Sécurité sociale

# OAuth2 Integration Client Credential

## *Dimona REST Web Service*

## DOCUMENT HISTORY

| Version | Date | Author | Description of changes/remarks |
|---|---|---|---|
| 00:01 | 4-12-2019 | Florent Marteau | Draft version |
| 0.1 | 4-12-2019 | Stéphane Flamme | Review |
| 0.2 | 30/06/2022 | Florent Marteau | New OAuth v4 URLs |
| 0.3 | 31/03/2023 | Aymerick Soyez | New OAuth v5 URLs |
| 0.4 | 04/03/2024 | Aymerick Soyez | Add “Dimona only” note |

# Inhoudsopgave

# 1. Presentatie van het protocol OAuth 2.0

OAuth 2.0 is een beveiligingsprotocol waarmee een persoon zijn toegangsrechten aan een resource kan delegeren. Met dit protocol kan een toepassing namens iemand handelen zonder de geheime informatie van die persoon te moeten verstrekken. Dit protocol wordt gebruikt om de REST-webservice te beveiligen.

De clienttoepassing haalt een Access Token op bij een authenticatieserver. Daarmee krijgt ze toegang tot een beschermde resource op naam van de eigenaar van de resource.

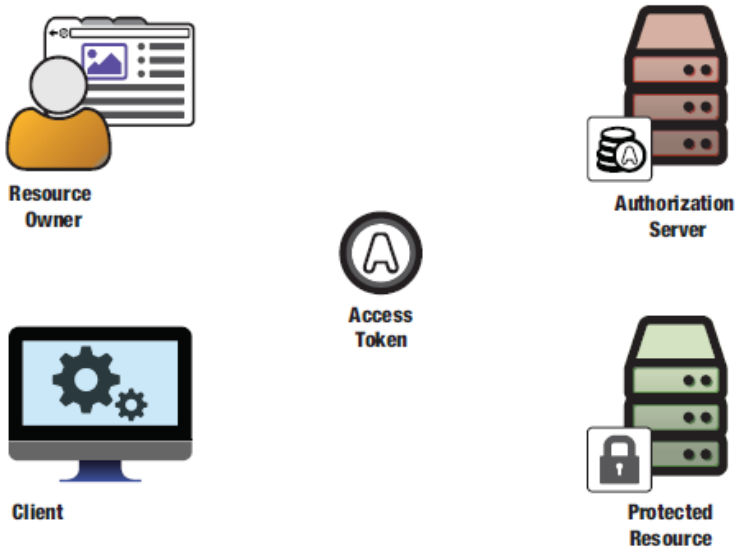
## 1.1. Lexicon

Hieronder staan de OAuth-specifieke termen.

- ***Resource***: iets waarvan we de toegang willen beschermen en waarvoor de Resource Owner kan kiezen om de toegangsrechten te delegeren.
  OAuth definieert de aard van de resource niet. Het is dus mogelijk om een systeem te bouwen voor het beheren van toegangsrechten tot applicaties. In dat geval is de resource het recht van toegang.
- ***Token***: een object dat is verkregen bij de Authorization Server dat toelaat om te bewijzen dat de eigenaar van de resource ons toegang heeft gegeven tot dre resource. OAuth definieert twee verschillende soorten Tokens:
  - **Access Token**, gebruikt om toegang te krijgen tot een resource.
  - **Refresh Token**, om een Access Token in sommige autorisatiestromen te vernieuwen zonder dat de aanwezigheid van de Resource Owner vereist is.
- ***Scope:*** Laat toe de toegangsrechten tot de resource die is gedelegeerd door de Resource Owner te definiëren. Met de Scope ‘read’, heeft een persoon bijvoorbeeld toegang tot de resource, maar hij kan ze niet wijzigen.

## 1.2. Actoren

- ***Resource Owner**:* De eigenaar van de resource die een toepassing in zijn naam wil laten handelen. Over het algemeen is dit een persoon.
- ***Protected Resource**:* De beschermde resource waartoe de eigenaar toegang heeft. Die kan verschillende vormen aannemen, maar over het algemeen is het een web-API die verschillende toegangsrechten kan hebben (lezen, schrijven...).
- ***Client**:* De 'client' is de toepassing die toegang heeft tot de beschermde resource namens de eigenaar van die resource. Deze toepassing kan bijvoorbeeld een applicatie op een server zijn, of een javascript- of native toepassing.
- ***Authorization Server:*** Met de autorisatieserver kan een A*ccess Token af*geleverd worden dat de clienttoepassing kan gebruiken om op te treden in plaats van de eigenaar van de resource. De autorisatieserver zal daartoe de eigenaar authenticeren en verifiëren of die akkoord is.



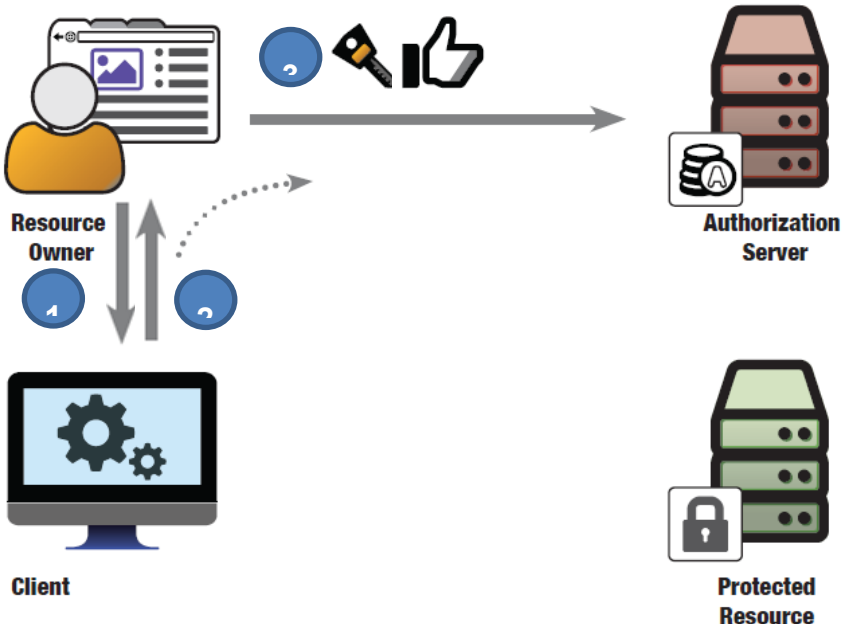

Figuur 1: de OAuth 2.0-actoren

## 1.3. Stappen in het autorisatieproces

In het kader van het gebruik van de client credential-stroom, handelt een clienttoepassing niet namens een derde, maar in haar eigen naam. In dit scenario is de client dus de Resource Owner.

### 1.3.1. Identificatie en authenticatie van de client

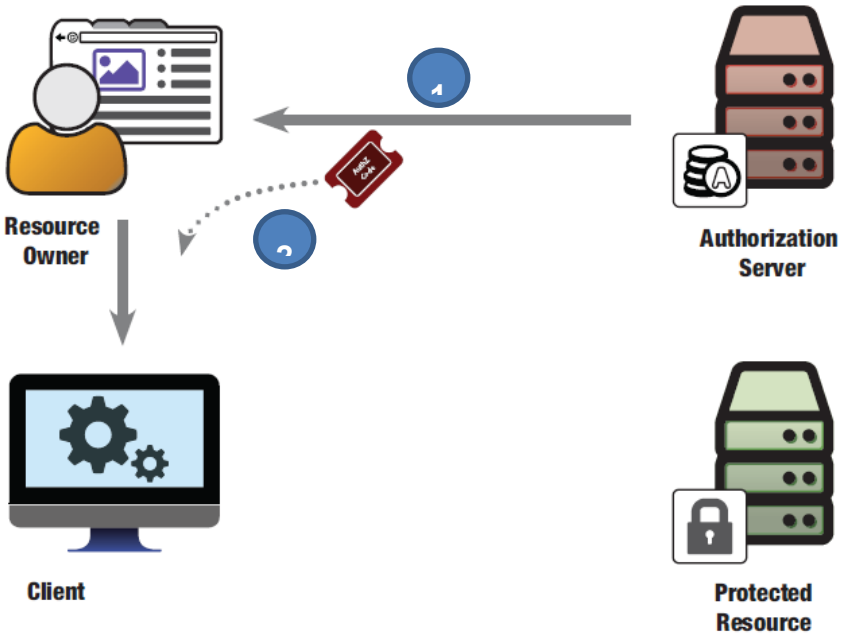
Aan het begin van het proces neemt de Resource Owner contact op met de Client (1) en laat hem weten dat hij in zijn naam mag handelen. Naar aanleiding van dit verzoek neemt de Client contact op met de Authorization Server (2) en geeft aan dat hij wil kunnen handelen in plaats van de Resource owner. Hiervoor brengt de client de Resource Owner in contact met de Authorization Server door enerzijds informatie (3) te verzenden die toelaat de tot de resource gevraagde rechten te identificeren (via 'scope') en anderzijds een URL waarmee de Authorization Server contact kan opnemen op met de client zodra het verzoek is aanvaard is.




Figuur 2: De Client stuurt de Resource Owner door naar de Authorization Server voor authenticatie en akkoord

### 1.3.2. Tokenverzoek bij de autorisatieserver

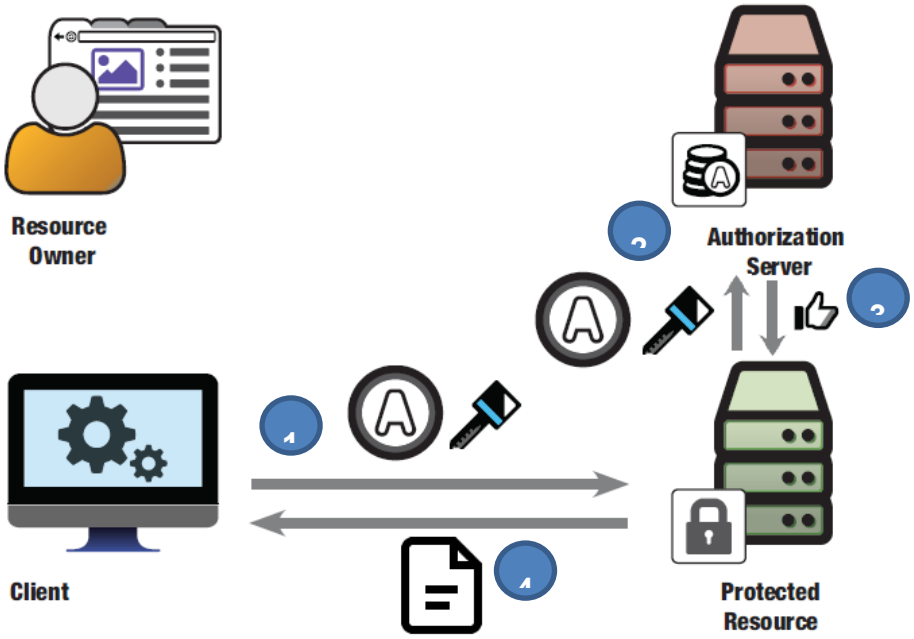
Nadat de Resource Owner is geïdentificeerd en geverifieerd bij de Authorization Server, vraagt die laatste de goedkeuring van de Resource Owner (1) met vermelding van de rechten (scope) die door de client op de resource zijn aangevraagd. Nadat de goedkeuring is verkregen, stuurt de Authorization Server de Resource Owner door naar de client (2) met behulp van de eerder verkregen redirection-URL.




Figuur 3: De Authorization Server verzendt een Authorization Code naar de Client

### 1.3.3. Toegang tot een beschermde resource

Zodra de client een Access Token heeft, en dit zolang het Access Token geldig blijft, kan het gebruiken om toegang te krijgen tot de resource. Hiertoe zal hij zijn identificatie, zijn secret en het Access Token doorgeven aan de Resource Server (1). Die laatste neemt contact op met de Authorization Server door die informatie (2) te verzenden zodat de Authorization Server de legitimiteit van het verzoek verifieert. Nadat het verzoek is goedgekeurd (3) door de Authorization Server, stuurt de Resource Server de gevraagde resource terug naar de Client.

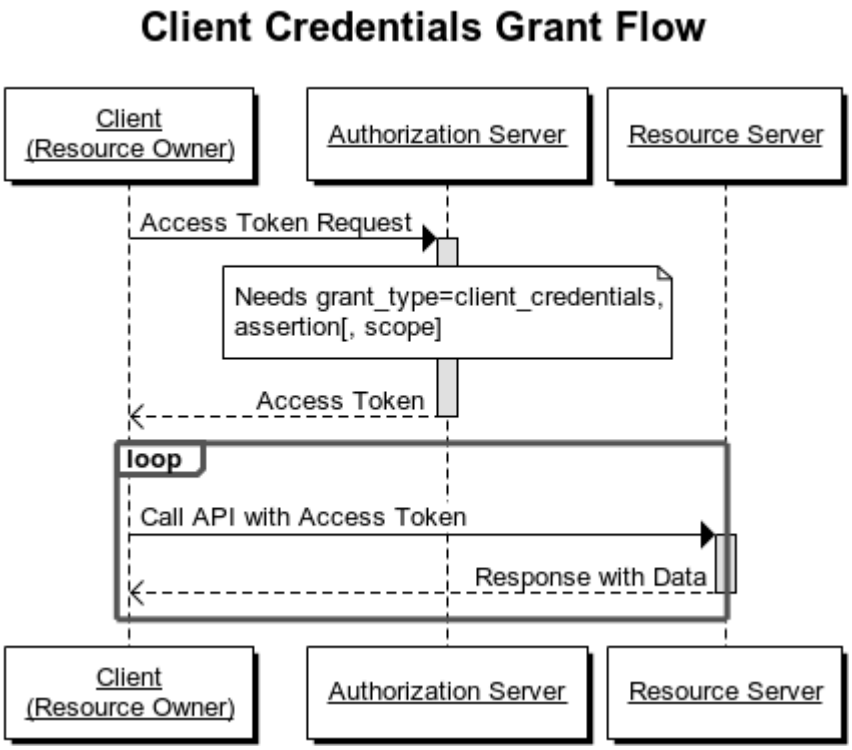



**Figuur 4: Een Access Token gebruiken om een door OAuth beveiligde resource te verkrijgen**

# 2. Client credential-integratie

In dit gedeelte wordt specifiek ingegaan op de client credential-stroom voor de Dimona REST Web Service. Deze stroom wordt gebruikt wanneer de client namens zichzelf handelt. Deze stroom maakt de authenticatie mogelijk van een rechtspersoon in plaats van een natuurlijk persoon. Hij wordt daarom gebruikt voor bedrijven of dienstverleners die in hun naam toegang willen hebben tot een REST-resource.

In dit gedeelte vind je de nodige informatie omtrent het oproepen van de autorisatieserver en het formaat van het mogelijke antwoord.



## 2.1. Client Credentials grant flow

De client kan een Access Token aanvragen door enkel zijn credentials (of een ander ondersteund authenticatiemiddel) te gebruiken wanneer hij toegang vraagt tot een beveiligde resource die hij controleert, of wanneer hij toegang vraagt tot beveiligde resources op basis van een autorisatie die hem voorafgaand werd verleend door de autorisatieserver.

De client credential-stroom kan enkel gebruikt worden door clients door het systeem vertrouwd worden.

### 2.1.1. Autorisatieverzoek en antwoord

Aangezien de clientauthenticatie wordt gebruikt als autorisatietoewijzing, is er geen extra autorisatieverzoek nodig.

### 2.1.2. Access token request

De client doet een verzoek bij het Token Endpoint door de volgende parameters toe te voegen, met behulp van het formaat 'application/x-www-form-urlencoded' met een UTF-8-codering in het veld van het HTTP-verzoek zelf. Deze parameters zijn rechtstreeks afgeleid van RFC 7519:

**grant_type**

VEREIST. De waarde moet 'client_credentials' zijn.

**scope**

OPTIONEEL. De scope van de zoekopdracht.

***client_assertion_type***

VEREIST. De waarde moet 'urn:ietf:params:oauth:client-assertion-type:jwt-bearer' zijn.

***client_assertion***

VEREIST. Een ondertekende JWT die de client authentiseert bij de autorisatieserver, zoals beschreven in RFC 7519. De inhoud van de JWT wordt beschreven in het gedeelte 'Client authentification'.

Voorbeeld van een HTTP-verzoek:

POST /REST/oauth/v5/token HTTP/1.1

```
Content-Type=application/x-www-form-urlencoded

charset=UTF-8

client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-
type%3Ajwt-
bearer&grant_type=client_credentials&scope=scope%3Awarlock%3Atest%3Arest%3Aappl
ication&client_assertion=eyJhbGciOiJSUzI1NiJ9.eyJhdWQiOiJodHRwczpcL1wvb2F1dGhpb
nQuc29jaWFsc2VjdXJpdHkuYmUiLCJzdWIiOiJ3YXJsb2NrOnRlc3Q6d2ViOjEiLCJpc3MiOiJ3YXJs
b2NrOnRlc3Q6d2ViOjEiLCJleHAiOjE1Mzk2NzQ2NjEsImlhdCI6MTUzOTY3NDYxMSwianRpIjoiNGI
0ZmIyMjIyNjE1ZGNhMTA4MGMzOWQ2YTVhMjFmYzAxZTI5Yzk5NGVjNjU5OTUwMDdlNDdhYzJkOTRhMT
liZjQzZjhmYzg0ZWZmYmJhOTA0NjM1YjZjMGZkYjE1NmI5YWFmYTQ5ZTFhYzE4MmFlYWY1MGIwMDY3M
zc4ZWJlZTIifQ.hRnJs5I4HVlC_YwQM6zVq0vj1JhuczTdweoLjfUnFmGusFSkDKQXUpqZMbi8LOwxy
eIyIszO3uy6TEOwZbyzp57Ms4YS2OmyQrF_3wvblM5rIkTxLBdsKoOxvIjyZKWmopy1V3CKUhtOL8UL
JeMquX4BHF1PBW4rRYJjGzhMKMZ5RFO-
wwOmzoa6VrMPpCsVpaKb0BPexGcNQJHyT2gjyLd2QZ2Ij5ctrxq7TEOn4bk0RvYyM8xxXQ70WAl7QsC
sZCYjKz55yGmdb9YKBU__i1uIOlELa0DtKrK3vP2SJxnVIBeC9IA_MWya9sBwqCyW8WS7dmr6UHpwGl
EWGr-u_Q
```

### 2.1.3. Access token response

Als het verzoek voor een Access Token geldig en geautoriseerd is, verstrekt de autorisatieserver een Access Token. Een Refresh Token wordt nooit verstrekt in de client credential-stroom. Als het verzoek mislukt of ongeldig is, stuurt de autorisatieserver een antwoord terug met een fout.

Het antwoord bevat de volgende parameters, zoals beschreven in RFC 7519:

***access_token***

Het Access Token, verstrekt door de autorisatieserver.

***token_type***

Type van het token, zoals beschreven in RFC 6749 section 7.1.

***expires_in***

De levensduur van het Access Token, in seconden. De waarde '3600' geeft bijvoorbeeld aan dat het Access Token een uur na het genereren van het antwoord zal vervallen.

Voorbeeld van een correct antwoord:

```
HTTP/1.1 200 OK
Date: Tue, 16 Oct 2018 09:37:56 GMT
Server: Apache
Content-Length: 298
Content-Type: application/json; charset=UTF-8
{
    "access_token":"7hbq6oh04a8h5asq1kon18fl4m",
    "scope":"scope:warlock:test:rest:application",
    "token_type":"Bearer",
    "expires_in":43199
}
```

### 2.1.4. Error response

De autorisatieserver reageert met een http 400-status (ongeldig verzoek) en bevat de volgende parameters, zoals beschreven in RCF 7519 :

**error**

VEREIST. Een unieke ASCII-foutcode [US-ASCII], mogelijke opties :

*invalid_request*

Er ontbreekt een verplichte parameter in het verzoek, er is een niet-ondersteunde waarde voor een parameter (andere dan het grant type) of een herhaalde parameter. Het verzoek gebruikt meer dan één methode om de client te authenticeren of is slecht geformatteerd.

*invalid_client*

De authenticatie van de client is mislukt (client niet bekend, authenticatie niet aanwezig of authenticatiemethode niet ondersteund).

*invalid_grant*

De gebruikte authenticatiestroom is onjuist.

*unauthorized_client*

De geauthenticeerde client mag dit type stroom niet gebruiken.

*unsupported_grant_type*

De stroom wordt niet ondersteund door de autorisatieserver.

*invalid_scope*

De gevraagde scope is ongeldig, onbekend, onjuist formaat, of wordt niet aangeboden door de eigenaar van de resource.

**error_description**

OPTIONEEL. Bericht met aanvullende informatie over de opgetreden fout.

**error_uri**

OPTIONEEL. Een URL van de webpagina met documentatie van de verkregen fout.

Voorbeeld van een fout antwoord:

HTTP/1.1 400 Bad Request

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

```
{
        "error":"invalid_request",
        "error_description":"Request was missing the client_id parameter.",
        "error_uri":"https://tools.ietf.org/html/rfc6749",
}
```

## 2.2. Client Authentification

Een ondertekende JWT die de client binnen de autorisatieserver authentiseert, zoals beschreven in RFC 7519. De inhoud van de JWT wordt beschreven in het gedeelte ‘Client authentification’ van deze RFC:

**jti**

VEREIST. Deze parameter biedt een unieke identificatiecode voor de JWT. Deze parameter is hoofdlettergevoelig.

**iss**

VEREIST. Deze parameter identificeert de toepassing die de JWT levert. Deze parameter is hoofdlettergevoelig.

**sub**

VEREIST. Deze parameter identificeert de 'principal' die het onderwerp is van de JWT. Deze parameter is hoofdlettergevoelig en bevat een string- of URI-waarde.

**aud**

VEREIST. Deze parameter definieert de ontvanger van de JWT.

**exp**

VEREIST. Deze parameter definieert de vervaldatum waarop of waarna de JWT niet langer mag worden geaccepteerd.

**nbf**

VEREIST. Deze parameter identificeert de timestamp vóór dewelke de JWT niet mag worden geaccepteerd.

**iat**

VEREIST. Deze parameter definieert de timestamp waarop de JWT is aangemaakt.

# 3. URLs

URL van de server: **https://services.socialsecurity.be/**

Toegangspunt:

- Back channel: **https://services.socialsecurity.be/REST/oauth/v5/token**

Audience: **https://services.socialsecurity.be/REST/oauth/v5/token**