



**International
Standard**

ISO/IEC 26133

**Information technology — OpenID
connect — OpenID connect dynamic
client registration 1.0 incorporating
errata set 2**

**First edition
2024-10**

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 26133:2024

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 26133:2024



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2024

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and <https://patents.iec.ch>. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by the OpenID Foundation (OIDF) (as OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 2) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

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 defines how an OpenID Connect Relying Party can dynamically register with the End-User's OpenID Provider, providing information about itself to the OpenID Provider, and obtaining information needed to use it, including the OAuth 2.0 Client ID for this Relying Party.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 26133:2024

Table of Contents

- 1. Introduction**
 - 1.1. Requirements Notation and Conventions**
 - 1.2. Terminology**
- 2. Client Metadata**
 - 2.1. Metadata Languages and Scripts**
- 3. Client Registration Endpoint**
 - 3.1. Client Registration Request**
 - 3.2. Client Registration Response**
 - 3.3. Client Registration Error Response**
- 4. Client Configuration Endpoint**
 - 4.1. Forming the Client Configuration Endpoint URL**
 - 4.2. Client Read Request**
 - 4.3. Client Read Response**
 - 4.4. Client Read Error Response**
- 5. "sector_identifier_uri" Validation**
- 6. String Operations**
- 7. Validation**
- 8. Implementation Considerations**
 - 8.1. Compatibility Notes**
 - 8.2. Implementation Notes on Stateless Dynamic Client Registration**
- 9. Security Considerations**
 - 9.1. Impersonation**
 - 9.2. Native Code Leakage**
 - 9.3. TLS Requirements**
- 10. IANA Considerations**
 - 10.1. OAuth Dynamic Client Registration Metadata Registration**
 - 10.1.1. Registry Contents**
 - 10.2. OAuth Token Endpoint Authentication Methods Registration**
 - 10.2.1. Registry Contents**
- 11. References**
 - 11.1. Normative References**
 - 11.2. Informative References**

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 26133:2024

Information technology — OpenID Connect — OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 2

1. Introduction

TOC

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 [\[RFC6749\]](#) 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.

In order for an OpenID Connect Relying Party to utilize OpenID Connect services for an End-User, the RP needs to register with the OpenID Provider to provide the OP information about itself and to obtain information needed to use it, including an OAuth 2.0 Client ID. This specification describes how an RP can register with an OP, and how registration information for the RP can be retrieved.

The previous versions of this specification are:

- [OpenID Connect Registration 1.0 incorporating errata set 1](#) [OpenID.Registration.Errata1]
- [OpenID Connect Registration 1.0 \(final\)](#) [OpenID.Registration.Final]

1.1. Requirements Notation and Conventions

TOC

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [RFC2119].

In the .txt version of this specification, 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. In the HTML version of this specification, values to be taken literally are indicated by the use of `this fixed-width font`.

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", "Authorization Code", "Authorization Endpoint", "Authorization Server", "Client", "Client Authentication", "Client Identifier", "Client Secret", "Grant Type", "Protected Resource", "Redirection URI", "Refresh Token", "Response Type", and "Token Endpoint" defined by [OAuth 2.0](#) [RFC6749], the terms "JSON Web Token (JWT)" and "Nested JWT" defined by [JSON Web Token \(JWT\)](#) [JWT], the term "Base64url Encoding" defined by [JSON Web Signature \(JWS\)](#) [JWS], and the terms defined by [OpenID Connect Core 1.0](#) [OpenID.Core].

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. 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 by the Client. 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 [\[RFC8259\]](#) 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 Type described in OAuth 2.0 Section 4.1.
- `implicit`: The Implicit Grant Type described in OAuth 2.0 Section 4.2.
- `refresh_token`: The Refresh Token Grant Type described in OAuth 2.0 Section 6.

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`
- `id_token token`: `implicit`
- `code id_token`: `authorization_code, implicit`
- `code token`: `authorization_code, implicit`
- `code id_token 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 omitted, 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 loopback URLs using the `http` scheme; loopback URLs use `localhost` or the IP loopback literals `127.0.0.1` or `:::1` as the hostname. Authorization Servers MAY place additional constraints on Native Clients. Authorization Servers MAY reject Redirection URI values using the `http`

scheme, other than the loopback case for 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 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](#).

`jwtks_uri`

OPTIONAL. URL for the Client's JWK Set [\[JWK\]](#) document, which MUST use the `https` scheme. 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` (public key use) parameter value is REQUIRED for all keys in the referenced JWK Set to indicate each key's intended usage. Although some algorithms allow the same key to be used for both signatures and encryption, doing so is NOT RECOMMENDED, as it is less secure. The JWK `x5c` parameter MAY be used to provide X.509 representations of keys provided. When used, the bare key values MUST still be present and MUST match those in the certificate. The JWK Set MUST NOT contain private or symmetric key values.

`jwtks`

OPTIONAL. Client's JWK Set [\[JWK\]](#) document, passed by value. The semantics of the `jwtks` parameter are the same as the `jwtks_uri` parameter, other than that the JWK Set is passed by value, rather than by reference. This parameter is intended only to be used by Clients that, for some reason, are unable to use the `jwtks_uri` parameter, for instance, by native applications that might not have a location to host the contents of the JWK Set. If a Client can use `jwtks_uri`, it MUST NOT use `jwtks`. One significant downside of `jwtks` is that it does not enable key rotation (which `jwtks_uri` does, as described in Section 10 of [OpenID Connect Core 1.0](#) [OpenID.Core]). The `jwtks_uri` and `jwtks` parameters MUST NOT be used together. The JWK Set MUST NOT contain private or symmetric key values.

`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](#). Providers that use pairwise `sub` (subject) values SHOULD utilize the `sector_identifier_uri` value provided in the Subject Identifier calculation for pairwise identifiers.

subject_type

OPTIONAL. `subject_type` requested for responses to this Client. The `subject_types_supported` discovery parameter contains a list of the supported `subject_type` values for the OP. Valid types include `pairwise` and `public`.

id_token_signed_response_alg

OPTIONAL. JWS `alg` algorithm [\[JWA\]](#) REQUIRED for signing the ID Token issued to this Client. The value `none` MUST NOT be used as the ID Token `alg` value unless the Client uses only Response Types that return no ID Token from the Authorization Endpoint (such as when only using the Authorization Code Flow). The default, if omitted, 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. If this is requested, the response will be signed then encrypted, with the result being a Nested JWT, as defined in [\[JWT\]](#). The default, if omitted, is that no encryption is performed.

id_token_encrypted_response_enc

OPTIONAL. JWE `enc` algorithm [\[JWA\]](#) REQUIRED for encrypting the ID Token issued to this Client. If `id_token_encrypted_response_alg` is specified, the default `id_token_encrypted_response_enc` value is `A128CBC-HS256`. When `id_token_encrypted_response_enc` is included, `id_token_encrypted_response_alg` MUST also be provided.

userinfo_signed_response_alg

OPTIONAL. JWS `alg` algorithm [\[JWA\]](#) REQUIRED for signing UserInfo Responses. If this is specified, the response will be [JWT](#) [JWT] serialized, and signed using JWS. The default, if omitted, is for the UserInfo Response to return the Claims as a UTF-8 [\[RFC3629\]](#) encoded JSON object using the `application/json` content-type.

userinfo_encrypted_response_alg

OPTIONAL. [JWE](#) [JWE] alg algorithm [JWA](#) REQUIRED for encrypting UserInfo Responses. If both signing and encryption are requested, the response will be signed then encrypted, with the result being a Nested JWT, as defined in [JWT](#). The default, if omitted, is that no encryption is performed.

userinfo_encrypted_response_enc

OPTIONAL. JWE enc algorithm [JWA](#) REQUIRED for encrypting UserInfo Responses. If `userinfo_encrypted_response_alg` is specified, the default `userinfo_encrypted_response_enc` value is A128CBC-HS256. When `userinfo_encrypted_response_enc` is included, `userinfo_encrypted_response_alg` MUST also be provided.

request_object_signing_alg

OPTIONAL. [JWS](#) [JWS] alg algorithm [JWA](#) that MUST be used for signing Request Objects sent to the OP. All Request Objects from this Client MUST be rejected, if not signed with this algorithm. Request Objects are described in Section 6.1 of [OpenID Connect Core 1.0](#) [OpenID.Core]. This algorithm MUST be used both when the Request Object is passed by value (using the `request` parameter) and when it is passed by reference (using the `request_uri` parameter). Servers SHOULD support RS256. The value `none` MAY be used. The default, if omitted, is that any algorithm supported by the OP and the RP MAY be used.

request_object_encryption_alg

OPTIONAL. [JWE](#) [JWE] alg algorithm [JWA](#) the RP is declaring that it may use for encrypting Request Objects sent to the OP. This parameter SHOULD be included when symmetric encryption will be used, since this signals to the OP that a `client_secret` value needs to be returned from which the symmetric key will be derived, that might not otherwise be returned. The RP MAY still use other supported encryption algorithms or send unencrypted Request Objects, even when this parameter is present. If both signing and encryption are requested, the Request Object will be signed then encrypted, with the result being a Nested JWT, as defined in [JWT](#). The default, if omitted, is that the RP is not declaring whether it might encrypt any Request Objects.

request_object_encryption_enc

OPTIONAL. `JWE_enc` algorithm [\[JWA\]](#) the RP is declaring that it may use for encrypting Request Objects sent to the OP. If `request_object_encryption_alg` is specified, the default `request_object_encryption_enc` value is `A128CBC-HS256`. When `request_object_encryption_enc` is included, `request_object_encryption_alg` MUST also be provided.

token_endpoint_auth_method

OPTIONAL. Requested Client Authentication method for the Token Endpoint. The options are `client_secret_post`, `client_secret_basic`, `client_secret_jwt`, `private_key_jwt`, and `none`, as described in Section 9 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 MUST be rejected, if the JWT is not signed with this algorithm. Servers SHOULD support `RS256`. The value `none` MUST NOT be used. The default, if omitted, is that any algorithm supported by the OP and the RP MAY 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. If omitted, no default Maximum Authentication Age is specified.

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`. (If this is `false`, the `auth_time` Claim can still be dynamically requested as an individual Claim for the ID Token using the `claims` request

parameter described in Section 5.5.1 of [OpenID Connect Core 1.0](#) [OpenID.Core].) If omitted, the default value is `false`.

`default_acr_values`

OPTIONAL. Default requested Authentication Context Class Reference values. Array of strings that specifies the default `acr` values that the OP 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 the OP. Values specified in the `acr_values` request parameter or an individual `acr` Claim request override these default values.

`initiate_login_uri`

OPTIONAL. URI using the `https` scheme that a third party can use to initiate a login by the RP, as specified in Section 4 of [OpenID Connect Core 1.0](#) [OpenID.Core]. 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 RP for use at the OP. These URLs MUST use the `https` scheme unless the target Request Object is signed in a way that is verifiable by the OP. 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.

Additional Client Metadata parameters MAY also be used. Some are defined by other specifications, such as [OpenID Connect RP-Initiated Logout 1.0](#) [OpenID.RPInitiated].

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 5.2 (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. The Client Registration Endpoint SHOULD support the use of [Cross-Origin Resource Sharing \(CORS\)](#) [CORS] and/or other methods as appropriate to enable JavaScript Clients and other Browser-Based Clients to access it.

To support open Dynamic 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 at 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` with the parameters represented as top-level members of the root JSON object.

The following is a non-normative example registration 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":
    "□□□□□□□□",
  "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": "RSA-OAEP-256",
  "userinfo_encrypted_response_enc": "A128CBC-HS256",
  "contacts": ["ve7jtb@example.org",
"mary@example.org"],
  "request_uris":
  ["https://client.example.org/rf.txt
   #qpXaRLh_n93TTR9F252ValdatUQvQiJi5BDub2BezrA"]
}

```

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, other than the `redirect_uris` value, 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 sent by the Client that it does not understand.

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

A successful response SHOULD use the HTTP 201 Created status code and return a JSON document [RFC8259] using the `application/json` content type with the following fields and the Client Metadata parameters 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. The same Client Secret value MUST NOT be assigned to multiple Clients. This value is used by Confidential Clients to authenticate to the Token Endpoint, as described in Section 2.3.1 of OAuth 2.0, and

for the derivation of symmetric encryption key values, as described in Section 10.2 of [OpenID Connect Core 1.0](#) [OpenID.Core]. It is not needed for Clients selecting a `token_endpoint_auth_method` of `private_key_jwt` unless symmetric encryption will be used.

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. This URL MUST use the `https` scheme. 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. Its value is a JSON number representing the number of seconds from 1970-01-01T00:00:00Z 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. Its value is a JSON number representing the number of seconds from 1970-01-01T00:00:00Z as measured in UTC until the date/time.

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

```
HTTP/1.1 201 Created
Content-Type: application/json
Cache-Control: no-store

{
  "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=s
6BhdRkqt3",
  "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":
    "□□□□□□□□",
  "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": "RSA-OAEP-256",
  "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 Bad Request status code including a JSON object describing the error in the response body.

The JSON object describing the error contains two members:

error

Error code.

error_description

Additional text description of the error for debugging.

Other members MAY also be used.

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.

Other error codes MAY also be used.

The following is a non-normative example error response:

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

{
  "error": "invalid_redirect_uri",
  "error_description": "One or more redirect_uri values
are invalid"
}
```

4. 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]. The Client Configuration Endpoint SHOULD support the use of [Cross-Origin Resource Sharing \(CORS\)](#) [CORS] and/or other methods as appropriate to enable JavaScript Clients and other Browser-Based Clients to access it.

Operations on this endpoint are switched through the use of different HTTP methods [RFC7231]. The only method defined for use at this endpoint by this specification is the HTTP `GET` method.

4.1. 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.

4.2. 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. This operation SHOULD be idempotent -- not causing changes to the Client configuration.

The following is a non-normative example read 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
```

4.3. 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. Note that some values, including the `client_secret` value, might have been updated since the initial registration. The mechanisms for such updates are beyond the scope of this specification. However, since Read operations are intended to be idempotent, the Client Read Request itself SHOULD NOT cause changes to the Client's registered Metadata values.

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

A successful response SHOULD use the HTTP 200 OK status code and return a JSON document [RFC8259] using the `application/json` content type with the Client Metadata values as top-level members of the root JSON object.

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

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

{
  "client_id": "s6BhdRkqt3",
  "client_secret":
    "Oylyac56ijpAQ7G5ZZGL7MMQ6Ap6mEeuhSTFVps2N4Q",
  "client_secret_expires_at": 17514165600,
  "registration_client_uri":

"https://server.example.com/connect/register?client_id=s
6BhdRkqt3",
  "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":
      "□□□□□□□□",
    "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": "RSA-OAEP-256",
    "userinfo_encrypted_response_enc": "A128CBC-HS256",
    "contacts": ["ve7jtb@example.org",
"mary@example.org"],
    "request_uris":
      ["https://client.example.org/rf.txt
       #qpXaRLh_n93TTR9F252ValdatUQvQiJi5BDub2BeznA"]
  }

```

4.4. 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].

If the Client does not exist on this server, the Client is invalid, or the Registration Access Token used is invalid, the server MUST respond with the HTTP 401 Unauthorized status code. If the Client does not have permission to read its record, the server MUST return an HTTP 403 Forbidden. Note that for security reasons, to inhibit brute force attacks, endpoints MUST NOT return the HTTP 404 Not Found status code.

The following is a non-normative example error response:

```

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer error="invalid_token",
  error_description="The access token expired"
Cache-Control: no-store

```

5. "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 8.1 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. This MUST be validated at registration time; there is no requirement for the OP to retain the contents of this JSON file or to retrieve or revalidate its contents in the future.

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

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

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

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

6. 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 [\[UNICODE\]](#) 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.

7. 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.

8. 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".

This specification is compatible with the [OAuth 2.0 Dynamic Client Registration Protocol](#) [RFC7591]. Implementations MAY use additional Client Metadata values defined by [\[RFC7591\]](#), such as `software_statement`, should they facilitate their use cases.

Implementations wanting to support additional operations defined in [OAuth 2.0 Dynamic Client Registration Management Protocol](#) [RFC7592], such as Update, can do so using that specification, while being mindful that the specification is experimental.

8.1. Compatibility Notes

TOC

NOTE: Potential compatibility issues that were previously described in the original version of this specification have since been addressed.

8.2. Implementation Notes on 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, because issuing a Registration Access Token might require per-Client state at the Authorization Server. In that case, no Client Configuration Endpoint or Registration Access Token will be returned by the initial registration of the Client.

9. 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 9.3](#) for more information on using TLS.

9.1. Impersonation

TOC

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 are logging in to the legitimate RP. An OP could also warn if the domain/site of the logo does not match the domain/site of registered Redirection URIs. An OP can also make warnings against untrusted RPs in all cases, especially if they are 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`.

9.2. 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.

9.3. 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. Implementations SHOULD follow the guidance in BCP 195 [\[RFC8996\]](#) [\[RFC9325\]](#), which provides recommendations and requirements for improving the security of deployed services that use TLS.

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].

10. IANA Considerations

TOC

10.1. OAuth Dynamic Client Registration Metadata Registration

TOC

This specification registers the following client metadata definitions in the IANA "OAuth Dynamic Client Registration Metadata" registry [\[IANA.OAuth.Parameters\]](#) established by [\[RFC7591\]](#).

10.1.1. Registry Contents

TOC

- Client Metadata Name: `application_type`
- Client Metadata Description: Kind of the application -- "native" or "web"
- Change Controller: OpenID Foundation Artifact Binding Working Group - openid-specs-ab@lists.openid.net
- Specification Document(s): [Section 2](#) of this specification
- Client Metadata Name: `sector_identifier_uri`
- Client Metadata Description: URL using the `https` scheme to be used in calculating Pseudonymous Identifiers by the OP