Draft

N. Sakimura
NRI
J. Bradley
Ping Identity
M. Jones
Microsoft
October 15, 2013

TOC

OpenID Connect Dynamic Client Registration 1.0 - draft 20

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 describes how an OpenID Client can obtain the necessary Client Credentials required by the OpenID Connect protocol suite.

Table of Contents

1. Introduction

TOC

In order for an OpenID Connect Client to utilize OpenID services for an End-User, the Client needs to register with the OpenID Provider to acquire a Client ID and shared secret. This document describes how a new Client can register with the OP, and how registration information for the Client can be retrieved.

2-1

The Client Registration Endpoint MAY be co-resident with the Token Endpoint as an optimization in some deployments.

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 defines the following additional terms:

2-2

Client Registration Endpoint
: OAuth 2.0 Protected Resource through which a Client can be registered at an Authorization Server.

Client Configuration Endpoint
: OAuth 2.0 Endpoint through which registration information for a registered Client can be managed. This URL for this endpoint is returned by the Authorization Server in the Client Information Response.

Registration Access Token
: OAuth 2.0 Bearer Token issued by the Authorization Server through the Client Registration Endpoint that is used to authenticate the caller when accessing the Client's registration information at the Client Configuration Endpoint. This Access Token is associated with a particular registered Client.

Initial Access Token
: OAuth 2.0 Access Token optionally issued by an Authorization Server granting access to its Client Registration Endpoint. The contents of this token are service-specific and are

out of scope for this specification. The means by which the Authorization Server issues this token and the means by which the Registration Endpoint validates it are also out of scope.

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 "Client Registration Endpoint", reference these defined terms. Whenever the reader encounters them, their definitions found in this section must be followed.

---

2. Client Metadata

TOC

Clients have metadata associated with their unique Client Identifier at the Authorization Server. These can range from human-facing display strings, such as a Client name, to items that impact the security of the protocol, such as the list of valid redirect URIs.

The Client Metadata values are used in two ways:

- as input values to registration requests, and
- as output values in registration responses and read responses.

These Client Metadata values are used by OpenID Connect:

redirect_uris
: REQUIRED. Array of redirection URI values used in the Authorization Code and Implicit grant types. One of these registered redirection URI values MUST exactly match the `redirect_uri` parameter value used in each Authorization Request, with the matching performed as described in Section 6.2.1 of **[RFC3986]** (Simple String Comparison).

response_types
: OPTIONAL. JSON array containing a list of the OAuth 2.0 `response_type` values that the Client is declaring that it will restrict itself to using. If omitted, the default is that the Client will use only the `code` response type.

grant_types
: OPTIONAL. JSON array containing a list of the OAuth 2.0 grant types that the Client is declaring that it will restrict itself to using. The grant type values used by OpenID Connect are:
  - `authorization_code`: The Authorization Code Grant described in OAuth 2.0 Section 4.1.
  - `implicit`: The Implicit Grant described in OAuth 2.0 Section 4.2.
  - `refresh_token`: The Refresh Token Grant described in OAuth 2.0 Section 6.
  - `urn:ietf:params:oauth:grant-type:jwt-bearer`: The JWT Bearer grant type defined in **JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants** [OAuth.JWT].

  The following table lists the correspondence between `response_type` values that the Client will use and `grant_type` values that MUST be included in the registered `grant_types` list:
  - `code`: `authorization_code`
  - `id_token`: `implicit`
  - `token id_token`: `implicit`
  - `code id_token`: `authorization_code`, `implicit`
  - `code token`: `authorization_code`, `implicit`
  - `code token id_token`: `authorization_code`, `implicit`

  If omitted, the default is that the Client will use only the `authorization_code` grant type.

application_type
: OPTIONAL. Kind of the application. The default if not specified is `web`. The defined values are `native` or `web`. Web Clients using the OAuth implicit grant type MUST only register URLs using the `https` scheme as `redirect_uris`; they MUST NOT use `localhost` as the hostname. Native Clients MUST only register `redirect_uris` using custom URI schemes or URLs using the `http:` scheme with `localhost` as the hostname. Authorization Servers MAY place additional constraints on Native Clients. The Authorization Server MUST verify that all the registered `redirect_uris` conform to these constraints. This prevents sharing a Client ID across different types of Clients.

contacts
: OPTIONAL. Array of e-mail addresses of people responsible for this Client. This might be used by some providers to enable a Web user interface to modify the Client information.

client_name
: OPTIONAL. Name of the Client to be presented to the End-User. If desired, representation of this Claim in different languages and scripts is represented as described in **Section 2.1**.

logo_uri
: OPTIONAL. URL that references a logo for the Client application. If present, the server SHOULD display this image to the End-User during approval. The value of this field MUST point to a valid image file. If desired, representation of this Claim in different languages and scripts is represented as described in **Section 2.1**.

client_uri
: OPTIONAL. URL of the home page of the Client. The value of this field MUST point to a valid Web page. If present, the server SHOULD display this URL to the End-User in a followable fashion. If desired, representation of this Claim in different languages and scripts is represented as described in **Section 2.1**.

policy_uri
: OPTIONAL. URL that the Relying Party Client provides to the End-User to read about the how the profile data will be used. The value of this field MUST point to a valid web page. The OpenID Provider SHOULD display this URL to the End-User if it is given. If desired, representation of this Claim in different languages and scripts is represented as described in **Section 2.1**.

tos_uri
: OPTIONAL. URL that the Relying Party Client provides to the End-User to read about the Relying Party's terms of service. The value of this field MUST point to a valid web page. The OpenID Provider SHOULD display this URL to the End-User if it is given. If desired, representation of this Claim in different languages and scripts is represented as described in **Section 2.1**.

jwks_uri
: OPTIONAL. URL for the Client's JSON Web Key Set **[JWK]** document. If the Client signs requests to the Server, it contains the signing key(s) the Server uses to validate signatures from the Client. The JWK Set MAY also contain the Client's encryption keys(s), which are used by the Server to encrypt responses to the Client. 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.

sector_identifier_uri
: OPTIONAL. URL using the `https` scheme to be used in calculating Pseudonymous Identifiers by the OP. The URL references a file with a single JSON array of `redirect_uri` values. Please see **Section 6**. Providers that use pairwise `sub` (subject) values SHOULD provide a `sector_identifier_uri`.

4-1

subject_type
: OPTIONAL. `subject_type` requested for responses to this `client_id`. The `subject_types_supported` element of discovery contains a list of the supported `subject_type` values for this server. Valid types include `pairwise` and `public`.

id_token_signed_response_alg
: OPTIONAL. JWS `alg` algorithm **[JWA]** REQUIRED for the ID Token issued to this

4-2

client_id. The value none MUST NOT be used as the ID Token alg value unless the Client uses only the Authorization Code Flow. The default if not specified is RS256. The public key for validating the signature is provided by retrieving the JWK Set referenced by the jwks_uri element from **OpenID Connect Discovery 1.0** [OpenID.Discovery].

id_token_encrypted_response_alg
: OPTIONAL. JWE alg algorithm **[JWA]** REQUIRED for encrypting the ID Token issued to this client_id. If this is requested, the response will be signed then encrypted. The default, if not specified, is no encryption.

id_token_encrypted_response_enc
: OPTIONAL. JWE enc algorithm **[JWA]** REQUIRED for encryption of the ID Token issued to this client_id. If id_token_encrypted_response_alg is specified, the default for this parameter is A128CBC-HS256. If this is requested in combination with signing, the response will be signed then encrypted. If this is specified, the response will be **JWT** [JWT] serialized, and encrypted using JWE.

5-1

userinfo_signed_response_alg
: OPTIONAL. JWS alg algorithm **[JWA]** REQUIRED for UserInfo Responses. If this is specified the response will be **JWT** [JWT] serialized, and signed using JWS.

userinfo_encrypted_response_alg
: OPTIONAL. **JWE** [JWE] alg algorithm **[JWA]** REQUIRED for encrypting UserInfo Responses. If this is requested in combination with signing the response will be signed then encrypted. If this is specified the response will be **JWT** [JWT] serialized, and encrypted using JWE.

5-2

userinfo_encrypted_response_enc
: OPTIONAL. JWE enc algorithm **[JWA]** REQUIRED for encryption of UserInfo Responses. If userinfo_encrypted_response_alg is specified the default for this value is A128CBC-HS256. If this is requested in combination with signing the response will be signed then encrypted. If this is specified the response will be **JWT** [JWT] serialized, and encrypted using JWE.

5-3

request_object_signing_alg
: OPTIONAL. **JWS** [JWS] alg algorithm **[JWA]** that MUST be used for signing Request Objects sent to the Authorization Server. All Request Objects from this client_id MUST be rejected if not signed with this algorithm. Servers SHOULD support RS256. The value none MAY be used.

token_endpoint_auth_method
: OPTIONAL. Requested authentication method for the 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
: OPTIONAL. **JWS** [JWS] alg algorithm **[JWA]** that MUST be used for signing the JWT **[JWT]** used to authenticate the Client at the Token Endpoint for the private_key_jwt and client_secret_jwt authentication methods. All Token Requests using these authentication methods from this client_id MUST be rejected if the JWT is not signed with this algorithm. Servers SHOULD support RS256. The value none MUST NOT be used.

default_max_age
: OPTIONAL. Default Maximum Authentication Age. Specifies that the End-User MUST be actively authenticated if the End-User was authenticated longer ago than the specified number of seconds. The max_age request parameter overrides this default value.

5-4

require_auth_time
: OPTIONAL. Boolean value specifying whether the auth_time Claim in the id_token is REQUIRED. It is REQUIRED when the value is true. The auth_time Claim request in the Request Object overrides this setting.

default_acr_values
: OPTIONAL. Default requested Authentication Context Class Reference values. Array of

strings that specifies the default `acr` values that the Authorization Server is being requested to use for processing requests from this Client, with the values appearing in order of preference. The Authentication Context Class satisfied by the authentication performed is returned as the `acr` Claim Value in the issued ID Token. The `acr` Claim is requested as a Voluntary Claim by this parameter. The `acr_values_supported` discovery element contains a list of the supported `acr` values supported by this server. Values specified in the `acr_values` request parameter or an individual `acr` Claim request override these default values.

initiate_login_uri

6-1

OPTIONAL. URI using the `https` scheme that the Authorization Server can call to initiate a login at the Client. The URI MUST accept requests via both `GET` and `POST`. The Client MUST understand the `login_hint` and `iss` parameters and SHOULD support the `target_link_uri` parameter.

request_uris

OPTIONAL. Array of `request_uri` values that are pre-registered by the Client for use at the Authorization Server. Servers MAY cache the contents of the files referenced by these URIs and not retrieve them at the time they are used in a request. OPs can require that `request_uri` values used be pre-registered with the `require_request_uri_registration` discovery parameter.
If the contents of the request file could ever change, these URI values SHOULD include the base64url encoded SHA-256 hash value of the file contents referenced by the URI as the value of the URI fragment. If the fragment value used for a URI changes, that signals the server that its cached value for that URI with the old fragment value is no longer valid.

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

2.1. Metadata Languages and Scripts

TOC

Human-readable Client Metadata values and Client Metadata values that reference human-readable values MAY be represented in multiple languages and scripts. For example, values such as `client_name`, `tos_uri`, `policy_uri`, `logo_uri`, and `client_uri` might have multiple locale-specific values in some Client registrations.

To specify the languages and scripts, **BCP47** [RFC5646] language tags are added to Client Metadata member names, delimited by a `#` character. The same syntax is used for representing languages and scripts for Client Metadata as is used for Claims, as described in Section 4.2.2 (Claims Languages and Scripts) of **OpenID Connect Core 1.0** [OpenID.Core].

If such a human-readable field is sent without a language tag, parties using it MUST NOT make any assumptions about the language, character set, or script of the string value, and the string value MUST be used as-is wherever it is presented in a user interface. To facilitate interoperability, it is RECOMMENDED that any human-readable fields sent without language tags contain values suitable for display on a wide variety of systems.

---

3. Client Registration Endpoint

TOC

The Client Registration Endpoint is an OAuth 2.0 Protected Resource through which a new Client registration can be requested. The OpenID Provider MAY require an Initial Access Token that is provisioned out-of-band (in a manner that is out of scope for this specification) to restrict registration requests to only authorized Clients or developers.

To support open registration, the Client Registration Endpoint SHOULD accept registration

requests without OAuth 2.0 Access Tokens. These requests MAY be rate-limited or otherwise limited to prevent a denial-of-service attack on the Client Registration Endpoint. If an Initial Access Token is required for Client registration, the Client Registration Endpoint MUST be able to accept these Access Tokens in the manner described in the **OAuth 2.0 Bearer Token Usage** [RFC6750] specification.

3.1. Client Registration Request

TOC

To register a new Client to the Authorization Server, the Client sends an HTTP `POST` message to the Client Registration Endpoint with any Client Metadata parameters that the Client chooses to specify for itself during the registration. The Authorization Server assigns this Client a unique Client Identifier, optionally assigns a Client Secret, and associates the Metadata given in the request with the issued Client Identifier. The Authorization Server MAY provision default values for any items omitted in the Client Metadata.

The Client sends an HTTP `POST` to the Client Registration Endpoint with a content type of `application/json` and all parameters as top-level members of a JSON object.

For example, a Client could send the following registration request to the Client Registration Endpoint:

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

```
POST /connect/register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: server.example.com
Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.eyJ ...

{
 "application_type": "web",
 "redirect_uris":
   ["https://client.example.org/callback",
    "https://client.example.org/callback2"],
 "client_name": "My Example",
 "client_name#ja-Jpan-JP":
   "クライアント名",
 "logo_uri": "https://client.example.org/logo.png",
 "subject_type": "pairwise",
 "sector_identifier_uri":
   "https://other.example.net/file_of_redirect_uris.json",
 "token_endpoint_auth_method": "client_secret_basic",
 "jwks_uri": "https://client.example.org/my_public_keys.jwks",
 "userinfo_encrypted_response_alg": "RSA1_5",
 "userinfo_encrypted_response_enc": "A128CBC-HS256",
 "contacts": ["ve7jtb@example.org", "mary@example.org"],
 "request_uris":
   ["https://client.example.org/rf.txt
     #qpXaRLh_n93TTR9F252ValdatUQvQiJi5BDub2BeznA"]
}
```

3.2. Client Registration Response

TOC

Upon successful registration, the Client Registration Endpoint returns the newly-created Client Identifier and, if applicable, a Client Secret, along with all registered Metadata about this Client, including any fields provisioned by the Authorization Server itself. The Authorization Server MAY reject or replace any of the Client's requested field values and substitute them with suitable values. If this happens, the Authorization Server MUST include these fields in the response to the Client. An Authorization Server MAY ignore values provided by the client, and MUST ignore any fields send by the Client that it does not understand.

8-1

The response MAY contain a Registration Access Token that can be used by the Client to perform subsequent operations upon the resulting Client registration.

The response should use an HTTP 201 Created status code.

All of the response items are returned as a **JSON document** [RFC4627] with the following fields as top-level members of the root JSON object.

client_id
: REQUIRED. Unique Client Identifier. It MUST NOT be currently valid for any other registered Client.

client_secret
: OPTIONAL. Client secret. This MUST be unique for each `client_id`. This value is used by Confidential Clients to authenticate to the Token Endpoint as described in OAuth 2.0 Section 2.3.1. It is not needed for Clients selecting a `token_endpoint_auth_method` of `private_key_jwt`.

registration_access_token
: OPTIONAL. Registration Access Token that can be used at the Client Configuration Endpoint to perform subsequent operations upon the Client registration.

registration_client_uri
: OPTIONAL. Location of the Client Configuration Endpoint where the Registration Access Token can be used to perform subsequent operations upon the resulting Client registration. Implementations MUST either return both a Client Configuration Endpoint and a Registration Access Token or neither of them.

client_id_issued_at
: OPTIONAL. Time at which the Client Identifier 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.

client_secret_expires_at
: REQUIRED if `client_secret` is issued. Time at which the `client_secret` will expire or 0 if it will not expire. The time is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.

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

8-2

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
 "client_id": "s6BhdRkqt3",
 "client_secret":
   "ZJYCqe3GGRvdrudKyZS0XhGv_Z45DuKhCUk0gBR1vZk",
 "client_secret_expires_at": 1577858400,
 "registration_access_token":
   "this.is.an.access.token.value.ffx83",
 "registration_client_uri":
   "https://server.example.com/connect/register?client_id=s6BhdRkqt3",
```

```
 "token_endpoint_auth_method":
   "client_secret_basic",
 "application_type": "web",
 "redirect_uris":
   ["https://client.example.org/callback",
    "https://client.example.org/callback2"],
 "client_name": "My Example",
 "client_name#ja-Jpan-JP":
   "クライアント名",
 "logo_uri": "https://client.example.org/logo.png",
 "subject_type": "pairwise",
 "sector_identifier_uri":
   "https://other.example.net/file_of_redirect_uris.json",
 "jwks_uri": "https://client.example.org/my_public_keys.jwks",
 "userinfo_encrypted_response_alg": "RSA1_5",
 "userinfo_encrypted_response_enc": "A128CBC-HS256",
 "contacts": ["ve7jtb@example.org", "mary@example.org"],
 "request_uris":
   ["https://client.example.org/rf.txt
     #qpXaRLh_n93TTR9F252ValdatUQvQiJi5BDub2BeznA"]
}
```

3.3. Client Registration Error Response

TOC

When an OAuth error condition occurs, the Client Registration Endpoint returns an Error Response as defined in Section 3 of the **OAuth 2.0 Bearer Token Usage** [RFC6750] specification.

When a registration error condition occurs, the Client Registration Endpoint returns a HTTP 400 status code including a JSON object describing the error in the response body.

The JSON object contains two members:

error
: Error code.

error_description
: Additional text description of the error for debugging.

This specification defines the following error codes:

invalid_redirect_uri
: The value of one or more `redirect_uris` is invalid.

invalid_client_metadata
: The value of one of the Client Metadata fields is invalid and the server has rejected this request. Note that an Authorization Server MAY choose to substitute a valid value for any requested parameter of a Client's Metadata.

The following is a non-normative example error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
 "error": "invalid_redirect_uri",
 "error_description": "The value of one or more redirect_uris are invalid."
}
```

4. Stateless Dynamic Client Registration

TOC

In some deployments, it is advantageous to enable Clients to obtain the information necessary to interact with the Authorization Server, such as a Client Identifier, without the requirement that state about the Client be stored at the Authorization Server. The interfaces defined by this specification can be used for stateless dynamic client registration.

One means of doing this is to encode necessary registration information about the Client into the `client_id` value returned by the initial registration of the Client. This has the effect of having the Client store this information, rather than the Authorization Server. The particular encodings used by different Authorization Servers will differ.

When stateless dynamic client registration is used by the Authorization Server, read operations are likely to not be possible. In that case, no Client Configuration Endpoint or Registration Access Token will be returned by the initial registration of the Client.

5. Client Configuration Endpoint

TOC

The Client Configuration Endpoint is an OAuth 2.0 protected resource that MAY be provisioned by the server for a specific Client to be able to view and update its registered information. The Client MUST use its Registration Access Token in all calls to this endpoint as an OAuth 2.0 Bearer Token **[RFC6750]**.

Operations on this endpoint are switched through the use of different HTTP methods **[RFC2616]**. The only method defined for use at this endpoint by this specification is the HTTP `GET` method.

5.1. Forming the Client Configuration Endpoint URL

TOC

If a Client Configuration Endpoint and a Registration Access Token are returned by the initial registration of the Client, the Authorization Server MUST provide the Client with the fully qualified URL in the `registration_client_uri` element of the Client Registration Response, per **Section 3.2**. The Authorization Server MUST NOT expect the Client to construct or discover this URL on its own. The Client MUST use the URL as given by the server and MUST NOT construct this URL from component pieces.

Depending on deployment characteristics, the Client Configuration Endpoint URL can take any number of forms. It is RECOMMENDED that this endpoint URL be formed through the use of a server-constructed URL string which combines the Client Registration Endpoint's URL and the issued Client ID for this Client, with the latter as either a path parameter or a query parameter. For example, a Client with the Client ID `s6BhdRkqt3` could be given a Client Configuration Endpoint URL of `https://server.example.com/register/s6BhdRkqt3` (path parameter) or of `https://server.example.com/register?client_id=s6BhdRkqt3` (query parameter). In both of these cases, the Client simply uses the URL as given.

These common patterns can help the Server to more easily determine the Client to which the request pertains, which MUST be matched against the Client to which the Registration Access Token was issued. If desired, the Server MAY simply return the Client Registration Endpoint URL as the Client Configuration Endpoint URL and change behavior based on the authentication context provided by the Registration Access Token.

5.2. Client Read Request

TOC

If the initial registration of the Client returned a Client Configuration Endpoint and a Registration Access Token, the current configuration of the Client on the Authorization Server can be read by making an HTTP `GET` request to the Client Configuration Endpoint with the Registration Access Token.

The following is a non-normative example request:

```
GET /connect/register?client_id=s6BhdRkqt3 HTTP/1.1
Accept: application/json
Host: server.example.com
Authorization: Bearer this.is.an.access.token.value.ffx83
```

5.3. Client Read Response

TOC

Upon a successful read operation, the Authorization Server SHOULD return all registered Metadata about this Client, including any fields provisioned by the Authorization Server itself. Some values, including the `client_secret` value, might have been updated since the initial registration.

The Authorization Server need not include the `registration_access_token` or `registration_client_uri` value in this response unless they have been updated.

The response should use an HTTP 200 OK status code.

The response is a JSON Document **[RFC4627]** with the Client Metadata as top-level members of a JSON object.

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

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
 "client_id": "s6BhdRkqt3",
 "client_secret":
   "OylyaC56ijpAQ7G5ZZGL7MMQ6Ap6mEeuhSTFVps2N4Q",
 "client_secret_expires_at": 17514165600,
 "registration_client_uri":
   "https://server.example.com/connect/register?client_id=s6BhdRkqt3",
 "token_endpoint_auth_method":
   "client_secret_basic",
 "application_type": "web",
 "redirect_uris":
   ["https://client.example.org/callback",
    "https://client.example.org/callback2"],
 "client_name": "My Example",
 "client_name#ja-Jpan-JP":
   "クライアント名",
 "logo_uri": "https://client.example.org/logo.png",
 "subject_type": "pairwise",
 "sector_identifier_uri":
   "https://other.example.net/file_of_redirect_uris.json",
 "jwks_uri": "https://client.example.org/my_public_keys.jwks",
```

```
 "userinfo_encrypted_response_alg": "RSA1_5",
 "userinfo_encrypted_response_enc": "A128CBC-HS256",
 "contacts": ["ve7jtb@example.org", "mary@example.org"],
 "request_uris":
   ["https://client.example.org/rf.txt
     #qpXaRLh_n93TTR9F252ValdatUQvQiJi5BDub2BeznA"]
}
```

5.4. Client Read Error Response

TOC

If the Registration Access Token used to make this request is not valid, the server MUST respond with an error as described in **OAuth Bearer Token Usage** [RFC6750].

When a read error condition occurs, the Client Configuration Endpoint returns a HTTP 401 Unauthorized status code. This indicates that the Access Token is invalid or the Client record requested is invalid or non-existent. Note that for security reasons, to inhibit brute force attacks, endpoints MUST NOT return 404 Not Found error codes.

The following is a non-normative example error response:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
```

12-1

6. "sector_identifier_uri" Validation

TOC

The sector identifier list provides a way for a group of Web sites under single administrative control to have consistent pairwise `sub` values, independent of their domain names, as described in Section 7.1 of **OpenID Connect Core 1.0** [OpenID.Core]. It also provides a way for Clients to change `redirect_uri` domains without having to re-register all of their users.

The value of the `sector_identifier_uri` MUST be a URL using the `https` scheme that references a JSON file containing an array of `redirect_uri` values. The values registered in `redirect_uris` MUST be included in the elements of the array, or registration MUST fail.

The following is a non-normative example request to and reply from a `sector_identifier_uri`.

```
GET https://other.example.net/file_of_redirect_uris.json HTTP/1.1
Accept: application/json
Host: client.example.org

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

[ "https://client.example.org/callback",
  "https://client.example.org/callback2",
  "https://client.other_company.example.net/callback" ]
```

7. String Operations

TOC

Processing some OpenID Connect messages requires comparing values in the messages to known values. For example, the member names in the Client registration response might be compared to specific member names such as `client_id`. 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.

---

8. 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.

---

9. Implementation Considerations

TOC

This specification defines features used by both Relying Parties and OpenID Providers that choose to implement Dynamic Client Registration. 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".

As of the time of this writing, this specification is compatible with the current version of **OAuth 2.0 Dynamic Client Registration Protocol** [I-D.ietf-oauth-dyn-reg]. The Dynamic Client Registration work is still ongoing at the IETF and changes may or may not be made there that cause it to diverge from this specification during the standardization process.

Implementations wanting to support additional operations defined in **[I-D.ietf-oauth-dyn-reg]**, such as Update, can do so using that specification, while being mindful that the specification is a work in progress, and may change.

---

10. Security Considerations

TOC

Since requests to the Client Registration Endpoint result in the transmission of clear-text credentials (in the HTTP request and response), all communication with the Registration Endpoint MUST utilize TLS. See **Section 10.2** for more information on using TLS.

A rogue RP might use the logo for the legitimate RP, which it is trying to impersonate. An OP needs to take steps to mitigate this phishing risk, since the logo could confuse users into thinking they're logging in to the legitimate RP. An OP could also warn if the domain/site of the logo doesn't match the domain/site of registered redirection URIs. An OP can also make warnings against untrusted RPs in all cases, especially if they're dynamically registered, have not been

trusted by any users at the OP before, and want to use the logo feature.

In a situation where the Authorization Server is supporting open Client registration, it needs to be extremely careful with any URL provided by the Client that will be displayed to the End-User (e.g. `logo_uri` and `policy_uri`). A rogue Client could specify a registration request with a reference to a drive-by download in the `policy_uri`. The Authorization Server SHOULD check to see if the `logo_uri` and `policy_uri` have the same host as the hosts defined in the array of `redirect_uris`.

10.1. Native Code Leakage

TOC

Implementers should be aware that on iOS, information is returned to native applications using custom URI schemes, but multiple applications can register the same URI scheme. In this case, it is nondeterministic which application receives the information. This can result in an Authorization Code being leaked to the wrong application. Several possible solutions to this have been proposed and are being discussed in the IETF OAuth working group. It is expected that a standard solution to this problem will be developed there. At that point, an extension to OpenID Connect may be published describing how to apply that solution to OpenID Connect.

10.2. 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].

11. IANA Considerations

TOC

This document makes no requests of IANA.

12. References

TOC

12.1. Normative References

TOC

[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.Discovery]	Sakimura, N., Bradley, J., Jones, M., and E. Jay, "**OpenID Connect Discovery 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**).
[RFC2616]	**Fielding, R.**, **Gettys, J.**, **Mogul, J.**, **Frystyk, H.**, **Masinter, L.**, **Leach, P.**, and **T. Berners-Lee**, "**Hypertext Transfer Protocol -- HTTP/1.1**," RFC 2616, June 1999 (**TXT**, **PS**, **PDF**, **HTML**, **XML**).
[RFC3986]	**Berners-Lee, T.**, **Fielding, R.**, and **L. Masinter**, "**Uniform Resource Identifier (URI): Generic Syntax**," STD 66, RFC 3986, January 2005 (**TXT**, **HTML**, **XML**).
[RFC4627]	Crockford, D., "**The application/json Media Type for JavaScript Object Notation (JSON)**," RFC 4627, July 2006 (**TXT**).
[RFC5246]	Dierks, T. and E. Rescorla, "**The Transport Layer Security (TLS) Protocol Version 1.2**," RFC 5246, August 2008 (**TXT**).
[RFC5646]	Phillips, A. and M. Davis, "**Tags for Identifying Languages**," BCP 47, RFC 5646, September 2009 (**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**).
[RFC6750]	Jones, M. and D. Hardt, "**The OAuth 2.0 Authorization Framework: Bearer Token Usage**," RFC 6750, October 2012 (**TXT**).
[USA15]	**Davis, M.**, **Whistler, K.**, and M. Dürst, "Unicode Normalization Forms," Unicode Standard Annex 15, 09 2009.

12.2. Informative References

TOC

[I-D.ietf-oauth-dyn-reg]	Richer, J., Bradley, J., Jones, M., and M. Machulak, "**OAuth 2.0 Dynamic Client Registration Protocol**," draft-ietf-oauth-dyn-reg-14 (work in progress), July 2013 (**TXT**).
[OpenID.Session]	Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and N. Agarwal, "**OpenID Connect Session Management 1.0**," October 2013.

Appendix A. Acknowledgements

TOC

The OpenID Community would like to thank the following people for the work they have done in the drafting and editing of this specification.

Amanda Anganes (aanganes@mitre.org), MITRE

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

Brian Campbell (bcampbell@pingidentity.com), Ping Identity

Vladimir Dzhuvinov (vladimir@nimbusds.com), Nimbus Directory Services

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

Edmund Jay (ejay@mgi1.com), Illumila

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

Justin Richer (jricher@mitre.org), MITRE

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

Appendix B. Notices

TOC

Appendix C. Document History

TOC

[[To be removed from the final specification]]

-20

- Fixed #867 - Allow ID Tokens to use "alg":"none" when using the Authorization Code Flow and when explicitly requested at registration time.
- Fixed #868 - Clarified when "alg":"none" can and cannot be used.
- Fixed #875 - Added `token_endpoint_auth_signing_alg` parameter.
- Fixed #881 - Updated statement about relationship to OAuth Dynamic Registration.
- Fixed #865 - Described the possibility of implementations using the update operation defined by the OAuth Dynamic Registration spec.
- Fixed #863 - Described how to perform stateless dynamic client registration. This included making the Client Configuration Endpoint and Registration Access Token optional.
- Fixed #864 - Described Native application code leakage problem on iOS.
- 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 registration parameters defined in that specification are used.
- Defined the HTTP status code values for responses.
- Added internationalization to the `logo_uri`, `tos_uri`, and `policy_uri` fields.

-19

- Fixed #842 - Made `post_logout_redirect_uri` treatment parallel to `redirect_uri`.
- Corrected `error_code` to `error`.
- 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 `default_acr_values` values are in order of preference and that `default_acr_values` requests the `acr` Claim as a Voluntary Claim.
- Fixed #859 - Added IMPORTANT NOTE TO READERS about the terminology definitions being a normative part of the specification.

-18

- Added subsection on Forming the Client Configuration Endpoint URL to help clarify its semantics to developers. This was explained in the OAuth Registration spec but wasn't previously explained here.
- Renamed `expires_at` to `client_secret_expires_at` and `issued_at` to `client_id_issued_at`, tracking OAuth Registration changes.
- Stated that the JWS Compact Serialization and the JWE Compact Serialization are always used for JWS and JWE data structures.

-17

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

-16

- Fixed #803 - No longer use `client_id` query parameter.
- Fixed #804 - Removed `access_token` from client metadata.

-15

- Fixed #708 - Registration access token requirement.
- Fixed #734 - Invalid JSON in examples.
- Fixed #736 - Client Update Operation Response: expires_at should be removed from example.
- Fixed #735 - Require expires_at value in Client Register response.
- Added Security Considerations section about TLS version requirements and usage.
- 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.
- Fixed #746 - Deleted the `operation` parameter.
- Fixed #745 - Deleted the `rotate_secret` operation.
- Changed the Japanese client name to make it sound more natural.
- Added optional `issued_at` response value.
- Added client update example.
- Fixed #727 - Deleted invalid_client_secret error.
- Fixed #744 - Promoted `max_age` to being a top-level parameter.
- Fixed #765 - Created `acr_values` top-level request parameter and changed `default_acr` registration parameter to `default_acr_values`.
- Fixed #747 - Changed requests from being form-urlencoded to JSON.
- Fixed #755 - Removed client update operation.
- Fixed #751 - Added client read operation.
- Fixed #749 - Added `registration_access_url`.
- Fixed #756 - State that an updated `client_secret` value can be returned by a read operation.
- Fixed #774 - Moved invalid client_id from 3.3 to 4.3 and fixed example.
- Fixed #774 - Removed invalid client_id and made GET return 403 Forbidden.

- Fixed missing registration_access_url in GET example response.
- Fixed #776 - Removed client adding client_id query parameter but make the examples include it as part of the registration_access_url.
- Fixed #775 - Made redirects_uri, contacts, and default_acr_values arrays to match the examples.
- Changed `invalid_configuration_parameter` error to `invalid_client_metadata` to match the OAuth Registration spec.
- Fixed #777 - Added `Pragma: no-cache` to the example responses that were missing it.
- Fixed #773 - Added `request_uris` registration parameter to pre-register `request_uri` values. Also clarified that Request File contents may be cached.
- Fixed #758 - State the registration requests can be rate-limited to prevent a DoS attack.
- Fixed #782 - Changed uses of "_url" in identifiers to "_uri".
- Fixed #783 - Changed `registration_access_url` to `registered_client_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`.
- Added the `response_types` registration parameter.
- Added the `grant_types` registration parameter.
- Added table documenting correspondence between `response_type` values used and `grant_type` values used.
- Fixed #788 - Renamed "OpenID Request Object" to "Request Object".

-14

- Changed the syntax of some elements to match the syntax used in the OAuth Dynamic Client Registration draft. Specifically, changed `type` to `operation`, changed `associate` to `register`, and changed `application_name` to `client_name`. Also changed the responses of `client_register` and `client_update` to include full client information instead of just the Client ID.
- 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 #698 - Inconsistent use of articles.
- Deleted `javascript_origin_uris`, which is no longer present in Session.
- Reference and provide note to implementers about **OAuth Dynamic Client Registration Protocol** [I-D.ietf-oauth-dyn-reg].
- Changed token_endpoint_auth_method example result value from "client_secret_basic client_secret_post" to "client_secret_basic" since the definition requires the value to be a single method.

-13

- 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 #685 - The policy URL should be different from the terms-of-service URL. A new `tos_url` registration parameter was added.
- Clarified that `jwk_url` and `jwk_encryption_url` refer to documents containing JWK Sets - not single JWK keys.
- Re #601 add initiate_login_uri for unsolicited request

-12

- Made application_type REQUIRED and added an explanation about redirect_uris registration
- Section 2.1 clarification that updates replace all parameters previously set.
- Section 2.3 add rotate_secret to invalid client_id error
- Added registration_access_token for updating and made client secret optional
- added registration_access_token to example response
- removed client_id from request as the client_id is implicit in the access token for updates
- Changed redirect_uris from RECOMMENDED for code and REQUIRED for implicit to REQUIRED
- Changed 2.1 to only allow access_token as a parameter if type is rotate_secret
- Fixed reference in application_name and added example of ja-Hani-JP encoded name.
- Made application_type OPTIONAL with web as the default
- Fixes #642 - Registration separates application errors from bearer.
- Updated references to OAuth and Bearer to reflect current drafts
- Fix typo error_description
- Re #642 change error to error_code in 2.3 example
- 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 #673 - Registration 2.1: Rename require_signed_request_object to request_object_alg. The actual change was to rename require_signed_request_object to request_object_signing_alg, following the naming convention used in the resolution to issue #614.
- Fixed #666 - JWS signature validation vs. verification.
- Referenced OAuth 2.0 RFCs -- RFC 6749 and RFC 6750.
- Fixed #674 - Description of require_auth_time.

-11

- Made `rotate_secret` a separate registration request type and stop client secret changing with every response, per issue #363
- Changed default ID Token signing algorithm to RS256, per issue #571
- Changed client.example.com to client.example.org, per issue #251
- Added text for authz to the registration endpoint, per issue #587
- Use standards track version of JSON Web Token spec (draft-ietf-oauth-json-web-token)

-10

- Split encrypted response configurations into separate parameters for alg, enc, int
- Removed extra "s" from signed response parameter names
- Add reference to JWA
- Updated Notices
- Updated References

-09

- Removed erroneous spanx declarations from example
- Fixed example in Sec 2.2 to show expires_at
- Fixed Sec 2.1.1 to clarify it is the registration server doing the certificate check
- Fixed Sec 2.1.1 example to include http portion of response
- Fixed #542 Sec 2.1 userinfo_signed_response_algs fixed to say signature. Clarify

response is signed.
- Fixed Sec 2.1 userinfo_encrypted_response_algs Clarify response is JWE containing JWT
- Fixes #529 Sec 2.3 Clarify error response is Bearer and fix example
- Add default_max_age registration parameter
- Add default_acr registration parameter
- Add require_auth_time registration parameter

-08

- Replaced token_endpoint with a defined term Token Endpoint [OAuth 2.0]
- Added policy_url parameter
- Renamed expires_in to expires_at
- Registration Endpoint can be OAuth Protected
- Added parameters for requiring encryption and/or signing of OpenID Request Object, UserInfo and ID Token
- Added token_endpoint_auth_type and list of valid authentication types
- Added JWK and X509 URLs for signature and encryption
- Added user_id_type
- Changed sector_identifier to sector_identifier_url and added URL verification
- Use RFC 6125 to verify TLS endpoints
- Changed 'contact' to 'contacts', 'redirect_uri' to 'redirect_uris'
- Changed redirect_uris to RECOMMENDED for code flow and REQUIRED for implicit flow Clients
- Removed js_origin_uri
- Added section about string comparison rules needed
- Clarified redirect_uris matching
- Update John Bradley email and affiliation for Implementer's Draft

-07

- Changed request from posting a JSON object to being HTTP Form encoded.
- Added x509_url to support optional encryption.

-06

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

-05

- Changed `redirect_url` to `redirect_uri` and `js_origin_url` to `js_origin_uri`.

-04

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

-03

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

-02

- Incorporate working group decisions from 23-Jun-11 spec call.

-01

- Initial version.

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/

2-1 Oct 25, 2013, 11:57 AM, George Fletcher
Does co-resident mean the same as the /token API endpoint? Or just on the same server? Overloading the /token endpoint seems like a bad idea.

2-2 Oct 25, 2013, 11:57 AM, George Fletcher
Could (or should) these endpoint also be optionally returned in the openid connect discovery JSON response?

4-1 Oct 25, 2013, 11:57 AM, George Fletcher
This sentence doesn't make sense to me. I think the point is that providers that support pair wise 'sub' identifiers must support client registered sector_identifier_uri values.

4-2 Oct 25, 2013, 11:57 AM, George Fletcher
This implies to me that the client gets to dictate what algorithm is going to be used to sign the ID Token. That seems the opposite of all the other cases where the server gets to chose. Is the expected response a registration failure if the client requests an alg not supported by the AS?

5-1 Oct 25, 2013, 11:57 AM, George Fletcher
If this value is not specified, then the default is that user info responses are NOT signed?

5-2 Oct 25, 2013, 11:57 AM, George Fletcher
This wording is used many times for these different responses that can be both signed and/or encrypted. I'm wondering if this sentence should include something like....If this option is specified when no signing algorithm is specifed, then ...

5-3 Oct 25, 2013, 11:57 AM, George Fletcher
Do we need to specify a default?

5-4 Oct 25, 2013, 11:57 AM, George Fletcher
As an OP I'm not sure I want to honor this parameter from a client. I'm assuming that the OP can ignore any parameter from the client it wants and just not return any value for that parameter in the response.

6-1 Oct 25, 2013, 11:57 AM, George Fletcher
Would a reference to the relevant section of core be useful here? I don't remember this flow very well and this is the first mention of target_link_uri in this document.

8-1 Oct 25, 2013, 11:57 AM, George Fletcher
'send' should be 'sent'

8-2 Oct 25, 2013, 11:57 AM, George Fletcher
I think this should be a 201 response instead of a 200.

12-1 Oct 25, 2013, 11:57 AM, George Fletcher
It seems like there is some pretty complicated OP logic required to process the sector_identifier_uri. Given that the the list of allowed redirect_uris in the JSON file can change at any time! the OP would need to pull the file and verify that the current client redirect_uri is still present in the list. That is too much over head to do at token issuance. Should we have some guidance that redirect_uris can be added to the sector_identifier_uri file but SHOULD NOT be removed. Removing a redirect_uri from the file results in undefined behavior? With this guidance the OP can do all the necessary checking at client registration time which seems reasonable.