API Authentication

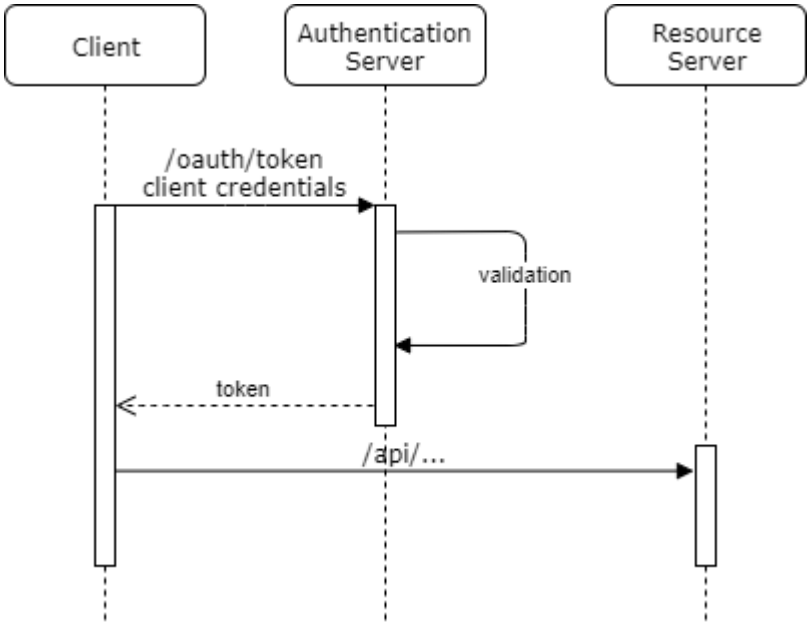
Restrictions

Currently the authentication server and resource server is one and the same server. They might be separated in the future. This description describes them as separate servers.

Client credentials which exist of an ID and a secret are created by Procura.

Authentication flow

The authentication flow uses the client credentials flow of OAuth 2.



The client, which MUST NOT be a browser, authenticates against the authentication server which validates the credentials and returns an access token.

The client credentials must be base 64 encoded in the 'Authorization' headers. First join the id and secret with a ':' and then encode it. E.g. encode("s6BhdRkqt3" + ":" + "gX1fBat3bV")

Example request with client id: s6BhdRkqt3 and secret: gX1fBat3bV:

```
POST /oauth/token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
scope=api
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
```

```
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"bearer",
  "expires_in":3600
}
```

The access token can then be used to make request against the resource server.

Example request:

```
GET /api/object HTTP/1.1
Host: server.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
```

When the token is expired the resource server returns a 401 Unauthorize HTTP status and the client must authenticate again.

References

https://tools.ietf.org/html/rfc6 749#section-4.4