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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Draft | N. Sakimura | |  | NRI | |  | J. Bradley | |  | Ping Identity | |  | M. Jones | |  | Microsoft | |  | June 27, 2013 | |

# OpenID Connect Dynamic Client Registration 1.0 - draft 19

### Abstract

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

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)  Introduction  
    [**1.1.**](#rnc)  Requirements Notation and Conventions  
    [**1.2.**](#terminology)  Terminology  
[**2.**](#client-metadata)  Client Metadata  
    [**2.1.**](#LanguagesAndScripts)  Metadata Languages and Scripts  
[**3.**](#ClientRegistration)  Client Registration  
    [**3.1.**](#RegistrationRequest)  Client Registration Request  
    [**3.2.**](#RegistrationResponse)  Client Registration Response  
    [**3.3.**](#RegistrationError)  Client Registration Error Response  
[**4.**](#AccessEndpoint)  Client Configuration Endpoint  
    [**4.1.**](#AccessURL)  Forming the Client Configuration Endpoint URL  
    [**4.2.**](#ReadRequest)  Client Read Request  
    [**4.3.**](#ReadResponse)  Client Read Response  
    [**4.4.**](#ReadError)  Client Read Error Response  
[**5.**](#sector.identifier.url.validation)  "sector\_identifier\_uri" Validation  
[**6.**](#stringops)  String Operations  
[**7.**](#Validation)  Validation  
[**8.**](#ImplementationConsiderations)  Implementation Considerations  
[**9.**](#Security)  Security Considerations  
    [**9.1.**](#TLS_requirements)  TLS Requirements  
[**10.**](#IANA)  IANA Considerations  
[**11.**](#rfc.references1)  References  
    [**11.1.**](#rfc.references1)  Normative References  
    [**11.2.**](#rfc.references2)  Informative References  
[**Appendix A.**](#Acknowledgements)  Acknowledgements  
[**Appendix B.**](#Notices)  Notices  
[**Appendix C.**](#History)  Document History  
[**§**](#rfc.authors)  Authors' Addresses

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

### 1.  Introduction

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.

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

Note: This specification will likely be modified to use the [OAuth 2.0 Dynamic Client Registration Protocol (Richer, J., Bradley, J., Jones, M., and M. Machulak, “OAuth 2.0 Dynamic Client Registration Protocol,” May 2013.)](#I-D.ietf-oauth-dyn-reg) [I‑D.ietf‑oauth‑dyn‑reg] specification once it is stable. While currently self-contained, this specification intentionally uses the same syntax and identifiers as the OAuth Registration draft. As of the time of this writing, the two specifications are consistent with one another.

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

### 1.1.  Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119 (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.)](#RFC2119) [RFC2119].

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

All uses of [JSON Web Signature (JWS) (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](#JWS) [JWS] and [JSON Web Encryption (JWE) (Jones, M., Rescorla, E., and J. Hildebrand, “JSON Web Encryption (JWE),” May 2013.)](#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.

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

### 1.2.  Terminology

This specification uses the terms "Access Token", "Refresh Token", "Authorization Code", "Authorization Grant", "Authorization Server", "Authorization Endpoint", "Client", "Client Identifier", "Client Secret", "Protected Resource", "Resource Owner", "Resource Server", and "Token Endpoint" defined by [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749], and the terms defined by [OpenID Connect Messages 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Messages 1.0,” June 2013.)](#OpenID.Messages) [OpenID.Messages].

This specification defines the following additional terms:

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.

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

### 2.  Client Metadata

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.

Client Metadata values used by OpenID Connect are:

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] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.)](#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 [OAuth JWT Bearer Token Profiles (Jones, M., Campbell, B., and C. Mortimore, “JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0,” March 2013.)](" \l "OAuth.JWT) [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 (Metadata Languages and Scripts)](#LanguagesAndScripts).

logo\_uri

OPTIONAL. URL that references a logo for the Client application. The value of this field MUST point to a valid image file.

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 (Metadata Languages and Scripts)](#LanguagesAndScripts).

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 2.2.1 of [OpenID Connect Messages 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Messages 1.0,” June 2013.)](#OpenID.Messages) [OpenID.Messages]. Other Authentication methods MAY be defined by extensions. If unspecified or omitted, the default is client\_secret\_basic HTTP Basic Authentication Scheme as specified in Section 2.3.1 of [OAuth 2.0 (Hardt, D., “The OAuth 2.0 Authorization Framework,” October 2012.)](#RFC6749) [RFC6749].

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 OpenID Provider SHOULD display this URL to the End-User if it is given.

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 OpenID Provider SHOULD display this URL to the End-User if it is given.

jwks\_uri

OPTIONAL. URL for the Client's JSON Web Key Set [[JWK] (Jones, M., “JSON Web Key (JWK),” May 2013.)](#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. OPTIONAL.

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 5 ("sector\_identifier\_uri" Validation)](#sector.identifier.url.validation). Providers that use pairwise sub (subject) values SHOULD provide a sector\_identifier\_uri.

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.

request\_object\_signing\_alg

OPTIONAL. [JWS (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Signature (JWS),” May 2013.)](#JWS) [JWS] alg algorithm [[JWA] (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) that MUST be used for requests to the Authorization Server. The valid values are listed in Section 3.1 of [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) [JWA]. All Request Objects from this client\_id MUST be rejected if not signed by this algorithm. Servers SHOULD support RS256.

userinfo\_signed\_response\_alg

OPTIONAL. JWS alg algorithm [[JWA] (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) REQUIRED for UserInfo Responses. The valid values are listed in Section 3.1 of [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) [JWA]. If this is specified the response will be [JWT (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Token (JWT),” May 2013.)](#JWT) [JWT] serialized, and signed using JWS.

userinfo\_encrypted\_response\_alg

OPTIONAL. [JWE (Jones, M., Rescorla, E., and J. Hildebrand, “JSON Web Encryption (JWE),” May 2013.)](#JWE) [JWE] alg algorithm [[JWA] (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) REQUIRED for encrypting UserInfo Responses. The valid values are listed in Section 4.1 of [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) [JWA]. If this is requested in combination with signing the response will be signed then encrypted. If this is specified the response will be [JWT (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Token (JWT),” May 2013.)](#JWT) [JWT] serialized, and encrypted using JWE.

userinfo\_encrypted\_response\_enc

OPTIONAL. JWE enc algorithm [[JWA] (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) REQUIRED for symmetric encryption of UserInfo Responses. The valid values are listed in Section 4.2 [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) [JWA]. 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 (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Token (JWT),” May 2013.)](#JWT) [JWT] serialized, and encrypted using JWE.

id\_token\_signed\_response\_alg

OPTIONAL. JWS alg algorithm [[JWA] (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) REQUIRED for the ID Token issued to this client\_id. The valid values are listed in Section 3.1 of [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) [JWA], with the exception of none, which MUST NOT be used as the ID Token alg value. 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 (Sakimura, N., Bradley, J., Jones, M., and E. Jay, “OpenID Connect Discovery 1.0,” June 2013.)](#OpenID.Discovery) [OpenID.Discovery].

id\_token\_encrypted\_response\_alg

OPTIONAL. JWE alg algorithm [[JWA] (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) REQUIRED for encrypting the ID Token issued to this client\_id. The valid values are listed in Section 4.1 of [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) [JWA]. 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] (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) REQUIRED for symmetric encryption of the ID Token issued to this client\_id. The valid values are listed in Section 4.2 of [JWA (Jones, M., “JSON Web Algorithms (JWA),” May 2013.)](#JWA) [JWA]. 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 (Jones, M., Bradley, J., and N. Sakimura, “JSON Web Token (JWT),” May 2013.)](#JWT) [JWT] serialized, and encrypted using JWE.

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.

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 acr Claim request override these default values.

initiate\_login\_uri

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.

post\_logout\_redirect\_uris

OPTIONAL. Array of URLs supplied by the RP to which it MAY request that the End-User's User-Agent be redirected using the post\_logout\_redirect\_uri parameter after a logout has been performed, as specified in [OpenID Connect Session Management 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and N. Agarwal, “OpenID Connect Session Management 1.0,” June 2013.)](#OpenID.Session) [OpenID.Session].

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.

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

### 2.1.  Metadata Languages and Scripts

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 (Phillips, A. and M. Davis, “Tags for Identifying Languages,” September 2009.)](#RFC5646) [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 2.5.2 (Claims Languages and Scripts) of [OpenID Connect Messages 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Messages 1.0,” June 2013.)](#OpenID.Messages) [OpenID.Messages].

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.

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

### 3.  Client Registration

The Client Registration Endpoint is an OAuth 2.0 Protected Resource through which a Client can request a new registration and manage the Metadata associated with it. 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.

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 (Jones, M. and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” October 2012.)](#RFC6750) [RFC6750] specification.

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

### 3.1.  Client Registration Request

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"]

}

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

### 3.2.  Client Registration Response

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.

The response also contains a Registration Access Token that is used by the Client to perform subsequent operations upon the resulting Client registration.

All of the response items are returned as a [JSON document (Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” July 2006.)](#RFC4627) [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

REQUIRED. Access Token that is used at the Client Configuration Endpoint to perform subsequent operations upon the Client registration.

registration\_client\_uri

REQUIRED. Location where the Access Token can be used to perform subsequent operations upon the resulting Client registration.

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):

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"]

}

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

### 3.3.  Client Registration Error Response

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 (Jones, M. and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” October 2012.)](#RFC6750) [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."

}

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

### 4.  Client Configuration Endpoint

The Client Configuration Endpoint is an OAuth 2.0 protected resource that is 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] (Jones, M. and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” October 2012.)](#RFC6750).

Operations on this endpoint are switched through the use of different HTTP methods [[RFC2616] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.)](#RFC2616). The only method defined for use at this endpoint by this specification is the HTTP GET method.

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

### 4.1.  Forming the Client Configuration Endpoint URL

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 (Client Registration Response)](#RegistrationResponse). 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.

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

### 4.2.  Client Read Request

To read the current configuration of the Client on the Authorization Server, the Client makes 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

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

### 4.3.  Client Read Response

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 is a JSON Document [[RFC4627] (Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON),” July 2006.)](#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"]

}

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

### 4.4.  Client Read Error Response

When a read error condition occurs, the Client Configuration Endpoint returns a HTTP 403 Forbidden 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

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

### 5.  "sector\_identifier\_uri" Validation

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 2.8.1 of [OpenID Connect Messages 1.0 (Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “OpenID Connect Messages 1.0,” June 2013.)](#OpenID.Messages) [OpenID.Messages]. 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" ]

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

### 6.  String Operations

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

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

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

### 7.  Validation

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

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

### 8.  Implementation Considerations

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". No other implementation considerations for implementations of Dynamic Client Registration are defined by this specification.

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

### 9.  Security Considerations

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 9.1 (TLS Requirements)](#TLS_requirements) 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.

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

### 9.1.  TLS Requirements

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

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

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

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

### 10.  IANA Considerations

This document makes no requests of IANA.

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

### 11.  References

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

### 11.1. Normative References

|  |  |
| --- | --- |
| **[JWA]** | Jones, M., “[JSON Web Algorithms (JWA)](http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms),” draft-ietf-jose-json-web-algorithms (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms-11)). |
| **[JWE]** | Jones, M., Rescorla, E., and J. Hildebrand, “[JSON Web Encryption (JWE)](http://tools.ietf.org/html/draft-ietf-jose-json-web-encryption),” draft-ietf-jose-json-web-encryption (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-jose-json-web-encryption-11)). |
| **[JWK]** | Jones, M., “[JSON Web Key (JWK)](http://tools.ietf.org/html/draft-ietf-jose-json-web-key),” draft-ietf-jose-json-web-key (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-jose-json-web-key-11)). |
| **[JWS]** | Jones, M., Bradley, J., and N. Sakimura, “[JSON Web Signature (JWS)](http://tools.ietf.org/html/draft-ietf-jose-json-web-signature),” draft-ietf-jose-json-web-signature (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-jose-json-web-signature-11)). |
| **[JWT]** | Jones, M., Bradley, J., and N. Sakimura, “[JSON Web Token (JWT)](http://tools.ietf.org/html/draft-ietf-oauth-json-web-token),” draft-ietf-oauth-json-web-token (work in progress), May 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-08)). |
| **[OAuth.JWT]** | Jones, M., Campbell, B., and C. Mortimore, “[JSON Web Token (JWT) Bearer Token Profiles for OAuth 2.0](http://tools.ietf.org/html/draft-ietf-oauth-jwt-bearer),” draft-ietf-oauth-jwt-bearer (work in progress), March 2013 ([HTML](http://tools.ietf.org/html/draft-ietf-oauth-jwt-bearer-05)). |
| **[OpenID.Discovery]** | Sakimura, N., Bradley, J., Jones, M., and E. Jay, “[OpenID Connect Discovery 1.0](http://openid.net/specs/openid-connect-discovery-1_0-17.html),” June 2013. |
| **[OpenID.Messages]** | Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., and E. Jay, “[OpenID Connect Messages 1.0](http://openid.net/specs/openid-connect-messages-1_0-20.html),” June 2013. |
| **[OpenID.Session]** | Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and N. Agarwal, “[OpenID Connect Session Management 1.0](http://openid.net/specs/openid-connect-session-1_0-14.html),” June 2013. |
| **[RFC2119]** | [Bradner, S.](mailto:sob@harvard.edu), “[Key words for use in RFCs to Indicate Requirement Levels](http://tools.ietf.org/html/rfc2119),” BCP 14, RFC 2119, March 1997 ([TXT](http://www.rfc-editor.org/rfc/rfc2119.txt), [HTML](http://xml.resource.org/public/rfc/html/rfc2119.html), [XML](http://xml.resource.org/public/rfc/xml/rfc2119.xml)). |
| **[RFC2246]** | [Dierks, T.](mailto:tdierks@certicom.com) and [C. Allen](mailto:callen@certicom.com), “[The TLS Protocol Version 1.0](http://tools.ietf.org/html/rfc2246),” RFC 2246, January 1999 ([TXT](http://www.rfc-editor.org/rfc/rfc2246.txt)). |
| **[RFC2616]** | [Fielding, R.](mailto:fielding@ics.uci.edu), [Gettys, J.](mailto:jg@w3.org), [Mogul, J.](mailto:mogul@wrl.dec.com), [Frystyk, H.](mailto:frystyk@w3.org), [Masinter, L.](mailto:masinter@parc.xerox.com), [Leach, P.](mailto:paulle@microsoft.com), and [T. Berners-Lee](mailto:timbl@w3.org), “[Hypertext Transfer Protocol -- HTTP/1.1](http://tools.ietf.org/html/rfc2616),” RFC 2616, June 1999 ([TXT](http://www.rfc-editor.org/rfc/rfc2616.txt), [PS](http://www.rfc-editor.org/rfc/rfc2616.ps), [PDF](http://www.rfc-editor.org/rfc/rfc2616.pdf), [HTML](http://xml.resource.org/public/rfc/html/rfc2616.html), [XML](http://xml.resource.org/public/rfc/xml/rfc2616.xml)). |
| **[RFC3986]** | [Berners-Lee, T.](mailto:timbl@w3.org), [Fielding, R.](mailto:fielding@gbiv.com), and [L. Masinter](mailto:LMM@acm.org), “[Uniform Resource Identifier (URI): Generic Syntax](http://tools.ietf.org/html/rfc3986),” STD 66, RFC 3986, January 2005 ([TXT](http://www.rfc-editor.org/rfc/rfc3986.txt), [HTML](http://xml.resource.org/public/rfc/html/rfc3986.html), [XML](http://xml.resource.org/public/rfc/xml/rfc3986.xml)). |
| **[RFC4627]** | Crockford, D., “[The application/json Media Type for JavaScript Object Notation (JSON)](http://tools.ietf.org/html/rfc4627),” RFC 4627, July 2006 ([TXT](http://www.rfc-editor.org/rfc/rfc4627.txt)). |
| **[RFC5246]** | Dierks, T. and E. Rescorla, “[The Transport Layer Security (TLS) Protocol Version 1.2](http://tools.ietf.org/html/rfc5246),” RFC 5246, August 2008 ([TXT](http://www.rfc-editor.org/rfc/rfc5246.txt)). |
| **[RFC5646]** | Phillips, A. and M. Davis, “[Tags for Identifying Languages](http://tools.ietf.org/html/rfc5646),” BCP 47, RFC 5646, September 2009 ([TXT](http://www.rfc-editor.org/rfc/rfc5646.txt)). |
| **[RFC6125]** | Saint-Andre, P. and J. Hodges, “[Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)](http://tools.ietf.org/html/rfc6125),” RFC 6125, March 2011 ([TXT](http://www.rfc-editor.org/rfc/rfc6125.txt)). |
| **[RFC6749]** | Hardt, D., “[The OAuth 2.0 Authorization Framework](http://tools.ietf.org/html/rfc6749),” RFC 6749, October 2012 ([TXT](http://www.rfc-editor.org/rfc/rfc6749.txt)). |
| **[RFC6750]** | Jones, M. and D. Hardt, “[The OAuth 2.0 Authorization Framework: Bearer Token Usage](http://tools.ietf.org/html/rfc6750),” RFC 6750, October 2012 ([TXT](http://www.rfc-editor.org/rfc/rfc6750.txt)). |
| **[USA15]** | [Davis, M.](mailto:markdavis@google.com), [Whistler, K.](mailto:ken@unicode.org), and M. Dürst, “Unicode Normalization Forms,” Unicode Standard Annex 15, 09 2009. |

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

### 11.2. Informative References

|  |  |
| --- | --- |
| **[I-D.ietf-oauth-dyn-reg]** | Richer, J., Bradley, J., Jones, M., and M. Machulak, “[OAuth 2.0 Dynamic Client Registration Protocol](http://tools.ietf.org/html/draft-ietf-oauth-dyn-reg-11),” draft-ietf-oauth-dyn-reg-11 (work in progress), May 2013 ([TXT](http://www.ietf.org/internet-drafts/draft-ietf-oauth-dyn-reg-11.txt)). |

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

### Appendix A.  Acknowledgements

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.

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

### Appendix B.  Notices

Copyright (c) 2013 The OpenID Foundation.

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

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

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

### Appendix C.  Document History

[[ To be removed from the final specification ]]

-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.

-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 (Richer, J., Bradley, J., Jones, M., and M. Machulak, “OAuth 2.0 Dynamic Client Registration Protocol,” May 2013.)](" \l "I-D.ietf-oauth-dyn-reg) [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.

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

### Authors' Addresses

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