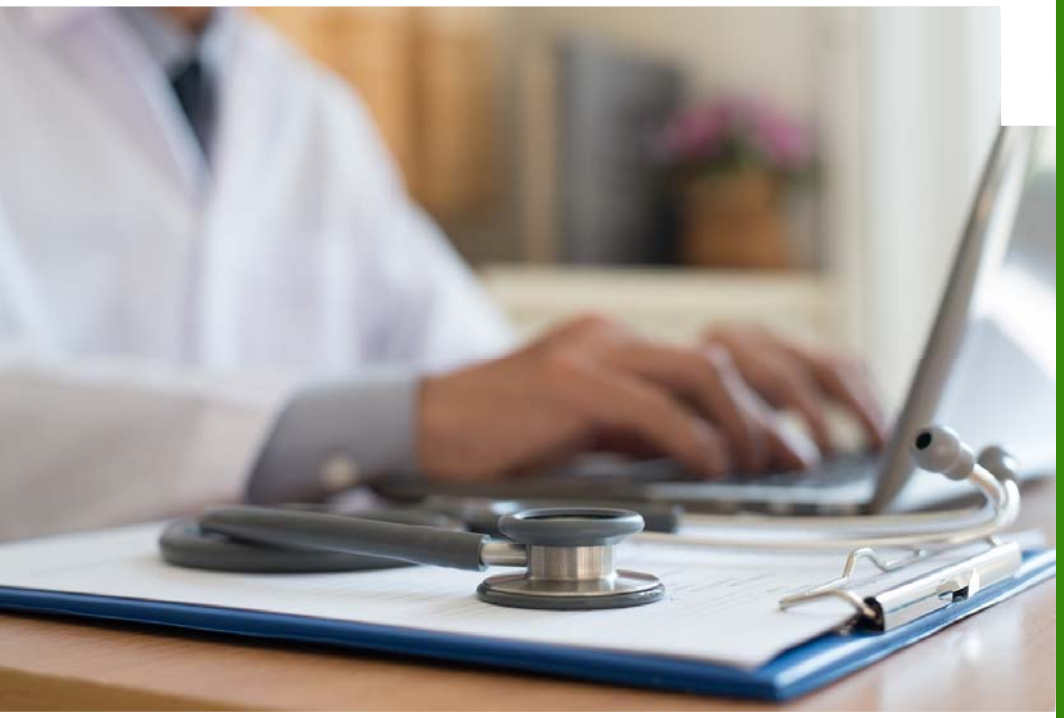
Logius
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties



eID Routeringsvoorziening OpenID Connect

Coen Glasbergen

13 februari 2019

Routeringsvoorziening@logius.nl

Inhoud

Wet Digitale Overheid

eID en Routeringsvoorziening

OpenID Connect

Feedback

Wet Digitale Overheid

- Standaardiseren en verbreden authenticatiemogelijkheden
- Verhogen betrouwbaarheidsniveau authenticatie

- Machtigen

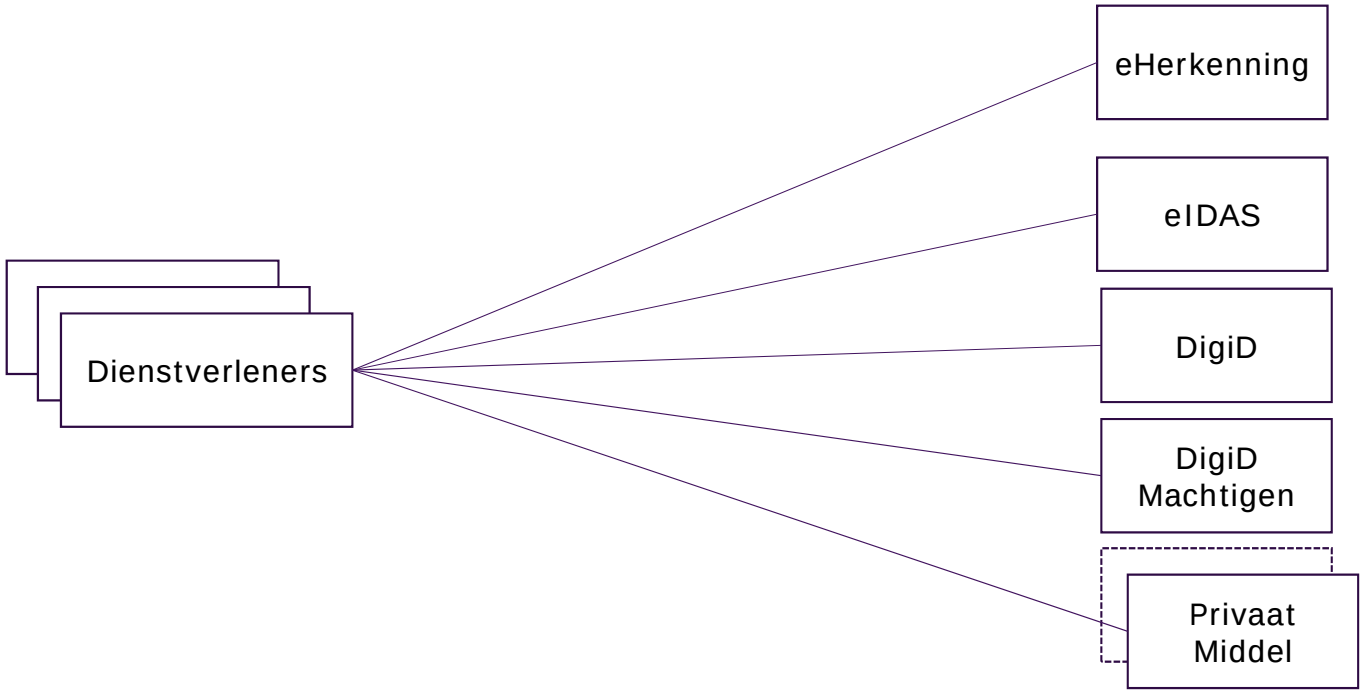
- Privaat alternatief voor DigiD

- eHerkenning

- eIDAS

→ Aansluitverplichting

Wet Digitale Overheid - aansluitverplichting

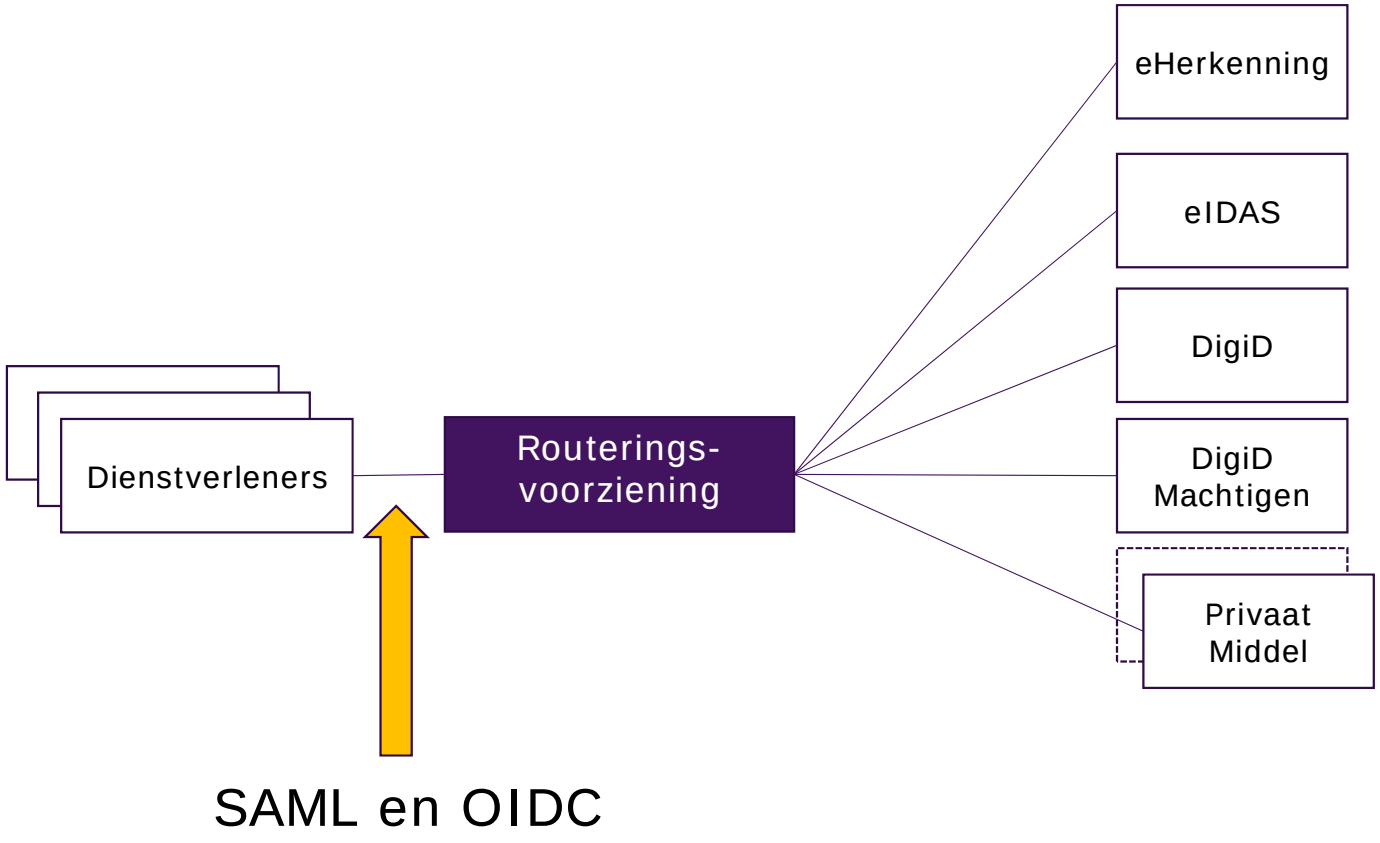


→ Ontzorging

Ontzorging - Routeringsvoorziening

- Één aanspreekpunt
- Één contract
- Één factuur
- Één *koppelvlak*

Wet Digitale Overheid



Waarom SAML

- Bekend, dus snellere adoptie

Waarom op termijn geen (DigiD) SAML

- Functioneel:
  - Geen attributen
  - Geen machtigingen
- Technisch:
  - Geen doorontwikkeling SAML
  - Mobiel

→ OpenID Connect

OpenID Connect

- Gebaseerd op OAuth2
- Standaard – profiel - koppelvlak

- Besproken met Forum Standaardisatie
- Profiel notificeren voor ‘Pas toe of leg uit’-lijst
- In lijn brengen met bestaande profiel OpenID iGov en OAuth2 NL/iGov

→ Detailleren

Detaillering in specificatie

- Definiëren van scope
- Definiëren van flows

- Verzorgen van interoperabiliteit
- Reduceren van ambiguïteit

- Definiëren van security requirements
- Definiëren van operational requirements

We vragen feedback

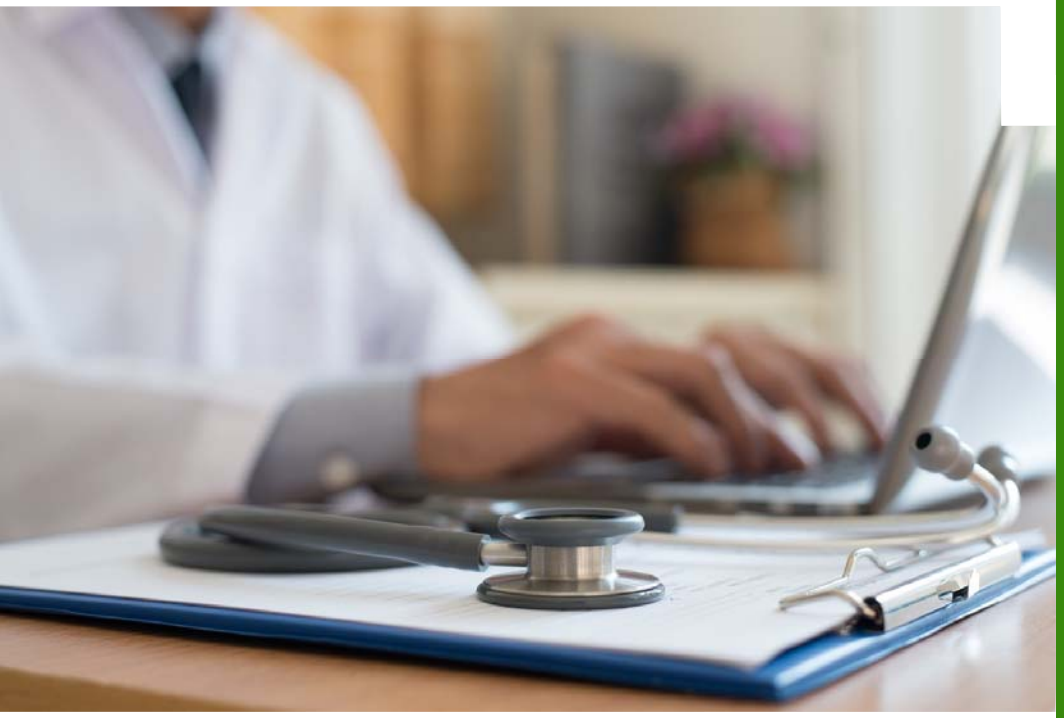
- Verzoek / kans om
  - Feedback te geven over de concept specificaties
  - Stellen van vragen over flows en use cases. Bijvoorbeeld, vragen over hoe een use case te implementeren
  - Laat het ons weten als de concept specificaties:
    - Onduidelijk zijn
    - Multi-interpreteerbaar zijn
    - Moeilijk implementeerbaar zijn

→ Meld je aan op Routeringsvoorziening@Logius.nl

Beoogde planning

- Eind februari : eerste ronde feedback
- Medio maart :
  - Update met daarin feedback verwerkt
  - Verzoek voor feedback op deze herziene versie
- Eind maart : tweede ronde feedback
- April : release 1.0
- April : Notificatie bij Forum Standaardisatie

Logius
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties



eID Routeringsvoorziening OpenID Connect

Coen Glasbergen

13 februari 2019

Routeringsvoorziening@logius.nl

OAuth2 & OpenID Connect

- **OAuth2**
  - Authorization framework for the web.
- **OpenID Connect (OIDC)**
  - Authentication, identity layer built on top of OAuth2
  - OpenID Connect is a superset of OAuth2

OAuth2 – Key Concepts

- **Scope**
  - The scope of the access request or token.
- **Access Token**
  - Credentials used to access protected resources.
- **Endpoints**
  - Authorization endpoint
  - Token endpoint
- **Plain Http & JSON**
  - No xml

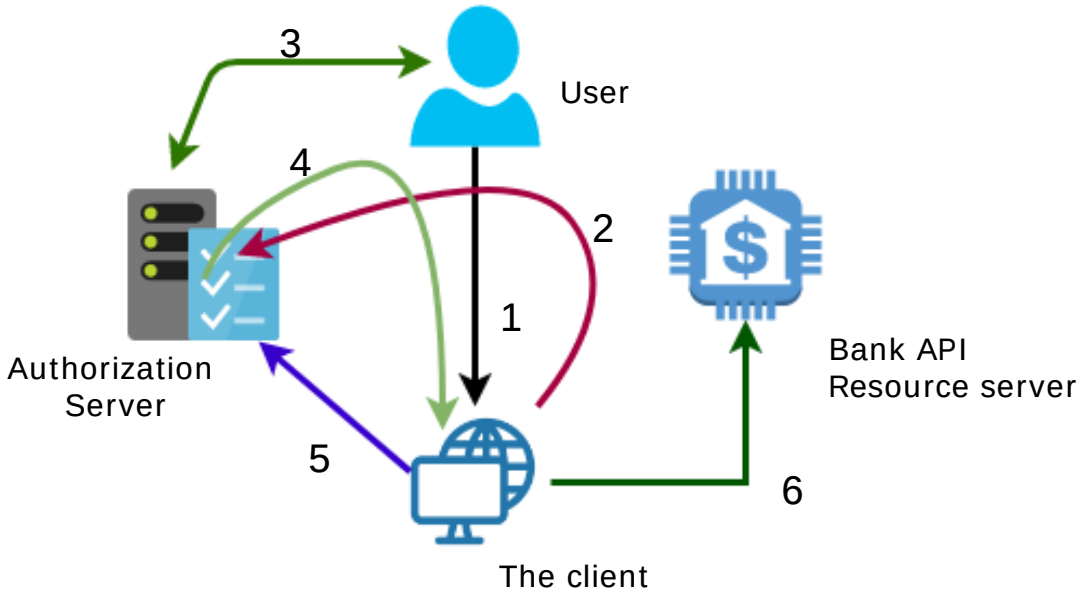
OAuth 2.0 : The Foundation For OIDC

Example Use Case:

- A user : Resource owner
- An application : The client
- Bank : Resource server
- Authorization server
- Scope : Reading transactions

The user wants to authorize the application to access his/her bank account transactions without storing his/her password, using an access token instead.

Bank trusts the authorization server and will accept an access token from it.

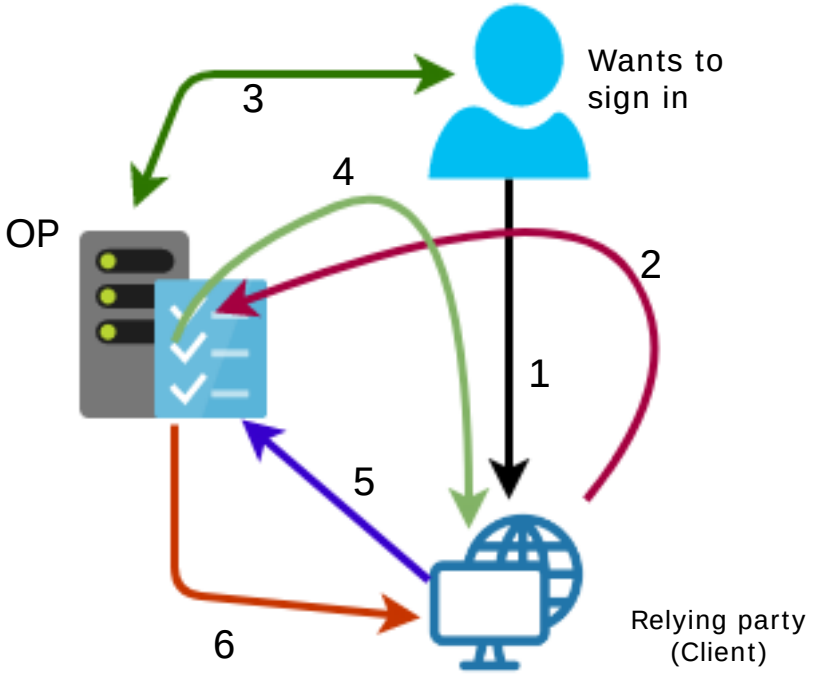


OpenID Connect

- **OAuth2 + new flows and definitions**
  - openid scope
  - ID token
  - Claims (user attributes)
  - Authentication, discovery, registration and more flows
- **Dependencies**
  - JSON
  - JWT : Json Web Tokens
  - JWE : Json encryption
  - JWS : Json signatures
  - JWK : Json web keys

OIDC Authentication Flow (using authorization code flow)

- **Steps**
  1. User wants to sign in to the web site
  2. User is redirected to OpenID provider (OP)
  3. Authenticates
  4. User redirected back to the relying party with code
  5. The web site exchanges the code in a back channel call
  6. Response contains id_token which contains user information



ID Token

- **Contains user information (claims)**
- **JWT format**
- **Similar to the Assertion in SAML**

SAML, OAuth2, OIDC Mappings

SAML	OAuth2	OIDC
Identity Provider	Authorization Server	OpenID Provider
Service Provider	Client	Relying Party
Assertion	Access Token	ID Token
AuthnRequest	Authorization Request	Authentication Request
Redirecting the user with an artifact after authentication	Authorization response with a code when using code flow	Authentication response with a code when using code flow
Artifact Resolution Request For Obtaining The Assertion	Token Request	Token Request
Artifact resolution response containing a SAML Assertion	Token response	Token response
Attribute Query		Userinfo request
XML	JSON	JSON

Specification Topics

- Flow Definitions
- ID token format and contents
- Access token format and contents
- Client registration : How will you get a client?
- Client authentication
- Scopes : Scope management and registering scopes
- Userinfo : Getting user details
- Introspection : Validating access tokens

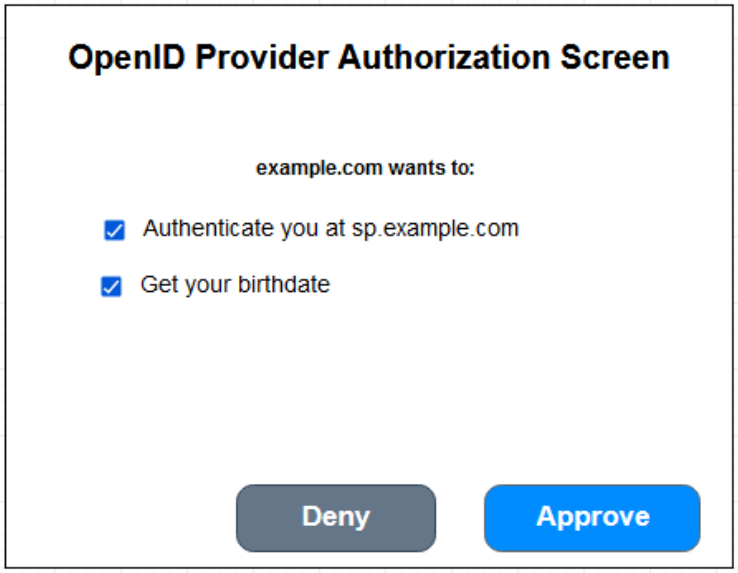
Example flow: Step 1 – Authentication Request

- An unauthenticated user wants to access your web site. You redirect the user (user's browser) to the OpenID Provider with a HTTP redirect:

```
HTTP/1.1 302 Found
Location: https://openidprovider.example.com/authorization-endpoint?
    response_type=code
    &scope=openid%20birthdate%20urn%3Anl-gdi-services%3Aea0eb2b6-9cf5-
4a6e-99de-e41da799eb8b
    &client_id=client_id_assigned_by_the_OP
    &state=random_state_value_generated_by_the_Client
    &nonce=nonce_value_generated_by_the_Client
    &redirect_uri=https%3A%2F%2Fclient.example.com%2Fcallback-url
```

Example flow: Step 2 – Authorization/Consent

- Authentication: The OpenID Provider authenticates the user and displays an authorization/consent page:



Example flow: Step 3 – Authentication Response

- The OpenID Provider redirects the user (user's browser) to the redirect uri from step-1, with a code value

```
HTTP/1.1 302 Found
  Location: https://client.example.com/callback-url?
    code=random_code_value
    &state=the_state_value_from_step1
```

Example flow: Step 4 – Token Request

- The client gets the code value from the previous step and sends a backend request to the OpenID Provider token endpoint:

```
POST /token-endpoint HTTP/1.1
Host: openidprovider.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=code_value_from_step3&
redirect_uri=https%3A%2F%2Fserviceprovider.example.com%2Fcallback-url&
client_assertion=ey...Mz6w&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
```

Example flow: Step 5 – Token Response

- The token endpoint responds with a JSON which contains an access_token and an id_token in JWT format.

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

{
"access_token":"eyJh...<removed for formatting purposes>...opz82RIPyw",
"id_token":"eyJhbGciOiJSUzI1N... <removed for formatting purposes>...2JKeCcqaJzA",
"token_type":"Bearer",
"expires_in":299
}
```

Example flow: Step 6 – ID Token

- id_token from step 5 contains the user information. Validating and decoding it gives us the following JSON, where “sub” contains the user identifier :

```
{
 "alg": "RS256",
 "kid": “id_of_the_key_used_for_signing_this_id_token"
}
{
 "scope": "openid birthdate urn:nl-gdi-services:ea0eb2b6-9cf5-4a6e-99de-e41da799eb8b",
 "client_id": “your_client_id",
 "jti": “unique_and_random_id_token_id",
 "iss": "https://openidprovider.example.com:9031",
 "aud": "https://resourceserver.example.com",
 "sub": “unique_user_identifier",
 "azp": "pkjwt_1",
 "exp": 1544627159
}
```