Draft

N. Sakimura
NRI
J. Bradley
Ping Identity
M. Jones
Microsoft
E. Jay
Illumila

October 15, 2013

TOC

OpenID Connect Discovery 1.0 - draft 18

Abstract

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It enables 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.

This specification provides a mechanism for the OpenID Connect Client to discover the End-User's OpenID Provider as well as the necessary endpoints used by the OpenID Connect protocol suite.

Table of Contents

1. Introduction

TOC

In order for an OpenID Client to utilize OpenID Connect services for an End-User, the Client needs to know where the OpenID Provider is. OpenID Connect uses **WebFinger** [RFC7033] to locate the OpenID Provider for an End-User.

Once an OpenID Provider is identified, the endpoint and other configuration information for that OP is retrieved from a well-known location as a JSON document.

1.1. Requirements Notation and Conventions

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **RFC 2119** [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)** [JWS] and **JSON Web Encryption (JWE)** [JWE] data structures in this specification utilize the JWS Compact Serialization or the JWE Compact Serialization; the JWS JSON Serialization and the JWE JSON Serialization are not used.

1.2. Terminology

TOC

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** [RFC6749], and the terms defined by **OpenID Connect Core 1.0** [OpenID.Core].

This specification also defines the following terms:

Resource
: Entity that is the target of a request in WebFinger.

Host
: Server where a WebFinger service is hosted.

Identifier
: Value that uniquely characterizes an Entity in a specific context.
Note: this document defines various kinds of Identifiers, designed for use in different contexts. Examples include URLs using the `https` scheme and e-mail addresses.

IMPORTANT NOTE TO READERS: The terminology definitions in this section are a normative portion of this specification, imposing requirements upon implementations. All the capitalized words in the text of this specification, such as "Identifier", reference these defined terms. Whenever the reader encounters them, their definitions found in this section must be followed.

2. OpenID Provider Discovery

TOC

OpenID Provider discovery is OPTIONAL; if a Relying Party knows the OP information through an out-of-band mechanism, they can skip this step and proceed to **Section 4**.

Provider discovery requires the following information to make a discovery request:

resource
: Identifier of the target End-User that is the subject of the discovery request.

host
: Server where a WebFinger service is hosted.

rel
: URI identifying the type of service whose location is requested.

OpenID Connect uses the following discoverable `rel` value in **WebFinger** [RFC7033]:

Rel Type	URI
OpenID Connect Issuer	http://openid.net/specs/connect/1.0/issuer

3-1

To start discovery of OpenID endpoints, the End-User supplies an Identifier to the Client or Relying Party. The Client applies the normalization rules to the Identifier to determine the Resource and Host. Then it makes an HTTPS `GET` request to the Host's **WebFinger** [RFC7033] endpoint with the `resource` and `rel` parameters to obtain the location of the requested service.

3-2

The Issuer MUST be returned in the response. This includes a URI scheme (which MUST be `https`), a Host, and OPTIONALLY, a port.

2.1. Identifier Normalization

TOC

The purpose of normalization is to determine a normalized Resource and Host from the user input Identifier. This is then used as input to WebFinger to discover the Issuer.

The user input Identifier SHOULD be a URL or URI relative reference defined in **RFC 3986** [RFC3986]. The user input Identifier MUST include the authority component.

Note: A URI relative reference includes a string that looks like an e-mail address in the form of `userinfo@host`. This is a valid authority component of a URI but excludes various possible extra strings allowed in `addr-spec` syntax of **RFC 5322** [RFC5322].

The Identifier normalization rules MAY be extended by additional specifications to enable other identifier types such as telephone numbers or **XRIs** [XRI_Syntax_2.0] to also be used.

2.1.1. User Input Identifier Types

TOC

A user input Identifier can be categorized into the following types, which require different normalization processes:

1. User input Identifiers starting with the **XRI** [XRI_Syntax_2.0] global context symbols ('=','@', and '!') are RESERVED. Processing of these identifiers is out of scope for this specification.
2. All other user input Identifiers MUST be treated as a URI in one of the forms `scheme`

`"://" authority path-abempty [ "?" query ] [ "#" fragment ]` or `authority path-abempty [ "?" query ] [ "#" fragment ]` or `scheme ":" path-rootless` per **RFC 3986** [RFC3986].

Note: The user input Identifier MAY be in the form of `userinfo@host`. For the End-User, this would normally be perceived as being an e-mail address. However, it is also a valid userpart "@" host section of an `acct` URI **[I-D.ietf-appsawg-acct-uri]**, and this specification treats it such as to exclude various extra strings allowed in `addr-spec` of **RFC 5322** [RFC5322].

2.1.2. Normalization Steps

TOC

A string of any other type is interpreted as a URI in one of the forms `scheme "://" authority path-abempty [ "?" query ] [ "#" fragment ]` or `authority path-abempty [ "?" query ] [ "#" fragment ]` or `scheme ":" path-rootless` per **RFC 3986** [RFC3986] and is normalized according to the following rules:

1. If the user input Identifier does not have an **RFC 3986** [RFC3986] scheme portion, the string is interpreted as `[userinfo "@"] host [":" port] path-abempty [ "?" query ] [ "#" fragment ]` per **RFC 3986** [RFC3986].
2. If the userinfo component is present and all of the scheme component, path component, query component, and port component are empty, the `acct` scheme is assumed. In this case, the normalized URI is formed by prefixing `acct:` to the string as the scheme. Per **The 'acct' URI Scheme** [I-D.ietf-appsawg-acct-uri], if there is an at-sign character ('@') in the userinfo component, it needs to be percent-encoded as described in **RFC 3986** [RFC3986].
3. For all other inputs without a scheme portion, the `https` scheme is assumed, and the normalized URI is formed by prefixing `https://` to the string as the scheme.
4. When the input contains an explicit scheme such as `acct` that matches the RFC 3986 `scheme ":" path-rootless` syntax, no input normalization is performed.
5. If the resulting URI contains a fragment portion, it MUST be stripped off together with the fragment delimiter character "#".

The **WebFinger** [RFC7033] Resource in this case is the resulting URI, and the WebFinger Host is the authority component.

Note: Since the definition of `authority` in **RFC 3986** [RFC3986] is `[ userinfo "@" ] host [ ":" port ]`, it is legal to have a user input identifier like `userinfo@host:port`, e.g., `alice@example.com:8080`.

2.2. Non-Normative Examples

TOC

2.2.1. User Input Using E-Mail Address Syntax

TOC

To find the Issuer for the given user input in the form of an e-mail address `joe@example.com`, the WebFinger parameters are as follows:

WebFinger Parameter	Value
resource	acct:joe@example.com
host	example.com
rel	http://openid.net/specs/connect/1.0/issuer

Note that in this case, the `acct:` scheme **[I-D.ietf-appsawg-acct-uri]** is prepended to the Identifier.

Following the WebFinger specification, the Client would make the following request to get the discovery information (with line wraps within lines for display purposes only):

```
GET /.well-known/webfinger
  ?resource=acct%3Ajoe%40example.com
  &rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
  HTTP/1.1
Host: example.com

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

{
 "subject": "acct:joe@example.com",
 "links":
  [
   {
    "rel": "http://openid.net/specs/connect/1.0/issuer",
    "href": "https://server.example.com"
   }
  ]
}
```

2.2.2. User Input Using URL Syntax

TOC

To find the Issuer for the given URL, `https://example.com/joe`, the WebFinger parameters are as follows:

WebFinger Parameter	Value
resource	https://example.com/joe
host	example.com
rel	http://openid.net/specs/connect/1.0/issuer

Following the WebFinger specification, the Client would make the following request to get the discovery information (with line wraps within lines for display purposes only):

```
GET /.well-known/webfinger
  ?resource=https%3A%2F%2Fexample.com%2Fjoe
  &rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
  HTTP/1.1
Host: example.com

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

{
 "subject": "https://example.com/joe",
 "links":
  [
```

```
   {
    "rel": "http://openid.net/specs/connect/1.0/issuer",
    "href": "https://server.example.com"
   }
  ]
}
```

2.2.3. User Input Using Hostname and Port Syntax

TOC

If the user input is in the form of `host:port`, e.g., example.com:8080, then it is assumed as the authority component of the URL.

To find the Issuer for the given hostname, `example.com:8080`, the WebFinger parameters are as follows:

WebFinger Parameter	Value
resource	https://example.com:8080/
host	example.com:8080
rel	http://openid.net/specs/connect/1.0/issuer

Following the WebFinger specification, the Client would make the following request to get the discovery information (with line wraps within lines for display purposes only):

```
GET /.well-known/webfinger
  ?resource=https%3A%2F%2Fexample.com%3A8080%2F
  &rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
  HTTP/1.1
Host: example.com:8080

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

{
 "subject": "https://example.com:8080/",
 "links":
  [
   {
    "rel": "http://openid.net/specs/connect/1.0/issuer",
    "href": "https://server.example.com"
   }
  ]
}
```

2.2.4. User Input Using "acct" URI Syntax

TOC

To find the Issuer for the given user input in the form of an account URI `acct:juliet%40capulet.example@shoppingsite.example.com`, the WebFinger parameters are as follows:

WebFinger Parameter	Value
resource	acct:juliet%40capulet.example@shoppingsite.example.com

host	shoppingsite.example.com
rel	http://openid.net/specs/connect/1.0/issuer

Following the WebFinger specification, the Client would make the following request to get the discovery information (with line wraps within lines for display purposes only):

```
GET /.well-known/webfinger
  ?resource=acct%3Ajuliet%2540capulet.example%40shoppingsite.example.com
  &rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer
  HTTP/1.1
Host: shoppingsite.example.com

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

{
 "subject": "acct:juliet%40capulet.example@shoppingsite.example.com",
 "links":
  [
   {
    "rel": "http://openid.net/specs/connect/1.0/issuer",
    "href": "https://server.example.com"
   }
  ]
}
```

Note: It is common for sites to use e-mail addresses as local identifiers for accounts at those sites, even though the domain in the e-mail address one controlled by the site. For instance, the site `example.org` might have a local account named `joe@example.com`. As of the time of this writing, a discussion is ongoing among WebFinger contributors about the syntax that should be used when discovering information about such accounts with WebFinger. The current thinking seems to be that such accounts would be represented by quoting the '@' character in the userinfo portion of the account identifier when constructing the `acct:` URI representing the account. Such an example is `acct:joe%40example.com@example.org`. In a future version of this specification, it is possible that normalization rules will be defined allowing End-Users to input values like `joe@example.com@example.org` to initiate discovery on such accounts.

7-1

---

3. OpenID Provider Metadata

TOC

OpenID Providers have metadata describing their configuration. These OpenID Provider Metadata values are used by OpenID Connect:

issuer
: REQUIRED. URL using the `https` scheme with no query or fragment component that the OP asserts as its Issuer Identifier.

authorization_endpoint
: OPTIONAL. URL of the OP's Authentication and Authorization Endpoint **[OpenID.Core]**.

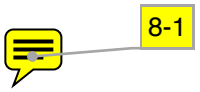
token_endpoint
: OPTIONAL. URL of the OP's OAuth 2.0 Token Endpoint **[OpenID.Core]**.

userinfo_endpoint
: RECOMMENDED. URL of the OP's UserInfo Endpoint **[OpenID.Core]**. This URL MUST use the `https` scheme and MAY contain port, path, and query parameter components.

jwks_uri
: REQUIRED. URL of the OP's JSON Web Key Set **[JWK]** document. This contains the

signing key(s) the Client uses to validate signatures from the OP. The JWK Set MAY also contain the Server's encryption key(s), which are used by Clients to encrypt requests to the Server. When both signing and encryption keys are made available, a `use` (Key Use) parameter value is REQUIRED for all keys in the document to indicate each key's intended usage.

registration_endpoint
: RECOMMENDED. URL of the OP's Dynamic Client Registration Endpoint **[OpenID.Registration]**.



scopes_supported
: RECOMMENDED. JSON array containing a list of the **OAuth 2.0** [RFC6749] scope values that this server supports. The server MUST support the `openid` scope value.

response_types_supported
: REQUIRED. JSON array containing a list of the OAuth 2.0 `response_type` values that this server supports. The server MUST support the `code`, `id_token`, and the `token id_token` response type values.

grant_types_supported
: OPTIONAL. JSON array containing a list of the OAuth 2.0 grant type values that this server supports. The server MUST support the `authorization_code` and `implicit` grant type values and MAY support the `urn:ietf:params:oauth:grant-type:jwt-bearer` grant type defined in **JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants** [OAuth.JWT]. If omitted, the default value is `["authorization_code", "implicit"]`.

acr_values_supported
: OPTIONAL. JSON array containing a list of the Authentication Context Class References that this server supports.

subject_types_supported
: REQUIRED. JSON array containing a list of the subject identifier types that this server supports. Valid types include `pairwise` and `public`.

id_token_signing_alg_values_supported
: REQUIRED. JSON array containing a list of the JWS signing algorithms (`alg` values) supported by the Authorization Server for the ID Token to encode the Claims in a JWT **[JWT]**. The algorithm `RS256` MUST be included. The value `none` MAY be supported, but MUST only be used with the Authorization Code Flow.

id_token_encryption_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWE encryption algorithms (`alg` values) supported by the Authorization Server for the ID Token to encode the Claims in a JWT **[JWT]**.

id_token_encryption_enc_values_supported
: OPTIONAL. JSON array containing a list of the JWE encryption algorithms (`enc` values) supported by the Authorization Server for the ID Token to encode the Claims in a JWT **[JWT]**.

userinfo_signing_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWS **[JWS]** signing algorithms (`alg` values) **[JWA]** supported by the UserInfo Endpoint to encode the Claims in a JWT **[JWT]**. The value `none` MAY be included.

userinfo_encryption_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWE **[JWE]** encryption algorithms (`alg` values) **[JWA]** supported by the UserInfo Endpoint to encode the Claims in a JWT **[JWT]**.

userinfo_encryption_enc_values_supported
: OPTIONAL. JSON array containing a list of the JWE encryption algorithms (`enc` values) **[JWA]** supported by the UserInfo Endpoint to encode the Claims in a JWT **[JWT]**.

request_object_signing_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWS signing algorithms (`alg` values) supported by the Authorization Server for the Request Object described in Section 5.1 of **OpenID Connect Core 1.0** [OpenID.Core]. These algorithms are used both when the Request Object is passed by value (using the `request` parameter) and when it is

passed by reference (using the `request_uri` parameter). Servers SHOULD support `none` and `RS256`.

request_object_encryption_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWE encryption algorithms (`alg` values) supported by the Authorization Server for the Request Object described in Section 5.1 of **OpenID Connect Core 1.0** [OpenID.Core]. These algorithms are used both when the Request Object is passed by value and when it is passed by reference.

request_object_encryption_enc_values_supported
: OPTIONAL. JSON array containing a list of the JWE encryption algorithms (`enc` values) supported by the Authorization Server for the Request Object described in Section 5.1 of **OpenID Connect Core 1.0** [OpenID.Core]. These algorithms are used both when the Request Object is passed by value and when it is passed by reference.

token_endpoint_auth_methods_supported
: OPTIONAL. JSON array containing a list of authentication methods supported by this Token Endpoint. The options are `client_secret_post`, `client_secret_basic`, `client_secret_jwt`, and `private_key_jwt`, as described in Section 8 of **OpenID Connect Core 1.0** [OpenID.Core]. Other authentication methods MAY be defined by extensions. If omitted, the default is `client_secret_basic` -- the HTTP Basic Authentication Scheme specified in Section 2.3.1 of **OAuth 2.0** [RFC6749].

token_endpoint_auth_signing_alg_values_supported
: OPTIONAL. JSON array containing a list of the JWS signing algorithms (`alg` values) supported by the Token Endpoint for the signature on the JWT **[JWT]** used to authenticate the Client at the Token Endpoint for the `private_key_jwt` and `client_secret_jwt` authentication methods. Servers SHOULD support `RS256`. The value `none` MUST NOT be used.

display_values_supported
: OPTIONAL. JSON array containing a list of the `display` parameter values that the OpenID Provider supports. These values are described in Section 2.1.2.1 of **OpenID Connect Core 1.0** [OpenID.Core].

claim_types_supported
: OPTIONAL. JSON array containing a list of the Claim Types that the OpenID Provider supports. These Claim Types are described in Section 4.6 of **OpenID Connect Core 1.0** [OpenID.Core]. Values defined by this specification are `normal`, `aggregated`, and `distributed`. If not specified, the implementation supports only `normal` Claims.

claims_supported
: RECOMMENDED. JSON array containing a list of the Claim Names of the Claims that the OpenID Provider MAY be able to supply values for. Note that for privacy or other reasons, this might not be an exhaustive list.

service_documentation
: OPTIONAL. URL of a page containing human-readable information that developers might want or need to know when using the OpenID Provider. In particular, if the OpenID Provider does not support Dynamic Client Registration, then information on how to register Clients needs to be provided in this documentation.

claims_locales_supported
: OPTIONAL. Languages and scripts supported for values in Claims being returned, represented as a JSON array of **BCP47** [RFC5646] language tag values. Not all languages and scripts are necessarily supported for all Claim values.

ui_locales_supported
: OPTIONAL. Languages and scripts supported for the user interface, represented as a JSON array of **BCP47** [RFC5646] language tag values.

claims_parameter_supported
: OPTIONAL. Boolean value specifying whether the OP supports use of the `claims` parameter, with `true` indicating support. If omitted, the default value is `false`.

request_parameter_supported
: OPTIONAL. Boolean value specifying whether the OP supports use of the `request` parameter, with `true` indicating support. If omitted, the default value is `false`.

request_uri_parameter_supported

OPTIONAL. Boolean value specifying whether the OP supports use of the `request_uri` parameter, with `true` indicating support. If omitted, the default value is `true`.

require_request_uri_registration
: OPTIONAL. Boolean value specifying whether the OP requires any `request_uri` values used to be pre-registered using the `request_uris` registration parameter. Pre-registration is REQUIRED when the value is `true`. If omitted, the default value is `false`.

op_policy_uri
: OPTIONAL. URL that the OpenID Provider provides to the person registering the Client to read about the OP's requirements on how the Relying Party can use the data provided by the OP. The registration process SHOULD display this URL to the person registering the Client if it is given.

op_tos_uri
: OPTIONAL. URL that the OpenID Provider provides to the person registering the Client to read about OpenID Provider's terms of service. The registration process SHOULD display this URL to the person registering the Client if it is given.

Other OpenID Provider Metadata values are also defined by other specifications, such as **OpenID Connect Session Management 1.0** [OpenID.Session].

4. Obtaining OpenID Provider Configuration Information

TOC

This step is OPTIONAL. The OpenID Provider endpoints and configuration information MAY be obtained out-of-band.

Using the Issuer discovered in **Section 2** or through direct configuration, the OpenID Provider's configuration can be retrieved.

OpenID Providers MUST make a JSON document available at the path formed by concatenating the string `/.well-known/openid-configuration` to the Issuer. The syntax and semantics of `.well-known` are defined in **RFC 5785** [RFC5785] and apply to the Issuer value when it contains no path component. `openid-configuration` MUST point to a JSON document compliant with this specification and MUST be returned using the `application/json` content type.

OpenID Providers supporting discovery MUST support receiving WebFinger requests via TLS. See **Section 7.1** for more information on using TLS.

4.1. OpenID Provider Configuration Request

TOC

An OpenID Provider Configuration Document MUST be queried using an HTTPS `GET` request at the previously specified path.

The Client would make the following request to the Issuer to get the Configuration information, if the Issuer contains no path component.

```
GET /.well-known/openid-configuration HTTP/1.1
Host: example.com
```

If the Issuer value contains a path component, any terminating `/` MUST be removed before appending `/.well-known/openid-configuration`. The Client would make the following request to the Issuer to get the Configuration information, if the Issuer string were `https://example.com/issuer1`

```
GET /issuer1/.well-known/openid-configuration HTTP/1.1
Host: example.com
```

Path components are allowed to support multiple issuers per host. This is required in some multi-tenant hosting configurations. This use of `.well-known` is for supporting multiple issuers per host, and unlike its use in **RFC 5785** [RFC5785], it does not provide general information about the host.

4.2. OpenID Provider Configuration Response

TOC

The response is a set of Claims about the OpenID Provider's configuration, including all necessary endpoints, supported scopes, and public key location information. The response MUST return the 200 OK response code and a plain text JSON object using the `application/json` content type that contains a set of Claims as its members that are a subset of the Metadata values defined in **Section 3**. Other Claims MAY also be returned.

Claims that return multiple values are represented as JSON arrays. Claims with zero elements MUST be omitted from the response.

The following is a non-normative example response:

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

{
 "issuer":
   "https://server.example.com",
 "authorization_endpoint":
   "https://server.example.com/connect/authorize",
 "token_endpoint":
   "https://server.example.com/connect/token",
 "token_endpoint_auth_methods_supported":
   ["client_secret_basic", "private_key_jwt"],
 "token_endpoint_auth_signing_alg_values_supported":
   ["RS256", "ES256"],
 "userinfo_endpoint":
   "https://server.example.com/connect/userinfo",
 "check_session_iframe":
   "https://server.example.com/connect/check_session",
 "end_session_endpoint":
   "https://server.example.com/connect/end_session",
 "jwks_uri":
   "https://server.example.com/jwks.json",
 "registration_endpoint":
   "https://server.example.com/connect/register",
 "scopes_supported":
   ["openid", "profile", "email", "address",
    "phone", "offline_access"],
 "response_types_supported":
   ["code", "code id_token", "id_token", "token id_token"],
 "acr_values_supported":
   ["urn:mace:incommon:iap:silver",
    "urn:mace:incommon:iap:bronze"],
 "subject_types_supported":
   ["public", "pairwise"],
```

```
 "userinfo_signing_alg_values_supported":
   ["RS256", "ES256", "HS256"],
 "userinfo_encryption_alg_values_supported":
   ["RSA1_5", "A128KW"],
 "userinfo_encryption_enc_values_supported":
   ["A128CBC-HS256", "A128GCM"],
 "id_token_signing_alg_values_supported":
   ["RS256", "ES256", "HS256"],
 "id_token_encryption_alg_values_supported":
   ["RSA1_5", "A128KW"],
 "id_token_encryption_enc_values_supported":
   ["A128CBC-HS256", "A128GCM"],
 "request_object_signing_alg_values_supported":
   ["none", "RS256", "ES256"],
 "display_values_supported":
   ["page", "popup"],
 "claim_types_supported":
   ["normal", "distributed"],
 "claims_supported":
   ["sub", "iss", "auth_time", "acr",
    "name", "given_name", "family_name", "nickname",
    "profile", "picture", "website",
    "email", "email_verified", "locale", "zoneinfo",
    "http://example.info/claims/groups"],
 "claims_parameter_supported":
   true,
 "service_documentation":
   "http://server.example.com/connect/service_documentation.html",
 "ui_locales_supported":
   ["en-US", "en-GB", "en-CA", "fr-FR", "fr-CA"]
}
```

4.3. OpenID Provider Configuration Validation

TOC

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.

If the configuration response contains the Issuer element, the value MUST exactly match the Issuer for the URL that was directly used to retrieve the configuration. Since the discovery process allows for multiple levels of redirection, this Issuer URL MAY be different from the one originally used to begin the discovery process.

5. String Operations

TOC

Processing some OpenID Connect messages requires comparing values in the messages to known values. For example, the member names in the provider configuration response might be compared to specific member names such as `issuer`. 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** [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.

6. Implementation Considerations

TOC

This specification defines features used by both Relying Parties and OpenID Providers that choose to implement Discovery. All of these Relying Parties and OpenID Providers MUST implement the features that are listed in this specification as being "REQUIRED" or are described with a "MUST". No other implementation considerations for implementations of Discovery are defined by this specification.

7. Security Considerations

TOC

7.1. TLS Requirements

TOC

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]** 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]** 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** [RFC6125].

8. IANA Considerations

TOC

8.1. Well-Known URI Registry

TOC

This specification registers the well-known URI defined in **Section 4** in the IANA Well-Known URI registry defined in **RFC 5785** [RFC5785].

8.1.1. Registry Contents

TOC

- URI suffix: `openid-configuration`
- Change controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
- Specification document: **Section 4** of this document

- Related information: (none)

9. References

TOC

9.1. Normative References

TOC

[I-D.ietf-appsawg-acct-uri] Saint-Andre, P., "**The 'acct' URI Scheme**," draft-ietf-appsawg-acct-uri-06 (work in progress), July 2013 (**TXT**).

[JWA] Jones, M., "**JSON Web Algorithms (JWA)**," draft-ietf-jose-json-web-algorithms (work in progress), October 2013 (**HTML**).

[JWE] Jones, M., Rescorla, E., and J. Hildebrand, "**JSON Web Encryption (JWE)**," draft-ietf-jose-json-web-encryption (work in progress), October 2013 (**HTML**).

[JWK] Jones, M., "**JSON Web Key (JWK)**," draft-ietf-jose-json-web-key (work in progress), October 2013 (**HTML**).

[JWS] Jones, M., Bradley, J., and N. Sakimura, "**JSON Web Signature (JWS)**," draft-ietf-jose-json-web-signature (work in progress), October 2013 (**HTML**).

[JWT] Jones, M., Bradley, J., and N. Sakimura, "**JSON Web Token (JWT)**," draft-ietf-oauth-json-web-token (work in progress), October 2013 (**HTML**).

[OAuth.JWT] Jones, M., Campbell, B., and C. Mortimore, "**JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants**," draft-ietf-oauth-jwt-bearer (work in progress), July 2013 (**HTML**).

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "**OpenID Connect Core 1.0**," October 2013.

[OpenID.Registration] Sakimura, N., Bradley, J., and M. Jones, "**OpenID Connect Dynamic Client Registration 1.0**," October 2013.

[RFC2119] **Bradner, S.**, "**Key words for use in RFCs to Indicate Requirement Levels**," BCP 14, RFC 2119, March 1997 (**TXT**, **HTML**, **XML**).

[RFC2246] **Dierks, T.** and **C. Allen**, "**The TLS Protocol Version 1.0**," RFC 2246, January 1999 (**TXT**).

[RFC3986] **Berners-Lee, T.**, **Fielding, R.**, and **L. Masinter**, "**Uniform Resource Identifier (URI): Generic Syntax**," STD 66, RFC 3986, January 2005 (**TXT**, **HTML**, **XML**).

[RFC5246] Dierks, T. and E. Rescorla, "**The Transport Layer Security (TLS) Protocol Version 1.2**," RFC 5246, August 2008 (**TXT**).

[RFC5322] **Resnick, P., Ed.**, "**Internet Message Format**," RFC 5322, October 2008 (**TXT**, **HTML**, **XML**).

[RFC5646] Phillips, A. and M. Davis, "**Tags for Identifying Languages**," BCP 47, RFC 5646, September 2009 (**TXT**).

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "**Defining Well-Known Uniform Resource Identifiers (URIs)**," RFC 5785, April 2010 (**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)**," RFC 6125, March 2011 (**TXT**).

[RFC6749] Hardt, D., "**The OAuth 2.0 Authorization Framework**," RFC 6749, October 2012 (**TXT**).

[RFC7033] Jones, P., Salgueiro, G., Jones, M., and J. Smarr, "**WebFinger**," RFC 7033, September 2013 (**TXT**).

[USA15] **Davis, M.**, **Whistler, K.**, and M. Dürst, "Unicode Normalization Forms," Unicode Standard Annex 15, 09 2009.

9.2. Informative References

TOC

[OpenID.Session] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and N. Agarwal, "**OpenID Connect Session Management 1.0**," October 2013.

[XRI_Syntax_2.0] Reed, D. and D. McAlpin, "Extensible Resource Identifier (XRI) Syntax V2.0," November 2005 (**HTML**, **PDF**).

Appendix A. Acknowledgements

TOC

This specification is the work of the OpenID AB/Connect Working Group, which includes dozens of active and dedicated participants. In particular, the following individuals contributed ideas,

feedback, and wording that influenced this specification:

Andrew Arnott, Dirk Balfanz, Casper Biering, John Bradley, Johnny Bufu, Brian Campbell, Blaine Cook, Pamela Dingle, Vladimir Dzhuvinov, George Fletcher, Dick Hardt, Roland Hedberg, Edmund Jay, Michael B. Jones, Torsten Lodderstedt, Nov Matake, Breno de Medeiros, Chuck Mortimore, Anthony Nadalin, Axel Nennker, John Panzer, Justin Richer, Nat Sakimura, Owen Shepherd, Andreas Solberg, and Kick Willemse.

Appendix B. Notices

TOC

Appendix C. Document History

TOC

[[To be removed from the final specification]]

-18

- Fixed #868 - Clarified when "alg":"none" can and cannot be used.
- Replaced uses of the OpenID Connect Messages and OpenID Connect Standard specifications with OpenID Connect Core.
- Fixed #885 - Removed normative Session Management definitions. When Session Management is supported, the Session Management discovery parameters defined in that specification are used.

-17

- Required that the OpenID Provider configuration information be returned using the `application/json` content type.

- Updated the acct: URI reference to draft-ietf-appsawg-acct-uri-05.
- Updated normative text so that e-mail addresses use the acct: scheme.
- Added an example for the acct: scheme.
- Fixed #856 - Updated normative text to clarify that no input normalization is performed when the input contains an explicit scheme such as `acct` that matches the RFC 3986 `scheme ":" path-rootless` syntax.
- Fixed #859 - Added IMPORTANT NOTE TO READERS about the terminology definitions being a normative part of the specification.

-16

- Removed the `version` discovery element.
- Added a note about the future possibility of acct: URIs like `acct:joe%40example.com@example.org` when e-mail addresses are used as local account identifiers at sites.
- Stated that the JWS Compact Serialization and the JWE Compact Serialization are always used for JWS and JWE data structures.

-15

- Fixed #820 - Removed assumption that Clients that want encrypted responses also sign requests.

-14

- Fixed #801 - Removed `schema` and `id` parameters to UserInfo Endpoint.

-13

- Added Security Considerations section about TLS version requirements and usage.
- 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.
- Change from Content-Type application/json to application/jrd+json, tracking the change made in WebFinger.
- Fixed #768 - Added required `version` value to example response.
- Fixed #771 - Added required `x509_url` value to example response.
- Fixed #769 - Added Claim Type identifiers and definition.
- Fixed #770 - Added `claims_locales_supported` and `ui_locales_supported`.
- Fixed #781 - Added `require_request_uri_registration` discovery parameter.
- Fixed #772 - Added `op_policy_url` and `op_tos_url`.
- Fixed #782 - Changed uses of "_url" in identifiers to "_uri".
- Fixed #703 - Added the PKIX JWK key type for X.509 certificates and consolidated the `x509_uri`, `x509_encryption_uri`, and `jwk_encryption_uri` parameters into a combined `jwk_uri` parameter.
- Fixed #786 - Changed the name of `jwk_uri` to `jwks_uri`.
- Moved OP metadata list to its own section.
- Added the `grant_types_supported` discovery parameter.
- Added the `claims_parameter_supported`, `request_parameter_supported`, and `request_uri_parameter_supported` discovery parameters.
- Fixed #788 - Renamed "OpenID Request Object" to "Request Object".
- Use legal `acr` values in examples.

-12

- Made the OpenID Foundation Artifact Binding Working Group the change controller for the values registered with IANA.

- Added `display_values_supported`, `claim_types_supported`, and `claims_supported` discovery elements, fixing issue #656.
- Added Implementation Considerations section.
- Fixed #656 - Changed `token_endpoint_auth_type` to `token_endpoint_auth_method` and `token_endpoint_auth_types_supported` to `token_endpoint_auth_methods_supported`.
- Fixed #697 - Added `service_documentation` to enable OPs not supporting dynamic registration to say how to register clients.
- Fixed #698 - Inconsistent use of articles.
- Fixed #628 - Defined REQUIRED, RECOMMENDED, and OPTIONAL discovery elements.
- Naming consistency changes. Renamed `check_session_iframe_url` to `check_session_iframe` and `end_session_endpoint_url` back to `end_session_endpoint`.
- Fixed #705 - Switched from using Simple Web Discovery to WebFinger. This also means that Identifiers using e-mail address syntax are prefixed by the `acct:` scheme when passed as `resource` parameters to WebFinger.

-11

- 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.
- Renamed `acrs_supported` to `acr_values_supported` for naming consistency.
- Fixed #676 Allow port number to be specified for e-mail syntax identifiers.
- Improved the fix for #625 Scheme extraction.
- Clarified that `jwk_url` and `jwk_encryption_url` refer to documents containing JWK Sets - not single JWK keys.

-10

- Fixed #621 Changed Identifier definition
- Fixed #625 Scheme extraction
- Fixed #652 Identifier normalization
- Fixed #640 Added check_session_endpoint and end_session_endpoint
- Fixed #627 Configuration response must be 200 OK
- updated OAuth reference
- Clarify the use of .well-known as part of a path for multi-tenant
- Fixes #665 Add client_secret_jwt to token_endpoint_auth_algs_supported
- 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.
- Removed section on Redirection, since it was removed from Simple Web Discovery in favor of the "simple-web-discovery" domain prefix.
- Referenced OAuth 2.0 RFC -- RFC 6749.

-09

- Removed Check ID Endpoint, per issue #570
- Added PAPE Reference to the Informative References, per issue #574
- Added "id_token" response type as being MTI for OpenID Providers
- Changed default OpenID Request Object signing algorithm to RS256, per issue #571
- Use standards track version of JSON Web Token spec (draft-ietf-oauth-json-web-

token)

-08

- Remove the no path component restriction from issuer, per issue #513
- Updated Notices
- Updated References

-07

- Rename iso29115_supported to acrs_supported
- Rename jwk_document to jwk_url
- specify full email address to be used for the principal parameter
- Added token_endpoint_auth_types_supported for list of Token Endpoint authentication types
- Added token_endpoint_auth_algs_supported for Token Endpoint supported authentication algorithms
- Added 'pairwise' and 'public' to supported identifier types
- Added valid signature and encryption algorithms for OpenID Request Object
- Added URLs for JWK and X509 encryption keys
- Use RFC 6125 to verify TLS endpoints
- Removed fallback mechanism when discovery endpoint is unreachable
- Removed Account URI scheme
- Changed 'contact' to 'contacts', 'redirect_uri' to 'redirect_uris'
- Added section about string comparison rules needed
- Allows extensions to identifier normalization via specifications
- Clarifies the host in a URL
- Update John Bradley email and affiliation for Implementer's Draft
- Change flows_supported to response_types_supported
- Register openid-configuration .well-known path in IANA Considerations
- Corrected instances of x509_url_encryption to x509_encryption_url and jwk_url_encryption to jwk_encryption_url

-06

- Changed Check Session Endpoint to Check ID Endpoint to match Basic.
- Changed certs_url to x509_url to match registration and JWE format.

-05

- Ticket #46 Added text to 3.3
- Deleted duplicate check session endpoint from 4.2
- Ticket #40 Added clarification of issuer url to 4.2
- Ticket #39 Cleaned up issuer examples and added verification text.

-04

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

-03

- Corrected examples.

-02

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

-01

- Incorporate working group decisions from 5-Jul-11 spec call.
- Consistency and cleanup pass, including removing unused references.

-00

- Initial version based upon former openid-connect-swd-1_0 spec.

Authors' Addresses

TOC

Nat Sakimura
Nomura Research Institute, Ltd.
Email: n-sakimura@nri.co.jp
URI: http://nat.sakimura.org/

John Bradley
Ping Identity
Email: ve7jtb@ve7jtb.com
URI: http://www.thread-safe.com/

Michael B. Jones
Microsoft
Email: mbj@microsoft.com
URI: http://self-issued.info/

Edmund Jay
Illumila
Email: ejay@mgi1.com

3-1 Oct 25, 2013, 9:28 AM, George Fletcher

Relying Party is capitalized but not defined. This probably doesn't matter, but just wanted to check given the earlier comment about "capitalized" terms being normative.

3-2 Oct 25, 2013, 9:28 AM, George Fletcher

I'm assuming this means... The Issuer MUST be returned as a result of the OP discovery flow. Webfinger allows for discovery endpoint redirection and requiring the Issuer in the response seems to preclude that option.

7-1 Oct 25, 2013, 9:28 AM, George Fletcher

What is account chooser doing in this case? Is it the IdPs responsibility to put the non-owned domaines loginID into Account Chooser?

8-1 Oct 25, 2013, 9:28 AM, George Fletcher

I'm not quite sure how to really comply with this as at AOL all the RS's define their own scopes. Keeping the AS up to date with all in use scopes has some operational issues. Also, some scopes we may wish to not publish. I realize that this item is just RECOMMENDED but that is still very strong. Curious how other Authorizations Servers are dealing with this.