-DATA-MODEL]. Verifiable Credentials must be embedded in the Verifiable Presentation following W3C Verifiable Credentials Specification [VC-DATA-MODEL]

Verifiable Presentation is data derived from one or more Verifiable Credentials, issued by one or more issuers, that is shared with a specific verifier. Verifiable Credential is a set of one or more claims made by an issuer.

Self-Issued OP may present credentials to the RP using Verifiable Presentation credential format by including it in the `vp` claim inside the ID token.

Whether the Self-Issued OP is a mobile client or a web client, 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, unless a different Response Mode was specified.

1. The `iss` (issuer) Claim Value is `https://self-issued.me/v2`.
2. A `sub_jwk` Claim is present, with its value being the public key used to check the signature of the ID Token.
3. The `sub` (subject) Claim value is either the base64url encoded representation of the thumbprint of the key in the `sub_jwk` Claim or a decentralized identifier.
4. No Access Token is returned for accessing a UserInfo Endpoint, so all Claims returned MUST be in the ID Token.

13. Self-Issued ID Token Validation

To validate the ID Token received, the RP MUST do the following:

1. The Relying Party (RP) MUST validate that the value of the `iss` (issuer) Claim is `https://self-issued.me`. If iss contains a different value, the ID Token is not Self-Issued, and instead it MUST be validated according to Section 3.1.3.

2. The RP MUST validate that the `aud` (audience) Claim contains the value of the `redirect_uri` that the RP sent in the Authentication Request as an audience.

3. The RP MUST validate the signature of the ID Token. When sub type is`jkt`, validation is done according to JWS [JWS] using the algorithm specified in the alg Header Parameter of the JOSE Header, using the key in the `sub_jwk` Claim. When sub type is`did`, vvalidation is done using the key derived as a result of DID Resolution as defined in [DID-CORE]. The key is a bare key in JWK format (not an X.509 certificate value) when sub type is`jkt` or may be another key format when sub type is `did`.

4. Default `alg` value is RS256. It MAY also be ES256, ES256K or EdDSA.

5. The RP MUST validate that the`sub` claim is bound to the `sub_jwk` value. When sub type is`jkt`, the RP MUST validate that the sub Claim value is the base64url encoded representation of the thumbprint of the key in the `sub_jwk` Claim, as specified in Section 6. When sub type is `did`, the RP MUST validate that the `kid` of the `sub_jwk` claim matches the verification method from the DID Document that is obtained by resolving decentralized identifier included in `sub` claim.

6. The current time MUST be before the time represented by the `exp` Claim (possibly allowing for some small leeway to account for clock skew). 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 RP specific.

7. If a `nonce` value was sent in the Authentication Request, a `nonce` Claim MUST be present and its value checked to verify that it is the same value as the one that was sent in the Authentication Request. The RP SHOULD check the `nonce`value for replay attacks. The precise method for detecting replay attacks is RP 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/v2",
 "sub": "NzbLsXh8uDCcd-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs",
 "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"
  },
  "vp": {
   "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
   ],
   "type": ["VerifiablePresentation"],
   "verifiableCredential": ["..."]
  }
}
```

14. References

14.1. Normative References

- [DID-CORE] https://github.com/w3c/did-core (not yet a ratified draft)
- [VC-DATA] https://www.w3.org/TR/vc-data-model/
- [RFC6749] https://tools.ietf.org/html/rfc6749
- [RFC6750] https://tools.ietf.org/html/rfc6750
- [OpenID.Core] https://openid.net/specs/openid-connect-core-1_0.html
- [RFC7638] https://tools.ietf.org/html/rfc7638
- [OpenID.Registration] https://openid.net/specs/openid-connect-registration-1_0.html
- [did-spec-registries] https://w3c.github.io/did-spec-registries/#did-methods

14.2. Non-Normative References

- [draft-jones-self*issued*identifier] https://bitbucket.org/openid/connect/src/master/SIOP/draft-jones-self_issued_identifier.md
- [siop-requirements] https://bitbucket.org/openid/connect/src/master/SIOP/siop-requirements.md

15. Relationships to other documents

The scope of this draft was an extention to OpenID Connect Chapter 7 Self-Issued OpenID Provider. However, some sections of it could become applicable more generally to the entire OpenID Connect specification.

16. IANA Considerations

TBD

17. Notices

18. Document History

- 01
  - Version proposed for working group adoption

Authors' Addresses

Kristina Yasuda
Microsoft
Email: kristina.yasuda@microsoft.com

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

Tobias Looker
Mattr
Email: tobias.looker@mattr.global