OAuth Dynamic Binding – Draft 00

Terminology

resource server
A server capable of accepting and responding to protected resource requests.

client
An application obtaining authorization and making protected resource requests.

authorization server
A server capable of issuing tokens after successfully authenticating the resource owner and obtaining authorization. The authorization server may be the same server as the resource server, or a separate entity.

end-user authorization endpoint
The authorization server's HTTP endpoint capable of authenticating the end-user and obtaining authorization.

token endpoint
The authorization server's HTTP endpoint capable of issuing tokens and refreshing expired tokens.

client identifier
An unique identifier issued to the client to identify itself to the authorization server. Client identifiers may have a matching secret.

client registration endpoint
The authorization server's HTTP endpoint capable of issuing client identifiers and optional client secrets.

Discovery of client registration endpoint

The client uses the [sitemeta] and [hostmeta] discovery mechanisms to learn about the URI of the client registration endpoint at the authorization server at which the client wants to register. The authorization server MUST provide a host-meta document containing:

- `Link` element with the following `rel` value of `http://oauth.net/as/registration` (REQUIRED)

```
<XRD>

<Host>http://server.example.com</Host>

<Link rel="http://oauth.net/as/registration " href="http://server.example.com/register">
<Title>Client Registration Endpoint</Title>
</Link>
```

Maciej Machulak
m.p.machulak@ncl.ac.uk

```
<!-- other content omitted -->


</XRD>
```

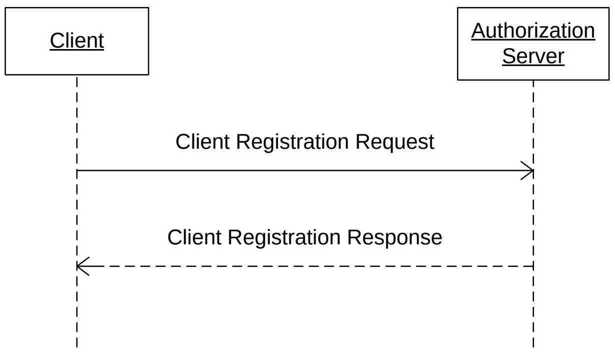
Overview

The dynamic client registration process results in the client being provisioned the client identifier and an optional client secret. This process differs in the way the client interacts with the authorization server but always results in the client receiving both client identifier and optional client secret or an error response. This specification defines two different flows for obtaining information from the client that is required before provision client with its credentials. These flows are the Push Client Registration and Pull Client Registration. This specification defines both flows.

Push Client Registration

The Push Client Registration Flow works as following:

1. The client sends required parameters to the client registration endpoint. The client MUST send its name, description and redirection URI and MAY send a URI to its icon.
2. The authorization server checks the data and returns a client identifier and an optional client secret.

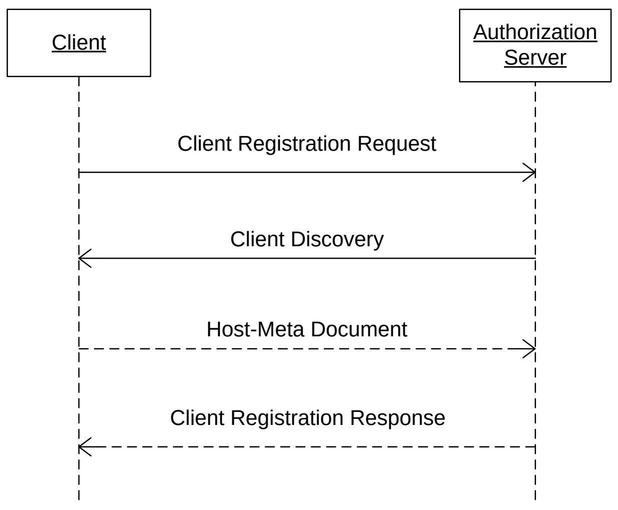


Pull Client Registration

The Pull Client Registration Flow works as following:

1. The client sends its URI to the client registration endpoint.
2. The authorization server uses the [sitemeta] and [hostmeta] discovery mechanisms on this URI in order to retrieve the host-meta document describing the client. The host-meta document MUST contain the client's name, description and redirection URI and MAY contain a URI to the client's icon.

Maciej Machulak
m.p.machulak@ncl.ac.uk



The authorization server must support both flows but the client is only required to support any of these two flows.

Push Client Registration

Client Registration Request

The client sends a JSON formatted document to the client registration endpoint. The client includes the following parameters in the request:

- type (REQUIRED) – this parameter must be set to “push”
- name (REQUIRED)
- url (REQUIRED)
- description (REQUIRED)
- icon (OPTIONAL)

The client MAY include additional information in the request and the authorization server MAY ignore this information.

TBS: type parameter names should be different

For example, the client sends:

```
POST /register HTTP/1.1
Host: server.example.com
Content-Type: application/json

{
   “type”:“push”,
   “name”:”Online Photo Gallery”,
   “url”:”http://onlinephotogallery.com”,
   “description”:”Not only uploading, but also editing capabilities!”,
   “icon”:”http://onlinephotogallery.com/icon.png”
}
```

The parameters are included in the entity body of the HTTP request using the "application/json" media type as defined by [RFC4627]. The parameters are serialized into a JSON structure by adding each parameter at the highest structure level. Parameter names and string values are included as JSON strings.

Client Registration Response

After receiving and verifying information received from the client, the authorization server issues the client identifier and an optional client secret, and constructs the response by adding the following parameters to the entity body of the HTTP response with a 200 status code (OK):

- client_id (REQUIRED)
- client_secret (OPTIONAL)

The parameters are included in the entity body of the HTTP response using the "application/json" media type as defined by [RFC4627]. The parameters are serialized into a JSON structure by adding each parameter at the highest structure level. Parameter names and string values are included as JSON strings.

The authorization server MUST include the HTTP “Cache-Control” response header field with a value of “no-store” in any response containing client_secret.

For example:

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

{
  "client_id":"5UO9XcL4TQTa",
  "client_secret":”WdRKN3zeTc20”
}
```

Error Message

If the request for registration is invalid or unauthorized, the authorization server constructs the response by adding the following parameters to the entity body of the HTTP response with a 400 status code (Bad Request) using the “application/json” media type:

- error (REQUIRED)
- description (OPTIONAL)

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

{
   "error":"unauthorized_client",
   "description":"This client is not on the white list of this Authorization Server"
}
```

TBS: We need standardized error codes.

Pull Client Registration

Client Registration Request

The client sends a JSON formatted document to the client registration endpoint. The client includes the following parameters in the request:

- type (REQUIRED) – this parameter must be set to “pull”
- url (REQUIRED)

The client MAY include additional information in the request and the authorization server MAY ignore this information.

For example, the client sends:

```
POST /register HTTP/1.1
Host: server.example.com
Content-Type: application/json

{
   “type”:“pull”,
   “url”:”http://onlinephotogallery.com”,
}
```

The parameters are included in the entity body of the HTTP request using the "application/json" media type as defined by [RFC4627]. The parameters are serialized into a JSON structure by adding each parameter at the highest structure level. Parameter names and string values are included as JSON strings.

Client Discovery

The authorization server should evaluate this request and MAY perform a [sitemeta] and [hostmeta] discovery mechanism on the provided URL to the host-meta document for the client. For example, the authorization server sends:

```
GET /.well-known/host-meta HTTP/1.1
Host: server.example.com
```

The authorization server retrieves the host-meta document which MUST contain:

- `Property` element with the following `type` value of http://oauth.net/client/name (REQUIRED)
- `Property` element with the following `type` value of http://oauth.net/client/description (REQUIRED)
- `Link` element with the following `rel` value of http://oauth.net/client/redirect_uri (REQUIRED)
- `Link` element with the following `rel` value of http://oauth.net/client/uri (REQUIRED)
- `Link` element with the following `rel` value of http://oauth.net/client/icon (OPTIONAL)

For example:

```
<XRD>

<Host>http://onlinephotogallery.com</Host>

<Property type="http://oauth.net/client/name">Online Photo Gallery</Property>
<Property type="http://oauth.net/client/description">Not only uploading, but also editing
capabilities!</Property>

<Link rel="http://oauth.net/client/uri" href="http://onlinephotogallery.com">
<Title>Client URI</Title>
</Link>

<Link rel="http://oauth.net/client/redirect_uri" href="http://onlinephotogallery.com/cb">
<Title>Client Redirect URI</Title>
</Link>

<Link rel="http://oauth.net/client/icon" href="http://onlinephotogallery.com/icon.png">
<Title>Client Icon</Title>
</Link>

</XRD>
```

TBS: Change that to JRD

Client Registration Response

After receiving and verifying information retrieved from the client, the authorization server issues the client identifier and an optional client secret, and constructs the response by adding the following parameters to the entity body of the HTTP response with a 200 status code (OK):

- client_id (REQUIRED)
- client_secret (OPTIONAL)

The parameters are included in the entity body of the HTTP response using the "application/json" media type as defined by [RFC4627]. The parameters are serialized into a JSON structure by adding each parameter at the highest structure level. Parameter names and string values are included as JSON strings.

The authorization server MUST include the HTTP “Cache-Control” response header field with a value of “no-store” in any response containing the client_secret.

For example:

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

{
  "client_id":"5UO9XcL4TQTa",
  "client_secret":”WdRKN3zeTc20”
}
```

Unsuccessful Client Discovery

If the host-meta discovery was not successful, the authorization server constructs the response by adding the following parameters to the entity body of the HTTP response with a 404 status code (Not Found) using the “application/json” media type:

- error (REQUIRED) This parameter must be set to “hostmeta_error”
- description (OPTIONAL)

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  "error":"hostmeta_error",
  "description":”The hostmeta document could not have been retrieved from the URL.”
}
```

Error Message

If the request for registration is invalid or unauthorized, the authorization server constructs the response by adding the following parameters to the entity body of the HTTP response with a 400 status code (Bad Request) using the “application/json” media type:

- error (REQUIRED)
- description (OPTIONAL)

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

{
  "error":"unauthorized_client",
  "description":”This client is not on the white list of this Authorization Server”
}
```

TBS: We need standardized error codes.

References

[sitemeta] Defining Well-Known Uniform Resource Identifiers (URIs) (RFC5785)
http://tools.ietf.org/html/rfc5785

[hostmeta] Web Host Metadata
http://tools.ietf.org/html/draft-hammer-hostmeta-13

Maciej Machulak
m.p.machulak@ncl.ac.uk