

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 31, 2013

J. Richer, Ed.  
The MITRE Corporation  
J. Bradley  
Ping Identity  
M. Jones  
Microsoft  
M. Machulak  
Newcastle University  
November 27, 2012

**OAuth Dynamic Client Registration Protocol**  
**draft-ietf-oauth-dyn-reg-02**

**Abstract**

This specification defines an endpoint and protocol for dynamic registration of OAuth Clients at an Authorizaiton Server.

**Status of this Memo**

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 31, 2013.

**Copyright Notice**

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4](#).e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">3</a>
<a href="#">1.1.</a>	<a href="#">Notational Conventions . . . . .</a>	<a href="#">3</a>
<a href="#">1.2.</a>	<a href="#">Terminology . . . . .</a>	<a href="#">3</a>
<a href="#">1.3.</a>	<a href="#">Requirements . . . . .</a>	<a href="#">4</a>
1.3.1.	The client needs to be uniquely identifiable by the authorization server . . . . .	<a href="#">4</a>
1.3.2.	The authorization server must collect metadata about a client for later user interaction . . . . .	<a href="#">4</a>
1.3.3.	The authorization server should have the option of strongly authenticating the client and its metadata .	<a href="#">4</a>
1.3.4.	Dynamic client registration must be possible from both web-server applications and applications with other capabilities and limitations, such as native applications . . . . .	<a href="#">4</a>
<a href="#">1.3.5.</a>	<a href="#">Transaction integrity must be ensured . . . . .</a>	<a href="#">5</a>
<a href="#">2.</a>	<a href="#">Client Metadata . . . . .</a>	<a href="#">5</a>
<a href="#">3.</a>	<a href="#">Client Registration Endpoint . . . . .</a>	<a href="#">7</a>
<a href="#">3.1.</a>	<a href="#">Client Registration Request . . . . .</a>	<a href="#">9</a>
<a href="#">3.2.</a>	<a href="#">Client Registration Response . . . . .</a>	<a href="#">10</a>
<a href="#">3.3.</a>	<a href="#">Client Update Request . . . . .</a>	<a href="#">11</a>
<a href="#">3.4.</a>	<a href="#">Client Update Response . . . . .</a>	<a href="#">12</a>
<a href="#">3.5.</a>	<a href="#">Rotate Secret Request . . . . .</a>	<a href="#">12</a>
<a href="#">3.6.</a>	<a href="#">Rotate Secret Response . . . . .</a>	<a href="#">13</a>
<a href="#">3.7.</a>	<a href="#">Client Registration Error Response . . . . .</a>	<a href="#">14</a>
<a href="#">4.</a>	<a href="#">IANA Considerations . . . . .</a>	<a href="#">15</a>
<a href="#">5.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">15</a>
<a href="#">6.</a>	<a href="#">Acknowledgments . . . . .</a>	<a href="#">16</a>
<a href="#">7.</a>	<a href="#">Document History . . . . .</a>	<a href="#">16</a>
<a href="#">8.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">17</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">18</a>



## **1. Introduction**

In some use-case scenarios, it is desirable or necessary to allow OAuth clients to obtain authorization from an OAuth authorization server without the two parties having previously interacted. Nevertheless, in order for the authorization server to accurately represent to end-users which client is seeking authorization to access the end-user's resources, a method for automatic and unique registration of clients is needed. The OAuth2 authorization framework does not define how the relationship between the Client and the Authorization Server is initialized, or how a given client is assigned a unique Client Identifier. Historically, this has happened out-of-band from the OAuth protocol. This draft provides a mechanism for a client to register itself with the Authorization Server, which can be used to dynamically provision a Client Identifier, and optionally a Client Secret.

As part of the registration process, this specification also defines a mechanism for the client to present the Authorization Server with a set of metadata, such as a display name and icon to be presented to the user during the authorization step. This draft provides a method for the client to register and update this information over time.

### **1.1. Notational 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 [[RFC2119](#)].

Unless otherwise noted, all the protocol parameter names and values are case sensitive.

### **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 [[RFC6750](#)].

This specification defines the following additional terms:

- o Client Registration Endpoint: The OAuth 2.0 Endpoint through which a Client can request new registration and manage the metadata associated with it.
- o Registration Access Token: An OAuth 2.0 Bearer Token issued by the Authorization Server through the Client Registration Endpoint



which is used by the Client to authenticate itself during update and secret rotation operations.

### **1.3. Requirements**

[[ Following are proposed requirements for dynamic client registration. This section is intended for discussion and will likely be removed in the final draft. ]]

#### **1.3.1. The client needs to be uniquely identifiable by the authorization server**

In order for an authorization server to do proper user-delegated authorization and prevent unauthorized access it must be able to identify clients uniquely. As is done today in OAuth, the client identifier (and optional secret) should thus be issued by the authorization server and not simply accepted as proposed by the client.

#### **1.3.2. The authorization server must collect metadata about a client for later user interaction**

In order for the authorization server to describe a client to an end-user in an authorization step it needs information about the client. This can be the client name at a minimum, but today servers usually request at least a description, a homepage URL, and an icon when doing manual registration.

#### **1.3.3. The authorization server should have the option of strongly authenticating the client and its metadata**

In order to prevent spoofing of clients and enable dynamic building of strong trust relationships, the authorization server should have the option to verify the provided information. This might be solved using message signature verification.

#### **1.3.4. Dynamic client registration must be possible from both web-server applications and applications with other capabilities and limitations, such as native applications**

Each instance of a native application (that is, the specific instance running on each device) that is installed and run by the same user may need the option of getting a unique client identifier. In this case, there are implications around gathering and displaying enough information to ensure that the end-user is delegating authorization to the intended application. The registration protocol should be simple and flexible enough to allow for multiple types of applications.



### **1.3.5. Transaction integrity must be ensured**

When a client sends information to a server endpoint, it might take time for this data to propagate through big server installations that spread across various data centers. Care needs to be taken that subsequent interactions with the user after the registration process, such as an authorization request, show the correct data.

## **2. Client Metadata**

Clients generally have an array of 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.

Extensions and profiles of this specification MAY expand this list, but MUST at least accept all parameters on this list. The Authorization Server MUST ignore any additional parameters sent by the Client that it does not understand.

`redirect_uris`

REQUIRED A space-delimited list of redirect URIs.

`client_name`

RECOMMENDED. Human-readable name of the Client to be presented to the user.

`client_url`

RECOMMENDED. URL of the homepage of the client. If present, the server SHOULD display this URL to the end user.

`logo_url`

OPTIONAL. URL that references a logo for the Client application. If present, the server SHOULD display this image to the end user during approval.

`contacts`

OPTIONAL. Space delimited list of email addresses for people responsible for this client. The Authorization Server MAY use these addresses available to end users for support queries. An Authorization Server MAY use these email addresses as identifiers for an administrative page for this client.





`tos_url`

OPTIONAL. URL that points to a human-readable Terms of Service for the Client. The Authorization Server SHOULD display this URL to the End-User if it is given.

`token_endpoint_auth_method`

OPTIONAL. The requested authentication type for the Token Endpoint. Valid values are:

- \* "none" this is a public client as defined in OAuth 2.0 and does not have a client secret
- \* "client\_secret\_post" the client uses the HTTP POST parameters defined in OAuth2.0 [section 2.3.1](#)
- \* "client\_secret\_basic" the client uses HTTP Basic defined in OAuth 2.0 [section 2.3.1](#)
- \* "client\_secret\_jwt" the client uses the JWT Assertion profile with a symmetric secret issued by the server
- \* `_private_key_jwt_` the client uses the JWT Assertion profile with its own private key

Other Authentication methods may be defined by extension. 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 [[RFC6749](#)].

`policy_url`

OPTIONAL. A URL location that the Client provides to the End-User to read about the how the profile data will be used. The Authorization Server SHOULD display this URL to the End-User if it is given.

`jwk_url`

OPTIONAL. URL for the Client's JSON Web Key [[JWK](#)] document that is used for signing Token Endpoint Requests. If `jwk_encryption_url` is not provided, the key at `jwk_url` is also used as the key to encrypt responses to the Client. If the Client registers both "`x509_url`" and "`jwk_url`", the keys contained in both formats MUST be the same.

`jwk_encryption_url`

OPTIONAL. URL for the Client's JSON Web Key [[JWK](#)] that is used to encrypt any responses to the Client. If the Client registers both "`jwk_encryption_url`" and "`x509_encryption_url`", the keys contained in both formats MUST be the same.



**x509\_url**

OPTIONAL. URL for the Client's PEM encoded X.509 Certificate or Certificate chain that is used for signing Token Endpoint Requests. If "x509\_encryption\_url" is not provided, "x509\_url" it is also used to encrypt responses to the Client. If the Client registers both "x509\_url" and "jwk\_url", the keys contained in both formats MUST be the same.

**x509\_encryption\_url**

OPTIONAL. URL for the Client's PEM encoded X.509 Certificate or Certificate chain that is used to encrypt the ID Token and User Info Endpoint Responses to the Client. If the Client registers both "jwk\_encryption\_url" and "x509\_encryption\_url", the keys contained in both formats SHOULD be the same.

**default\_max\_age**

OPTIONAL. Maximum age of a session in integer seconds. Specifies that the End-User must be actively authenticated if any present authentication is older than the specified number of seconds by default.

**default\_acr**

OPTIONAL. Default Authentication Context class Reference. String that specifies the default authentication context value that the Authorization server must use for processing requests from this client.

### **3. Client Registration Endpoint**

The Client Registration Endpoint is an OAuth 2.0 Endpoint defined in this document that is designed to allow a Client to register itself with the Authorization Server. The Client Registration Endpoint MUST accept HTTP POST messages with request parameters encoded in the entity body using the "application/x-www-form-urlencoded" format. The Client Registration Endpoint MUST be protected by a transport-layer security mechanism when sending requests to the Registration Endpoint. The server MUST support TLS 1.2 [RFC 5246](#) [[RFC5246](#)] and/or TLS 1.0 [[RFC2246](#)] and MAY support additional transport-layer mechanisms meeting its security requirements. When using TLS, the Client MUST perform a TLS/SSL server certificate check, per [RFC 6125](#) [[RFC6125](#)].

The Endpoint defines three operations that a client can take on it, switched by the "operation" parameter:



- o `client_register`: request that the Authorization Server generate a new Client Identifier (and optionally a Client Secret) and associate it with the set of presented metadata ([Section 2](#))
- o `client_update`: update the metadata ([Section 2](#)) associated with a Client Identifier
- o `rotate_secret`: issue a new Registration Access Token and, if applicable, a Client Secret for a Client

In order to facilitate registered clients updating their information, the Client Registration Endpoint issues a `request_access_token` for clients to securely identify themselves in future connections. As such, the Endpoint **MUST** accept requests with OAuth 2.0 Bearer Tokens [[RFC6750](#)] for these operations.

In order to support open registration and facilitate wider interoperability, the Client Registration Endpoint **SHOULD** allow `client_register` requests with no further authentication. These requests **MAY** be rate-limited to prevent a denial-of-service attack on the Client Registration Endpoint.

In addition, the Client Registration Endpoint **MAY** accept an initial authorization credential in the form of an OAuth 2.0 [[RFC6749](#)] access token in order to limit registration to only previously authorized parties. The method by which this access token is obtained by the registrant is generally out-of-band and is out of scope of this specification.

These two aspects, operation selection and client authentication, are represented by two parameters common to all operations:

`operation` **REQUIRED**. Values are `"client_register"` (for new registrations), `"rotate_secret"` to request rotation of the `"client_secret"`, and `"client_update"` (for updating parameters of an existing `"client_id"`).

`access_token` **OPTIONAL**. An OAuth2 Bearer token used to access the Client Registration Endpoint, as defined in OAuth2 Bearer. This parameter **MUST NOT** be sent if the Access Token is sent in the HTTP Authorization header as described in [Section 7.1](#) of OAuth 2.0 [[RFC6749](#)]. Access Tokens sent in the authorization header must be OAuth 2.0 Bearer Tokens [[RFC6750](#)].

Each operation takes a different parameter set, and all operations are described below.

The Client Registration Endpoint **MUST** ignore all parameters it does



not understand.

### **3.1. Client Registration Request**

This operation registers a new client to the Authorization Server. 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 request includes the two parameters described above as well as any parameters described in Client Metadata ([Section 2](#)).

operation

REQUIRED. MUST have the value "client\_register"

access\_token

OPTIONAL. used to restrict new client registration. This parameter MUST NOT be sent if the Access Token is sent in the HTTP Authorization header as described in [Section 7.1](#) of OAuth 2.0 [[RFC6749](#)]. Access Tokens sent in the authorization header must be OAuth 2.0 Bearer Tokens [[RFC6750](#)].

redirect\_uris REQUIRED

client\_name RECOMMENDED

client\_url RECOMMENDED

logo\_url OPTIONAL

contacts OPTIONAL

tos\_url OPTIONAL

token\_endpoint\_auth\_method OPTIONAL

policy\_url OPTIONAL

jwk\_url OPTIONAL

jwk\_encryption\_url OPTIONAL

x509\_url OPTIONAL

x509\_encryption\_url OPTIONAL

default\_max\_age OPTIONAL

default\_acr OPTIONAL

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





Following is a non-normative example request (with line wraps for display purposes only):

```
POST /register HTTP/1.1
```

```
Accept: application/x-www-form-urlencoded
```

```
Host: server.example.com
```

```
operation=client_register
```

```
&redirect_uri=https://client.example.org/callback
```

```
%20https://client.example.org/callback2
```

```
&client_name=My%20Example%20Client
```

```
&logo_url=https://client.example.org/logo.png
```

```
&token_endpoint_auth_type=client_secret_basic
```

```
&jwk_url=https://client.example.org/my_rsa_public_key.jwk
```

### **3.2. Client Registration Response**

Upon successful registration, the Client Registration Endpoint returns the newly-created Client Identifier and, optionally, a Client Secret. The response also contains a Registration Access Token that is to be used by the client to perform subsequent operations at this endpoint, such as `client_update` and `rotate_secret`. These items are returned as a JSON document with the following fields as top-level members of the root JSON object.

`client_id`

REQUIRED. The unique Client identifier, MUST NOT be currently valid for any other registered Client.

`client_secret`

OPTIONAL. The 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](#).

`registration_access_token`

REQUIRED. The Access token to be used by the client to perform "client\_update" and "rotate\_secret" requests.

`issued_at`

OPTIONAL. Specifies the timestamp when the Client Identifier was issued. The timestamp value MUST be a positive integer. The value is expressed in the number of seconds since January 1, 1970 00:00:00 GMT.

`expires_at`

OPTIONAL. The number of seconds from 1970-01-01T0:0:0Z as measured in UTC that the "client\_secret" will expire or "0" if they do not expire. See [RFC 3339](#) [[RFC3339](#)] for details regarding



date/times in general and UTC in particular.

Following is a non-normative example response:

HTTP/1.1 200 OK

Content-Type: application/json

Cache-Control: no-store

```
{
  "client_id": "s6BhdRkqt3",
  "client_secret":
    "cf136dc3c1fd9153029bb9c6cc9ecead918bad9887fce6c93f31185e5885805d",
  "registration_access_token": "reg-23410913-abewfq.123483",
  "expires_at": 2893276800
}
```

### **3.3. Client Update Request**

This operation updates a previously-registered client with new metadata at the Authorization Server. This request **MUST** be protected by the Registration Authorization Token associated with the Client. This request **MAY** include any fields described in Client Metadata ([Section 2](#)). The values of Client Metadata fields in this request **MUST** replace (not augment) the values previously associated with this Client. Empty values in Client Metadata **MUST** be taken as a request to clear any existing value of that field.

operation

REQUIRED, **MUST** have the value "client\_update"

access\_token

REQUIRED, unless presented in the Authorization Header as in OAuth2 Bearer [[RFC6750](#)]. The Registration Access Token that was issued during the client\_register step, or previous client\_update or rotate\_secret calls.

redirect\_uris REQUIRED

client\_name RECOMMENDED

client\_url RECOMMENDED

logo\_url OPTIONAL

contacts OPTIONAL

tos\_url OPTIONAL

token\_endpoint\_auth\_method OPTIONAL

policy\_url OPTIONAL

jwk\_url OPTIONAL

jwk\_encryption\_url OPTIONAL

x509\_url OPTIONAL



x509\_encryption\_url OPTIONAL

default\_max\_age OPTIONAL

default\_acr OPTIONAL

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

Following is a non-normative example request (with line wraps for display purposes only):

POST /register HTTP/1.1

Accept: application/x-www-form-urlencoded

Host: server.example.com

Authorization: Bearer reg-23410913-abewfq.123483

operation=client\_update

&redirect\_uri=https://client.example.org/callback

%20https://client.example.org/callback2

&client\_name=My%20Example%20

&logo\_url=https://client.example.org/logo.png

&token\_endpoint\_auth\_type=client\_secret\_basic

&jwk\_url=https://client.example.org/my\_rsa\_public\_key.jwk

### **3.4. Client Update Response**

Upon successful update, the Client Registration Endpoint returns a JSON document with the following fields as top-level members of the root JSON object.

client\_id

REQUIRED. The unique Client identifier, MUST equal the value of the client\_id returned in the original client\_register request.

Following is a non-normative example response:

HTTP/1.1 200 OK

Content-Type: application/json

Cache-Control: no-store

```
{
  "client_id": "s6BhdRkqt3",
}
```

[[ Editor's note: should this return the entire client data object, for confirmation and review, including any fields that may have been asserted by the AS? ]]

### **3.5. Rotate Secret Request**

This operation allows the client to rotate its current Registration Access Token as well as its Client Secret, if it has one.



operation REQUIRED. MUST have the value rotate\_secret

access\_token REQUIRED. The Registration Access Token that was issued during the client\_register step, or previous client\_update or rotate\_secret calls. This parameter MUST NOT be sent if the Access Token is sent in the HTTP Authorization header as described in [Section 7.1](#) of OAuth 2.0 [[RFC6749](#)]. Access Tokens sent in the authorization header must be OAuth 2.0 Bearer Tokens [[RFC6750](#)].

Following is a non-normative example request (with line wraps for display purposes only):

POST /register HTTP/1.1

Accept: application/x-www-form-urlencoded

Host: server.example.com

Authorization: Bearer reg-23410913-abewfq.123483

operation=rotate\_secret

### [3.6.](#) Rotate Secret Response

Upon successful rotation of the Registration Access Token and optionally the Client Secret, the Client Registration Endpoint returns a JSON document with the following fields as top-level members of the root JSON object.

client\_id

REQUIRED. The unique Client identifier, MUST match the client\_id issued in the original client\_register request.

client\_secret

REQUIRED if the server initially issued this Client a Client Secret, otherwise the server MUST NOT return a value. The value MUST be unique for each "client\_id".

registration\_access\_token

REQUIRED The Access token to be used by the client to perform subsequent "client\_update" and "rotate\_secret" requests.

issued\_at

OPTIONAL. Specifies the timestamp when the identifier was issued. The timestamp value MUST be a positive integer. The value is expressed in the number of seconds since January 1, 1970 00:00:00 GMT.

expires\_at

OPTIONAL. The number of seconds from 1970-01-01T0:0:0Z as measured in UTC that the "client\_secret" will expire or "0" if they do not expire. See [RFC 3339](#) [[RFC3339](#)] for details regarding





date/times in general and UTC in particular.

Following is a non-normative example response:

HTTP/1.1 200 OK

Content-Type: application/json

Cache-Control: no-store

```
{
  "client_id": "s6BhdRkqt3",
  "client_secret":
    "cf136dc3c1fd9153029bb9c6cc9eceed918bad9887fce6c93f31185e5885805d",
  "registration_access_token": "this.is.a.access.token.value.ffx83",
  "expires_at": 2893276800
}
```

The Authorization Server SHOULD discard and invalidate the Request Access Token and the Client Secret associated with this Client after successful completion of this request.

### **[3.7.](#) Client Registration Error Response**

When an OAuth error condition occurs, the Client Registration Endpoint returns an Error Response as defined in [Section 5.2](#) of the OAuth 2.0 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`

The error code, a single ASCII string.

`error_description`

The additional text description of the error for debugging.

This specification defines the following error codes:

`invalid_operation`

The value of "operation" is invalid or not supported.

`invalid_redirect_uri`

The value of one or more "redirect\_uris" is invalid.



#### invalid\_client\_metadata

The value of one of the client metadata ([Section 2](#)) fields is invalid.

Following is a non-normative example of an error response (with line wraps for display purposes only):

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

```
{
  "error": "invalid_operation",
  "error_description": "The value of the operation parameter must
    be one of client_register, rotate_secret or client_update."
}
```

## 4. IANA Considerations

This document makes no requests of IANA.

## 5. Security Considerations

[*Editor's note: Following are some security considerations taken whole from the UMA and OpenID Connect source drafts.* ]]

Since requests to the Client Registration Endpoint result in the transmission of clear-text credentials (in the HTTP request and response), the server MUST require the use of a transport-layer security mechanism when sending requests to the Registration Endpoint. The server MUST support TLS 1.2 [RFC 5246](#) [[RFC5246](#)] and/or TLS 1.0 [[RFC2246](#)] and MAY support additional transport-layer mechanisms meeting its security requirements. When using TLS, the Client MUST perform a TLS/SSL server certificate check, per [RFC 6125](#) [[RFC6125](#)].

As this endpoint is an OAuth2 Protected Resource, requests to the Registration Endpoint SHOULD have some rate limiting on failures to prevent the Registration Access Token from being disclosed through repeated access attempts.

The authorization server MUST treat all client metadata as self-asserted. A rogue Client might use the name and logo for the legitimate Client, which it is trying to impersonate. An Authorization Server needs to take steps to mitigate this phishing risk, since the logo could confuse users into thinking they're logging in to the legitimate Client. For instance, an Authorization



Server could warn if the domain/site of the logo doesn't match the domain/site of redirect URIs. An Authorization Server can also make warnings against untrusted Clients in all cases, especially if they're dynamically registered, have not been trusted by any users at the Authorization Server before.

In a situation where the Authorization Server is supporting open Client registration, it must be extremely careful with any URL provided by the Client that will be displayed to the user (e.g. "logo\_url" and "policy\_url"). A rogue Client could specify a registration request with a reference to a drive-by download in the "policy\_url". The Authorization Server should check to see if the "logo\_url" and "policy\_url" have the same host as the hosts defined in the array of "redirect\_uris".

While the Client Secret can expire, the Registration Access Token should not expire while a client is still actively registered. If this token were to expire, a Client could be left in a situation where it has no means of updating itself and must register itself anew. As the Registration Access Tokens are long-term credentials, they MUST be protected by the Client as a secret. [[ Editor's note: with the right error codes returned from client\_update, the AS could force the Client to call rotate\_secret before going forward, lessening the window for abuse of a leaked registration token. ]]

## **6. Acknowledgments**

The authors thank the OAuth Working Group, the User-Managed Access Working Group, and the OpenID Connect Working Group participants for their input to this document. In particular, the following individuals have been instrumental in their review and contribution to various versions of this document: Torsten Lodderstedt, Eve Maler, Thomas Hardjono, Christian Scholz, Nat Sakimura, George Fletcher, Amanda Anganes, and Domenico Catalano.

## **7. Document History**

[[ to be removed by RFC editor before publication as an RFC ]]

- 02

- o Reorganized contributors and references
- o Moved OAuth references to RFC



- o Reorganized model/protocol sections for clarity
- o Changed terminology to "client register" instead of "client associate"
- o Specified that client\_id must match across all subsequent requests
- o Fixed RFC2XML formatting, especially on lists
- 01
- o Merged UMA and OpenID Connect registrations into a single document
- o Changed to form-paramter inputs to endpoint
- o Removed pull-based registration
- 00
- o Imported original UMA draft specification

## **8. Normative References**

- [JWA] Jones, M., "JSON Web Algorithms", May 2012.
- [JWE] Jones, M., Rescorla, E., and J. Hildebrand, "JSON Web Encryption (JWE)", May 2012.
- [JWK] Jones, M., "JSON Web Key (JWK)", May 2012.
- [JWS] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature", May 2012.
- [JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token", May 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", [RFC 2246](#), January 1999.
- [RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", [RFC 2617](#), June 1999.





- [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", [RFC 3339](#), July 2002.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", [RFC 4627](#), July 2006.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", [RFC 5785](#), April 2010.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", [RFC 6125](#), March 2011.
- [RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), October 2012.
- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", [RFC 6750](#), October 2012.

#### Authors' Addresses

Justin Richer (editor)  
The MITRE Corporation

Phone:  
Fax:  
Email: [jricher@mitre.org](mailto:jricher@mitre.org)  
URI:

John Bradley  
Ping Identity

Email: [ve7jtb@ve7jtb.com](mailto:ve7jtb@ve7jtb.com)



Michael B. Jones  
Microsoft

Email: [mbj@microsoft.com](mailto:mbj@microsoft.com)

Maciej Machulak  
Newcastle University

Email: [m.p.machulak@ncl.ac.uk](mailto:m.p.machulak@ncl.ac.uk)

URI: <http://ncl.ac.uk/>