Edukoppeling

MDX Secure API OAuth profile
voor
M2M gegevensuitwisseling binnen het onderwijs

Edustandaard

Datum: februari 2023

Versie: 0.2

Status: concept

edustandaard

Inhoudsopgave

Inhoud

1. Historie

Versie	Auteur	Datum	Opmerking
0.1	E. Reinhoud (BES)	januari 2023	Initiële versie gebaseerd op uitgangspunten in het discussiestuk (versie 0.3).
0.2	E. Reinhoud (BES)	februari 2023	1. Commentaar versie 0.1 verwerkt 2. In het discussiedocument v0.3 hadden we met RFC8705 voor ogen dat mTLS een rol kon spelen bij client authenticatie en het binden van het token aan het client certificaat. In versie 0.1 van dit OAuth profiel hebben we de tokenbinding los gelaten omdat bleek dat RFC8705 niet het subject.serial ondersteunt (PKIo gebruikt dit voor het OIN). In deze 0.2 versie van het OAuth-profiel laten we ook client authenticatie op basis van het mTLS client certificaat los. mTLS speelt alleen op transportniveau, na TLS offloading moet de client met eigen identifier identificeerbaar zijn en geauthentiseerd kunnen worden op basis van een JWT. We gebruiken dan ook geen Basic Authentication en met de JWT-optie sluiten we ook aan op het iGOV NL profiel. 3. Verder is het OAuth-profiel nu een aanvulling op het REST-profiel. Het is dus in essentie geen MDX-profiel maar een laag boven op het REST-profiel. Met deze ontkoppeling kunnen dit profiel zonder de MDX/mTLS PKio-laag gebruiken door de verwijzingen naar het REST-profiel te verwijderen. 4. Een domain_hint werd in vorige versie gedefinieerd als een referentie naar een autorisatie voor een bepaalde organisatorische eenheid (groep betrokken en relevante attributen). Het autorisatiemechanisme op dataniveau is echter te complex en valt ook buiten deze Oauth CC standaard. Het veld domain_hint is een verwijzing naar een organisatorische eenheid en is vergelijkbaar met het routeringskenmerk. In deze versie wordt er een naam gebruikt die beter aansluit op waar het naar verwijst, een onderwijsorganisatie (edu_org_id) 5. Doordat we niet de autorisatie van een dataset standaardiseren is de beschrijving van het scenario niet relevant. Het scenario in deze versie sluit volledig aan bij dat van OAuth client credentials profiel 6. Uitgangspunten toegevoegd en deels opnieuw geformuleerd 7. Herstructurering document

2. Inleiding

2.1. Aanleiding

De aanleiding voor de introductie van Edukoppeling in het onderwijsdomein is een steeds groter wordende stroom van geautomatiseerde (machine-machine) processen in het onderwijs. Dit wordt veroorzaakt door vernieuwingen in het onderwijs zelf, in wetgeving, in de beschikbare techniek en de wens om het aantal (technische) koppelvlakafspraken binnen de perken te houden. In toenemende mate lopen de processen over organisaties heen, tussen onderwijsorganisaties (zowel op bestuursniveau van de onderwijsaanbieders, de “scholen”) onderling, tussen onderwijsorganisaties en overheidsorganisaties en tussen onderwijsorganisaties en private onderwijsgerelateerde organisaties. En vaak, als er iets nieuws komt, wordt er dan pas nagedacht over de benodigde infrastructuur. Als men niet oppast worden er evenveel infrastructurele oplossingen gerealiseerd als er geautomatiseerde processen zijn. Met Edukoppeling verandert dat. Edukoppeling is een meervoudig inzetbare infrastructuur waarvan de ontwikkeling en het beheer gemeenschappelijk wordt aangepakt.

In het onderwijs is het normaal geworden dat onderwijsinstellingen veel van hun processen laten ondersteunen door zogeheten SaaS-diensten (diensten ‘in the cloud’). Dit geldt voor onderwijskundige processen als ook voor hun administratieve processen. Het Edukoppeling Mandated Data eXchange[1] (MDX) protocol en verwante profielen houden met deze ontwikkeling rekening. De diensten van leveranciers waar een onderwijsorganisatie gebruik van maakt beheren gegevens (administraties) en wisselen vaak namens de onderwijsorganisatie gegevens uit met ketenpartijen. De Edukoppeling Mandated Data eXchange houden expliciet rekening met het uitwisselen van gegevens tussen leveranciers (verwerkers) namens een onderwijsorganisatie (eindorganisatie).

Dit document beschrijft de Edukoppeling MDX Secure API OAuth profile (verder aangeduid als OAuth-profiel) en is onderdeel van de Edukoppeling Architectuur. Het beschrijft de aanvullingen die gelden bovenop het Edukoppeling MDX Secure API REST profile[2]. Het kan dus als een extensie hiervan gezien worden en de eisen voor het REST-profiel zijn dus ook van toepassing op dit OAuth-profiel, zoals de toepassing van het Mandated Data eXchange (MDX) protocol.

Bij dit OAuth-profile wordt de uitwisseling extra beveiligd doordat deze alleen kan plaatsvinden met een OAuth toegangstoken. Het gebruik van een toegangstoken wordt gerealiseerd door toepassing van OAuth 2.0 client credentials profile. Randvoorwaarden voor de levering van een toegangstoken aan een client zijn onder andere het bestaan van een geldig mandaat in het (centrale) OSR register voor de client en de (lokale) autorisatie voor een bepaalde data set die de API ontsluit.

2.2. Doel en doelgroep

Het doel dat met dit profiel nagestreefd wordt is het op een generieke manier kunnen uitwisselen van gegevens binnen de onderwijssector. Het profiel ondersteunt zowel het scenario waarbij een Eindorganisatie zijn systeem zelf beheert in de eigen ICT-infrastructuur, als het scenario waarbij de Eindorganisatie deze als (SaaS-)dienst van een verwerker (leverancier) afneemt.

Dit document is bedoeld voor ICT-specialisten die betrokken zijn bij het ontwerpen en ontwikkelen van systeem-naar-systeem (M2M) koppelingen. Het gaat hier om werknemers (ontwikkelaars, architecten, projectmanagers, informatiemanagers etc.) werkzaam bij onderwijsgerelateerde organisaties, zowel in

Met opmerkingen [ER1]: EB: Deze wens veroorzaakt toch niet de groeiende stroom M2M koppelingen?

Met opmerkingen [ER2]: EB: Suggestie ipv infrastructuur: Wijze van koppelen

Met opmerkingen [BD3R2]: Zie ook de andere opmerking over infrastructuur irt Digikoppeling.

Met opmerkingen [ER4]: EB: Suggestie: evenveel verschillende soorten van koppelingen bedacht als er geautomatiseerde processen zijn. Dat is nadelig, omdat hiervoor veel kennis nodig is, dit onnodig veel en kostbaar onderhoudt vergt, dit de interoperabiliteit en aanpasbaarheid hindert (nog meer?).

Met opmerkingen [ER5]: EK: Logius zegt: Het (digikoppeling) is in beginsel geen infrastructuur maar een set aan afspraken over het gebruik van internationale open standaarden. Digikoppeling kent wel ondersteunende voorzieningen maar deze zijn gericht op ondersteuning van het ontwikkelproces bij implementatie van Digikoppeling en niet op directe ondersteuning van productie-situaties zelf.

Dit is bij ons precies zo. We gebruiken de ondersteunende voorzieningen van Digikoppeling en voegen daar nog extra voorzieningen aan toe (OSR, wellicht SEM achtige layers in de toekomst?)

Met opmerkingen [ER6R5]: EK: woord infrastructuur zou ik dan ook graag vermijden

Met opmerkingen [ER7]: EB: Edukoppeling is tevens een open standaard, wat maakt dat partijen met een lage drempel kunnen deelnemen, wat gunstig is voor het onderwijs.

Met opmerkingen [ER8]: EB: Zoals ik het begrijp (tekstsuggestie): Bij het MDX … wordt uitgewisseld conform het API REST profiel, maar wordt de uitwisseling extra beveiligd met een Oauth autorisatietoken.

[1] Voorheen werden de betreffende profielen ook wel SaaS-profielen genoemd. Omdat hier vaak discussie bij ontstond omdat het niet altijd een SaaS-dienst /SaaS-leverancier betrof is de nieuwe naam Mandated Data eXchange geïntroduceerd vanaf deze versie. Dit geldt ook voor alle andere stukken die na maart 2023 zijn gepubliceerd

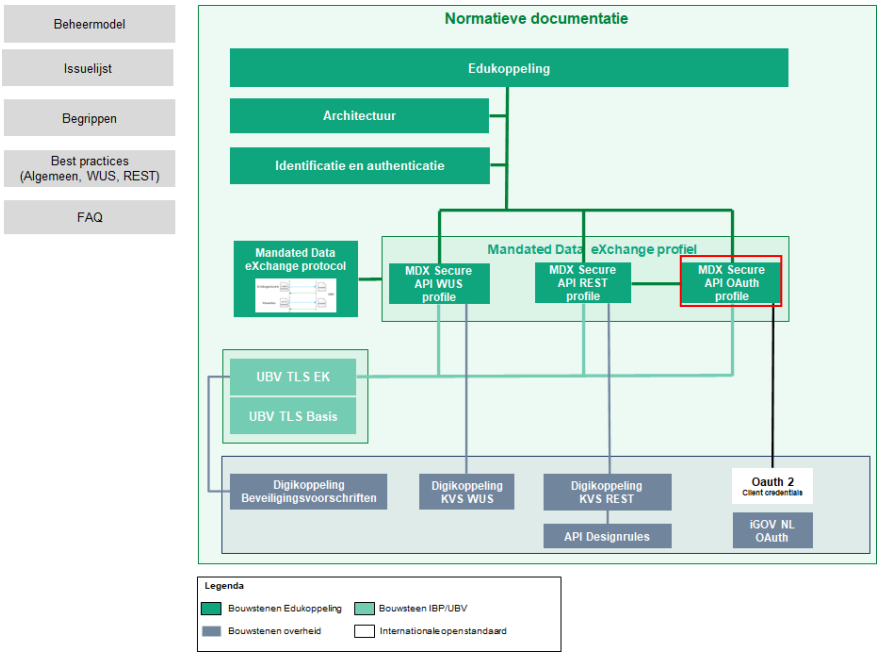
[2] Er loopt parallel aan de ontwikkeling van dit profiel nog wel een discussie om ook de scope van Edukoppeling te verbreden waarbij mogelijk ook andere OAuth-profielen relevant worden. Hierbij wordt wellicht niet een mTLS/PKIo in de basis gebruikt voor client authenticatie.

de publieke als private sector. Edukoppeling is voor een groot deel compliant aan de overheidsstandaard Digikoppeling. De Edukoppeling-documentatie dient derhalve naast de Digikoppeling-documentatie gebruikt te worden.

De lezer van dit document willen wij vragen om zaken die ontbreken of onduidelijk zijn te melden bij de beheerorganisatie Edustandaard[3].

2.3. Positionering binnen Edukoppeling Architectuur

Het Edukoppeling OAuth-profiel is onderdeel van de Edukoppeling Architectuur. Dit OAuth-profiel is een aanvulling op het Edukoppeling MDX Secure API REST-profiel. Het beschrijft welke aanvullende eisen er gelden rond het gebruik van de OAuth 2.0 client credentials profile. We maken hierbij gebruik van de internationale open standaarden zoals RFC6749 en RFC6750[4].



Figuur 1 - Positionering van OAuth-profiel binnen de Edukoppeling Architectuur

2.4. Functioneel toepassingsgebied

Het functionele toepassingsgebied van het OAuth-profiel betreft M2M-gegevensuitwisseling via een point-to-point verbinding voor uitwisseling van vertrouwelijke gegevens via een gesloten API[5]. Er worden bevragingen (pull) en meldingen (push) op basis van een request-response uitwisselingspatroon ondersteund. De client is in deze context geen browser, maar een systeem (applicatie). De systemen worden beheerd door verwerkers en voeren de uitwisseling uit op basis van

[3] https://www.edustandaard.nl/standaarden/afspraken/afspraak/edukoppeling/. Reageren kan via info@edustandaard.nl.

[4] Momenteel wordt ook binnen het iGOV NL aan een client credentials profiel gewerkt. We volgen de ontwikkelingen en zullen daar (op termijn) mogelijk gebruik van maken.

[5] Het Kennisplatform heeft o.a. een API Strategie ontwikkeld waarin verschillende soorten API's worden onderkend (https://geonovum.github.io/KP-APIs/API-strategie-algemeen/). Open API's: voor ontsluiten van diensten zonder toegangsbeperking bijv. open data. Gesloten API's: voor ontsluiten van diensten met toegangsbeperking bijv. persoonsgegevens en vertrouwelijke gegevens of diensten voor specifieke partijen (access-restricted and purpose-limited API's).

een mandaat van een eindorganisatie. Identificatie en authenticatie wordt in de point-to-point verbinding gerealiseerd met behulp van mTLS.

De gegevens kunnen op basis van de afspraken binnen dit profiel gerouteerd worden van verwerker naar eindorganisatie. Het profiel kan ook worden toegepast indien de eindorganisatie ook zelf de rol van verwerker heeft.

2.5. Notatiewijze voorschriften

Voor elk voorschrift wordt aangegeven in welke mate hier invulling aan moet worden gegeven. Hiermee kunnen we duidelijk aangeven wat de grenzen van dit profiel zijn ten opzichte van de mogelijke externe bron(nen) waar het voorschrift eventueel van wordt overgenomen. We gebruiken hiervoor de notatiewijze van RFC2119[6]. Deze gebruikt de volgende termen: "MUST" (“MOET”), "MUST NOT"(“MOET NIET”), "REQUIRED" (“VEREIST”), "SHALL" (“ZAL”), "SHALL NOT" (“ZAL NIET”), "SHOULD" (“ZOU”), "SHOULD NOT" (“ZOU NIET”), "RECOMMENDED" (“AANBEVOLEN”), "NOT RECOMMENDED" (“NIET AANBEVOLEN”), "MAY" (“MAG”), and "OPTIONAL" (“OPTIONEEL”).

[6] https://tools.ietf.org/html/rfc2119

3. OAuth client credentials profile

Dit OAuth-profiel is gebaseerd op het OAuth 2.0 raamwerk. OAuth is gebaseerd op RFC6749 die gepaard gaat met een Bearer Token Usage specificatie in RFC6750. Het geeft invulling aan het privacy-by-design principe doordat het voorkomt dat systemen van verschillende partijen persoonsgegevens kunnen uitwisselen zonder dat de betrokkene hier inzage of controle over heeft.

Het is een stap in de ontwikkeling van het beveiligen van API's. Bij eerdere mechanismen zoals HTTP Basic Authentication (gebruikersnaam & wachtwoord) en statische API-keys konden credentials verloren raken of door onbevoegde gebruikt worden. Ook was een fijnmazige (dynamisch) autorisatie niet goed in te richten. Met OAuth wordt er een stap gemaakt naar een tokens gebaseerde architectuur. Het biedt de mogelijkheid om (centraal) overzicht en controle te krijgen rond toegangsrechten. Het biedt ook eenvoud bij de ontwikkeling van applicaties doordat zij toegang kunnen krijgen tot gegevens op basis van een token.

Het OAuth raamwerk definieert 4 rollen:

1. Een Resource Owner (RO) beheert de gegevens (protected resource(s)) die door de API worden ontsloten. De RO wordt beschouwd als de "eigenaar" van de gegevens.
2. De Authorization Server (AS) geeft Access Tokens (AT) uit en kan deze ook weer intrekken.
3. De client is bijvoorbeeld een applicatie of website die de gegevens opvraagt namens de RO.
4. De Resource Server (RS) is de service die de gegevens ontsluit en opslaat. Dit is feitelijk de API.

OAuth onderkent dus zowel de client als de resource owner. De client en resource owner hebben elk hun eigen credentials en kunnen op basis hiervan geïdentificeerd en geauthentiseerd worden. Hierdoor kan in het interactiemodel de resource owner in staat wordt gesteld om via de authorization server aan te geven tot welke protected resource(s) de client toegang krijgt. Het access token dat de client van de authorization server krijgt vertegenwoordigt de autorisatie die de resource owner de client heeft gegeven.

OAuth onderkent verschillende grant types[7] voor verschillende use cases. De meest toegepaste grant types zijn het code grant en client credentials grant type. Dit OAuth-profiel is gebaseerd op de client credentials grant type. Hierbij speelt de resource owner geen rol. Er geldt over het algemeen dat de client credentials (identificatie en authenticatie) in principe voldoende zijn voor toegang tot de API. Bij de client credentials grant wordt uitgegaan van confidential clients (afgeschermde vertrouwde systemen) die de te gebruiken credentials goed kunnen beveiligen. Er kan namelijk worden gesteld dat de client op basis van de credentials impliciet geautoriseerd is en als de resource owner kan worden gezien.

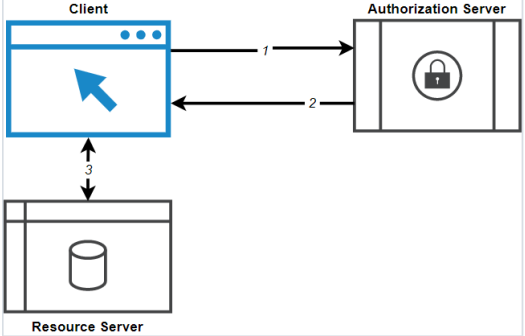
Belangrijke onderdelen binnen het client credentials grant type zijn de client credentials en het access token. Er zijn verschillende manieren om hieraan invulling te geven. Voor authenticatie van de client kan bijvoorbeeld gebruik worden gemaakt van Basic Authentication, mTLS of een JWT. Voor het access token kan bijvoorbeeld worden gekozen voor een string (i.c.m. introspection) of een zelfbeschrijvende JWT. De keuzes hierin vormen het belangrijkste onderdeel van dit OAuth-profiel.

Hieronder wordt het client credentials grant type schematisch weergegeven (zie Figuur 2).

[7] Het begrip “grant type” verwijst naar de manier (flow) waarop een client een access token krijgt. Elk grant type is geoptimaliseerd voor een bepaalde use case waarbinnen een bepaalde vorm(en) van een client wordt onderkend.

Stap 1. Request naar token endpoint
De client stuurt een verzoek naar het token endpoint van de authorization server. Deze valideert het verzoek op basis van de client credentials.

- De validatie kan gebaseerd zijn op (aanvullende) out-of-band overeengekomen criteria.
- Het verzoek kan referenties naar specifieke resources bevatten als er meerdere protected resources zijn en het access token een fijnmazige autorisatie moet bevatten.



Figuur 2 - OAuth client credentials profile (bron Kennisplatform API's | Geonovum)

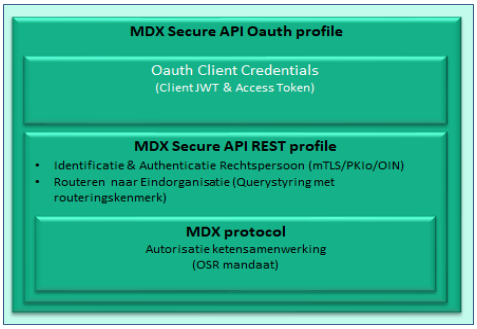
Stap 2. Respons van token endpoint
Na succesvolle validatie levert de authorization server een toegangstoken die overeenkomt met de gevraagde rechten, of een subset hiervan. De client ontvangt het access token en moet deze meesturen bij het zenden van een verzoek naar de protected resource.

Stap 3. Resource-interactie
Met het access token verzoekt de client namens zichzelf toegang tot de protected resource. De resource server waarop de protected resource wordt gehost valideert het access token voordat de client toegang krijgt. Er moet onder andere kunnen worden gevalideerd dat het Access token door de Authorization Server is uitgegeven en nog niet verlopen is. De levensduur moet zo kort mogelijk zijn, maar lang genoeg om tot werkbare interacties te komen. Als het access token valide is wordt toegang gegeven tot de protected resource. Indien dit niet het geval is wordt er een foutmelding gegeven conform RFC6749.

4. MDX Secure API OAuth client credentials profile

4.1. Uitgangspunten

1) Dit OAuth-profiel is een aanvulling op het MDX Secure API REST profiel. Het MDX Secure API REST profiel is op zijn beurt weer afgeleid van het Digikoppeling REST API profiel[8]. Voor dit OAuth-profiel dient men dus ook kennis te nemen van het Edukoppeling MDX Secure API REST profiel, het Digikoppeling REST API profiel en het MDX protocol dat kaders rond mandaten voor deelname aan een ketensamenwerking bevat. Deze afhankelijkheden worden weergegeven in Figuur 1 en **Fout! Verwijzingsbron niet gevonden.**.

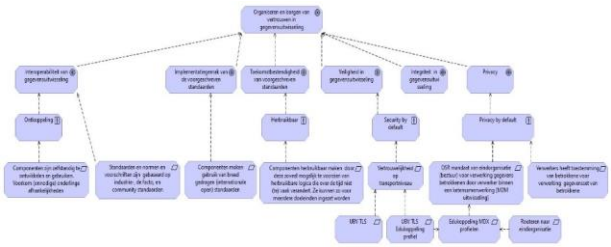


Figuur 3 - Afhankelijkheden met andere Edukoppeling standaarden

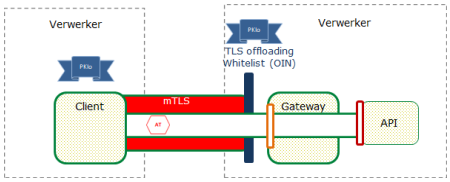
I. Een Authorization Server moet vooraf aan de uitgifte van een Access Token hebben geverifieerd dat de verwerker (client) is gemandateerd.

2) Dit OAuth-profiel gaat uit van het OAuth client credentials profiel (zie Figuur 2). Het gaat om confidential clients die een access token krijgen op basis van hun credentials. Met het access token krijgen zij toegang tot een protected resource. Heel vaak gaat het hier nog om bulkgegevens (persoonsgegevens van meerdere betrokkenen).

3) Het doel is om beter aan te sluiten op een meer actuele architectuurstijl en beveiligingsstandaarden. Deze standaarden zijn ontwikkeld voor moderne infrastructuur, worden door vele platforms ondersteund en ontwikkelaars beschikken over de benodigde kennis. We borgen hiermee ook de interoperabiliteit en herbruikbaarheid van onze onderwijsstandaarden.



4) We kunnen met de toevoeging van OAuth ook twee onafhankelijke beveiligingslagen onderkennen.



I. Verwerker identificatie (OIN-systematiek) en authenticatie (mTLS en PKIo) in de transportlaag.
II. Client (applicatie) die na identificatie en authenticatie (client credentials) bij AS met access token toegang krijgt tot API. Het zijn stappen om aan te sluiten bij de huidige

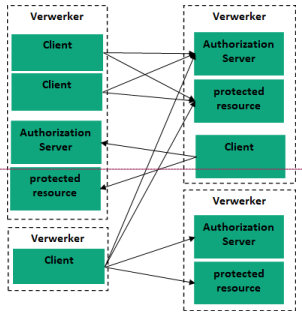
[8] Het Digikoppeling REST API Profiel maakt gebruik van de Kennisplatform REST-API Design Rules (zie https://publicatie.centrumvoorstandaarden.nl/api/adr/)

inrichtingsvormen bij organisaties. Met het OAuth-profiel treden we met de afspraken in prinicipe dieper in het organisatiedomein[9].

5) De huidige inrichtingsvorm is op basis van bearer tokens.
   I. Implementaties van OAuth 2.0 maken (over het algemeen) gebruik van zogenaamde bearer tokens. Een bearer token is een token dat de drager toegang geeft tot de beveiligde API's. Een bearer token is portable en draagt bij aan de ontkoppeling van systemen. Het maakt de API-beveiliging staatloos (stateless) omdat (alle) informatie voor de autorisatievraag in het token zijn opgenomen. Dit kan echter ook als een nadeel gezien worden als het token niet goed beveiligd wordt. De drager is impliciet geautoriseerd en is iedereen die over een kopie van het token weet te beschikken. Ze moeten dus bij opslag en transport beveiligd worden. Bij transport bijvoorbeeld met behulp van Transport Layer Security (TLS). Een meer veilig type token is een proof-of-possession (PoP) token[10]. Hierbij is het token (cryptrografisch) beveiligd doordat deze aan een specifieke drager is gebonden. De drager moet kunnen aantonen dat deze in bezit is van de "sleutel" van het "slot" dat in het token is opgenomen.
   II. Het Access Token is een bearer token. De client wordt geautoriseerd voor toegang tot een API op basis van een Access Token.
   III. Voor client authenticatie bij het token endpoint van de authorization server wordt er ook gebruik gemaakt van een bearer token, ook dit is een JWT[11].
   IV. Het gebruik van een bearer token betekent dat de rechten "bevroren" zijn gedurende de levensduur van het token. Mede hierom[12] is het wenselijk om een access token een zo kort mogelijke levensduur te geven.

6) We gaan er vanuit dat één partij de resource server en de betreffende authorization server beheert.
   I. Een verwerker kan de OAuth authorization server en de resource server rol hebben of de client rol. Het kan zijn dat een bepaalde verwerker alle rollen heeft.

7) In ketens kunnen er meerdere Authorization Servers en Resource Servers[13] combi's zijn die door verschillende verwerkers worden beheerd.
   I. Een partij die als client koppelingen heeft met API's van verschillende partijen moet dus bij betreffende AS van die partij een AT aanvragen.



8) Een Access Token is zelfbeschrijvend[14]

Met opmerkingen [KR9]: Dat betekent ook een registratie van de client in alle auth servers waar je een koppeling mee wilt leggen?. Een OSR als ketenauthserver, zou dat niet mooier zijn?

Met opmerkingen [ER10R9]: Klopt, zie plaatje. OSR zou als Auth server kunnen werken als partijen de beveiling van api's via OSR willen laten verlopen, een token van OSR geeft dan toegang. Dit betekend wel dat OSR alle (beveiligings)voorwaarden voor een AT moet kunnen ondersteunen. Een ander alternatief is dat een decentrale auth server een OSR token nodig heeft op het token end point (als 1 vd randvoorwaarden voor leveren van een AT door decentrale AS

[9] Naast de autorisatie bij de Resoure Server (API) kan een gateway eventueel al een validatie op JWT signature uitvoeren.
[10] Deze vorm van extra beveiliging wordt ondersteund met RFC8705 waar we helaas met PKIo certificaten geen gebruik van kunnen maken.
[11] Hierbij gaan we er vanuit dat bij transport mTLS wordt toegepast. Voor een mogelijk profiel met transport over TLS zou ook Basic Authentication als methode voor client authenticatie toegevoegd kunnen worden.
[12] Ook vanuit beveiliging is het wenselijk om de levensduur zo kort mogelijk te houden.
[13] We gaan bij deze versie uit van een co-located AS en RS combi in het beheer van een bepaalde partij. Er kan op termijn besloten dat er een centrale AS komt (wellicht OSR) voor de uitgifte van client credentials en eventueel de beveiliging van API's bij meerdere verschillende partijen. Dit zou dan wel meer een OSR implementatietraject zijn en een afsprakenstelsel zijn wat wellicht in het Secure API protocol beschreven kan worden. De interfaces van OSR beschrijven dan in zijn geheel het OAuth-profiel.
[14] Self-contained: having all that is needed in itself. Self-contained tokens have a data structure that contains metadata and claims to communicate the identity of the user or client over the wire. A popular format would be JSON Web Tokens (JWT). The recipient of a self-contained token can validate the token locally by checking the signature, expected issuer name and expected audience or scope (https://support.pingidentity.com/s/question/0D51W00007qYzmrSAC/selfcontained-vs-reference-token).

I. Dit OAuth-profiel gaat er vanuit dat introspection niet nodig is. Of andersom, bij toepassing van token introspection is het wenselijk om niet zelfbeschrijvende tokens toe te passen.

9) Token revocation wordt niet toegepast.
   I. De tokens zijn bijvoorbeeld een uur geldig. Naar verwachting is dit kort genoeg om geen token revocation noodzakelijk te achten.

10) De client wordt (handmatig) geregistreerd bij de Authorization Server (zie Client registratie Voordat een client over een access token kan beschikken en toegang krijgt tot een API moet deze eerst bij de betreffende authorization server geregistreerd zijn. Het registratieproces zelf vormt geen onderdeel van dit OAuth-profiel. Hieronder zijn wel een aantal gegevens opgenomen die bij de registratie van een client relevant worden geacht.

1) De client MOET een confidential client zijn.
2) De client MOET over bij de authorization server geregistreerd zijn met de volgende informatie:
   a) client_id
   b) client_name
   c) token_endpoint_auth_method
      i) Conform dit OAuth-profiel MOET dit de waarde “private_key_jwt” bevatten. De publieke sleutel is beschikbaar via de jwks_uri.
   d) grant_types
      i) Conform dit OAuth-profiel MOET dit de waarde "client_credentials” bevatten.
   e) Als de AS scopes ondersteunt dan MOET de client de te gebruiken scopes bij registratie aangeven.
      i) Deze kunnen (out-of-band van dit OAuth-profiel) ook later aangepast worden.

3) De client MOET bij registratie een identifier (client_id) van de authorization server krijgen.
   a) Deze client_id MOET herleidbaar zijn naar de verwerker (OIN).

4) De client MOET een public/private sleutelpaar hebben voor private_key_jwt client authenticatie bij het token endpoint van de authorization server.
   a) De client MOET een PKIo of private CA certificaten voor het JWT gebruiken.
   b) De authorization server gebruikt de publieke sleutel van het client certificaat om de JWT in stap #1 te valideren.
      i) De authorization server MOET voor validatiedoeleinden over de publieke sleutel kunnen beschikken.
      ii) Er MOET in een proces worden voorzien dat certificaten (probleemloos) kunnen worden vervangen.
         (1) Omdat certificaten over tijd kunnen wijzigen MOET de client (meta)data rond de publieke sleutel(s) beschikbaar stellen. Dit MOET via het registreren van een jwks_uri met verwijzing naar het JSON Web Key (JWK) Set (RFC7517) document (zie voorbeeld hieronder).
         (2) Als de client (issuer) een certificaat vervangt dan moet de publieke sleutel in de JSON Key Set met een nieuwe kid toegevoegd worden. Partijen moeten daarom regelmatig dit document updaten.

```
{
   "keys": [{
      "e": "AQAB",
      "use": "sig",
      "alg": "RS256",
      "kty": "RSA",
```

```
        "n": "oNqXxxWuX7LlovO5reRNau5f96K_o3DJx-wK7lrjBmp0qKwNszbbp8MvfrlVs-
oYXfj1rzqAeY6GJF5BETViDTT0i2fEz37J0HGAeTrO7Z5zI5Ure9Cb0lulLOZj1hF8piZzW
W_z_set2NyhafoZ-IG1NSe61mqHu7mTjuHYST84igz-
bPKhkJAVlmPPjHTO51hG9T_roVIkjXnvgqd2dCaJ0ExT2bR96jcyausbkdDNfPtJdfSCA
WYXGQnt0PmIysOHPtCkyFqv5ez8KXT7Q4CAYd7nxwfWNOFRHyLAyF__cYEJlBEK
GyJniSIPtkGBWrbXUQhKF6TEFa-RRRl8Dw==",
        "kid": "1516918956_0"
    }, {
        "e": "AQAB",
        "use": "sig",
        "alg": "RS256",
        "kty": "RSA",
        "n": "kMfHwTp2dIYybtvU-xzF2E3dRJBNBtNbb-d3-
Rm6nRUraxnTwZ6Fr1YpFBd1pnWzLzdtMv7ofCd28nx-
1mfYZ6qhqPWF1RpGe2vVOSTmcu-QpA9h-
rouqRKlv3jvXPn623Z2U1Wml0EIxyIzD3WLu7NkWEKSIcBzeY1TctpO5FSU3EyyCX1U
oIMuvYBP9tiZlc74yIZvky-
qT8Ej3S8L0JqhvD583E_uGMoLowguOl2yYr9zhubiqOxT3VsxvpJCu04TWmvf4XX34IQ
RyAHcPJFQ2QiBfLWvWyc6iP3JJYJvyapwc5vVEismryXnngyBX8NXHZaarMi6g5kTQi8
itw==",
        "kid": "1516918956_1"
    }]
}
```


4.2. Interacties

Verzoek naar Token Endpoint (STAP #1)

Het verzoek voor een access token dat de client naar het token endpoint van de authorization server stuurt MOET aan de volgende eisen voldoen:

1) Het verzoek MOET een HTTP POST over mTLS zijn conform het REST-profiel:
   a) authenticatie van de verwerker op basis de certificaat hiërarchie;
   b) identificatie (verwerker) op basis van OIN.

2) Het verzoek MOET een in de querystring een routeringskenmerk bevatten conform het REST-profiel:
   a) Op basis van het OIN en het routeringskenmerk MOET het mandaat gecontroleerd worden conform het REST-profiel.

3) Het verzoek MOET de a parameter bevatten met de waarde “client_credentials”.

4) Het verzoek MOET de **client_assertion_type** parameter bevatten met de waarde “urn:ietf:params:oauth:client-assertion-type:jwt-bearer”.

5) Het verzoek MOET de **client_assertion** parameter bevatten met de JWT voor client authenticatie.
   a) De client MOET voor ieder verzoek van een bepaalde context een nieuwe JWT genereren.
   b) De client MOET de JWT assertion ondertekenen met de private sleutel van de client.
      i) Het client certificaat MOET een PKIo of een certificaat van een private CA gebruiken voor de JWT.
      ii) De JWT voor client authenticatie MOET voldoen aan RFC7523 (Using JWTs for Client Authentication.
   c) De JWT payload bevat de volgende claims:
      i) iss: een unieke identifier van de uitgever van de verklaring.
         (1) In het verzoek naar de authorization server token endpoint MOET de iss claim gevuld worden met de "client_id".

ii) sub: een unieke identifier van de entiteit die geauthenticeerd is.
   (1) In het verzoek naar de authorization server token endpoint MOET de sub claim gevuld worden met de "client_id".
iii) aud: een unieke identifier van de ontvanger(s) van de verklaring
   (1) In het verzoek naar de authorization server token endpoint MOET de aud claim een waarde bevatten die de authorization server identificeerd. Dit MAG het token endpoint URL zijn.
iv) iat: een timestamp van het moment waarop de JWT is gecreëerd.
v) exp: een timestamp van het moment waarop de JWT verloopt.
vi) jti: een unieke identifier voor de gecreëerde JWT.
vii) Optioneel:
   (1) scope: De client MAG een Access Token aanvragen met een specifieke scope. Het wordt AANBEVOLEN om de scope conform de scope naming conventions van dit profiel toe te passen. Deze zorgt ervoor dat er een logische referentie naar de protected resource is i.c.m uit te voeren actie(s)).
      (a) Het AT is betekenisvol voor AS en RS en betekenisloos voor de client. Op basis hiervan zou kunnen worden aangenomen dat de scope dus niet relevant is voor de client en dus nooit opgenomen hoeft te worden in dit verzoek. Het wordt echter AANBEVOLEN om waar relevant een client meerdere scopes te geven en dat de client een specifieke scope in aanvraag opneemt. Dit is wenselijk gezien het bearer token betreft en het token minimale autorisaties moet bevatten.
   (2) edu_org_id: De client MAG een Access Token aanvragen in de context van een onderwijsorganisatie. De client MOET in dat geval de edu_org_id in het verzoek opnemen. Hiermee kan een meer fijnmazige (data)autorisatie ondersteund worden die door een bepaalde onderwijsorganisatie voor een client is geregistreerd. Hoe een partij dit als onderdeel van de authorization server inricht en hoe de interactie met de onderwijsorganisatie (admin) plaatsvindt is buiten de scope van dit OAuth-profiel.

d) De JWT header bevat de volgende (JSON Web Signature RFC7515) gegevens:
   i) typ: The "typ" (type) Header Parameter is used by JWS applications to declare the media type [IANA.MediaTypes] of this complete JWS.
   ii) alg: The "alg" (algorithm) Header Parameter identifies the cryptographic algorithm used to secure the JWS.
   iii) kid: The "kid" (key ID) Header Parameter is a hint indicating which key was used to secure the JWS.

```
{
   "typ" : "JWT",
   "alg" : "RS256"
   "kid" : “1516918956_0”
}
```

Een voorbeeld van het verzoek (stap #1) wordt weergegeven in het onderstaande Figuur 4.

```
POST /token?edu-to=0000000700099AA00005&edu-from=.. HTTP/1.1
Host: https://as.verwerker.com:443
Content-Length: <length>
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_assertion_type=urn%3Aietf%3Apa
rams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
&client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOi
J0b29sLmNvbSIsInN1YiI6Ind3dy5leGFtcGxlLmNvbSIsImF1ZCI6Imh0dHB
zOi8vd3d3LmV4YW1wbGUuY29tL2x0aS9hdXRoL3R
va2VuIiwiaWF0IjoiMTQ4NTkwNzIwMCIsImV4cCI6IjE0ODU5MDc1MDAiL
CJqdGkiOiIyOWY5MGMwNDdhNDRiMmVjZTczZDAwYTA5MzY0ZDQ5YiJ
9.liArqLDlF-xGcCu8ythy0HIzntxwZ90AYTnwH-daCQQ
```

Figuur 4 - Stap 1: Verzoek van client aan authorization server token endpoint

Verwerking v/h client verzoek bij het AS Token Endpoint

Het verzoek dat in stap #1 is opgesteld wordt door de authorization server gevalideerd.

1) De authorization server MOET controleren of het verzoek via een mTLS verbinding conform het REST-profiel verloopt.
2) De authorization server MOET controleren of het verzoek een querystring met het routeringskenmerk conform het REST-profiel heeft.
3) De authorization server MOET controleren of de client (verwerker OIN/routeringskenmerk) een mandaat heeft voor de betreffende ketensamenwerking conform het REST-profiel.
4) De authorization server MOET controleren of het request de volgende parameters bevat:
   a) grant_type parameter met de waarde “client_credentials”;
      i) Komt de authenticatie methode overeen met de geregistreerde authenitcatiemethode (token_endpoint_auth_method) voor deze client.
   b) client_assertion_type parameter bevatten met de waarde “urn:ietf:params:oauth:client-assertion-type:jwt-bearer”;
   c) client_assertion met een JWT;
      i) De authorization server MOET controleren of de JWT voldoet aan de eisen van de standaarden (RFC7519/RFC7523/RFC7521).
      ii) De authorization server MOET de waarden van de iss, sub, exp, aud and jti claims controleren gevolgd door een verificatie van de ondertekening.
         (1) De authorization server MOET controleren of de JWT ondertekend met het certificaat dat voor deze client is geregistreerd.
   d) als de JWT een scope bevat dan MOET de authorization server controleren of deze voor de client geregistreerd is;
         (1) Als dit niet het geval is dan is het resultaat een AT op basis van een default scope of een foutmelding.

Antwoord van Token Endpoint (STAP #2): verwerking succesvol

Als de autorization server request #1 succesvol heeft gevalideerd, MOET deze een HTTP 200 OK status code respons geven conform RFC6749 sectie 5.1.

1) Het respons media type MAG application/at+jwt" zijn
2) Het access token wordt in de access_token parameter opgenomen
   a) Het access token conformeert aan RFC9068.

b) Het access token MOET ondertekend zijn volgens een algoritme uit het UBV TLS Edukoppeling profiel.

```
Header:

  {"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}

 Claims:

  {
   "iss": "https://authorization-server.example.com/",
   "sub": "5ba552d67",
   "aud":  "https://rs.example.com/",
   "exp": 1639528912,
   "iat": 1618354090,
   "jti" : "dbe39bf3a3ba4238a513f51d6e1691c4",
   "client_id": "s6BhdRkqt3",

  }
```

Figuur 5 - Niet base64-encoded en niet ondertekent Access Token

3) De respons MOET NIET een refresh token bevatten

Antwoord van Token Endpoint (STAP #2): foutmelding RFC6749 (§ 5.2)

Als de autorisatie service request #1 niet succesvol heeft gevalideerd, MOET deze een HTTP 400 Bad Request status code respons geven. In de body is een foutmelding opgenomen conform sectie 5.2 in RFC6749.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
"error" : "invalid_scope"
}
```

Interactie met protected resource (STAP #3)

De client heeft van de authorization server een access token ontvangen. Het access token is voor de client betekenisloos en moet deze niet analyseren.

Het verzoek naar de protected resource MOET aan de volgende eisen voldoen:

1) De client MOET het servercertificaat van de resource server controleren;
   a) Deze MOET voldoen aan de eisen van het REST-profiel;

2) Het verzoek MAG een in de querystring een routeringskenmerk bevatten conform het REST-profiel;

3) Het wordt AANBEVOLEN dat de client het access token opneemt in de authorization request header (RFC2617). Hiervoor MAG ook de RFC6750 form-parameter gebruikt worden.

```
GET /resource/1 HTTP/1.1
Host: provider.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
```

Figuur 6 - Access token in Authorization request header

De resource server MOET het volgende controleren:

1) De resource server MOET controleren of het verzoek via een mTLS verbinding verloopt conform het REST-profiel ;
   a) authenticatie van de verwerker op basis de certificaat hiërarchie;
   b) identificatie (verwerker) op basis van OIN.

2) Het verzoek MOET het access token JWT in de authorization request header bevatten (zie Figuur 6) of als form-parameter de parameter confrom RFC6750;

3) Het Access token wordt gevalideerd conform RFC9068 (H4 Validating JWT Access Tokens).
11)).

12) Een client wordt geauthentiseerd o.b.v. JWT bearer token (conform RFC7523) met een asymmetrische ondertekening.
   I. De JWT is door de client ondertekend met de private key van de client.
   II. Het toepassen van een asymmetrische ondertekening gebruikt een openbaar/privé-sleutelpaar. Verschillende clients van dezelfde verwerker kunnen eventueel een eigen sleutelpaar gebruiken en kunnen zo apart geauthentiseerd worden. Verder wordt over het algemeen aangenomen dat het sleutelbeheer is doorgaans beter beheersbaar is dan symmetrische sleutels.
   III. Met het gebruik van een asymmetrische ondertekening wordt ook key management van belang (zie XXX).

13) We gaan er vanuit dat een autorisatieserver vele verschillende API's ondersteunt (die mogelijke verschillende randvoorwaarden stellen aan de uitgifte van een Access Token). De client moet daarom expliciet bij verzoek de autorisatieserver kunnen doorgeven voor welke API het Access Token gebruikt wordt. Hiervoor ondersteunt dit OAuth-profiel een optioneel gegeven "resource" (conform RFC8707).

14) Hoewel er bij het client credentials grant type vanuit wordt gegaan dat client credentials voldoende zijn om te beschikken over een access token zijn er ook scenario's mogelijk waar aanvullende randvoorwaarden gelden bij het uitgeven van een access token. Deze moet de Authorization Server verifiëren voordat de client een access token krijgt geleverd. Bij dit profiel (via de eisen van het REST-profiel) geldt dat bijvoorbeeld rond het valideren van het mandaat bij het OSR. In dit OAuth-profiel wordt rekening gehouden met nog een scenario waarbij de client kan aangeven dat het een uitwisseling in opdracht van een onderwijsorganisatie wordt uitgevoerd voor een bepaalde set leerlingen/medewerkers (betrokkenen). De beheerder van de AS/RS combi heeft een onderwijsorganisatie-medewerker de data autorisatie laten registreren. Hiervoor ondersteunt dit OAuth-profiel een optioneel gegeven "edu_org_id"[15]. Wanneer de data autorisaties geregistreerd worden en hoe de client over de edu_org_id komt te beschikken bij de aanvraag voor een access token valt wel buiten de scope van dit OAuth-profiel.

15) Voor het definiëren van de scope wordt AANBEVOLEN de naming convention toe te passen. De AS en RS moeten een eenduidig beeld hebben op welke protected resource een bepaalde scope betrekking heeft en welke acties uitgevoerd mogen worden.

16) Het API design MOET conform de principes van het Edukoppeling REST-profiel zijn.
   I. Principe API-51: "Publish OAS at a standard location in JSON-format"
      - Het wordt AANBEVOLEN minimaal OAS versie 3.1 te gebruiken.
        - Vanaf deze OAS versie kan mTLS vereist worden

[15] Dit is vergelijkbaar met routeringskenmerk, maar betreft een interne referentie.

II. Het wordt AANBEVOLEN om in de OAS-specificatie aan te geven dat dit Edukoppeling OAuth-profiel van toepassing is.

4.3. Authorization Server metadata

Om conform OAuth 2.0 te kunnen werken moeten de verschillende rollen eerst informatie uitwisselen voor het registratieproces. Dit Oauth-profiel ondersteunt geen dynamic registration[16]. Ter ondersteuning van de client registratie worden hieronder wel een aantal gegevens benoemd waarmee aan dit Oauth-profiel kan worden voldaan. Het wordt AANBEVOLEN dat client en resource servers deze info in cache houden.

1) issuer: De issuer identifier (URL) van de authorization server.

2) token_endpoint: Het token endpoint (URL) waar de client een verzoek voor een AT heen stuurt.

3) jwks_uri: De locatie (URL) van de JWK Set document. Deze bevat onder andere de (publieke) sleutel(s) voor het valideren van de door de AS geplaatste ondertekeningen.
   a) Voor de publieke sleutel zijn in de JWK Set de volgende gegevens opgenomen:
      i) kid: De key ID van het sleutelpaar voor ondertekening
      ii) kty: Het sleutel type
      iii) alg: het (default) algoritme
   b) Voor de client is het Access Token niet relevant en MOET deze niet verwerken of verifiëren, De jwks info is dus met name van belang voor de resource server. De communicatie tussen AS en RS is in principe buiten scope van dit profiel. Het is wel belangrijk dat de client kan vaststellen dat de AT van de AS afkomstig is. Dit kan op basis van de (m)TLS verbinding.

4) scopes_supported: Een lijst met de scopes die de authorization server ondersteunt.
   a) Het wordt AANBEVOLEN dat als de authorization server scopes verwacht dat er ook een default scope gedefinieerd wordt. Deze kan dan toegepast worden als in het verzoek van de client de scope ontbreekt. Het alternatief is een foutmelding.

5) grant_types_supported: Een lijst (zie RFC7591) met de grant types die de authorization server ondersteunt.
   a) Voor dit profiel moet de lijst de waarde “client_credentials” bevatten.

6) token_endpoint_auth_methods_supported: Een lijst (zie RFC8414) van client authenticatie methoden die de authorization server ondersteunt.
   a) Voor dit profiel moet de lijst de waarde "private_key_jwt" bevatten.

7) token_endpoint_auth_signing_alg_values_supported: Een lijst (zie RFC8414) van signing algoritmes die de authorization server ondersteunt.
   a) Voor dit profiel moet de lijst de waarde "RS256" bevatten. De lijst MOET NIET de waarde "none" bevatten.

4.4. Client registratie

Voordat een client over een access token kan beschikken en toegang krijgt tot een API moet deze eerst bij de betreffende authorization server geregistreerd zijn. Het registratieproces zelf vormt geen onderdeel van dit OAuth-profiel. Hieronder zijn wel een aantal gegevens opgenomen die bij de registratie van een client relevant worden geacht.

[16] Er wordt aangenomen dat een client handmatig wordt geregistreerd.

5) De client MOET een confidential client zijn.
6) De client MOET over bij de authorization server geregistreerd zijn met de volgende informatie:
   a) client_id
   b) client_name
   c) token_endpoint_auth_method
      i) Conform dit OAuth-profiel MOET dit de waarde “private_key_jwt” bevatten. De publieke sleutel is beschikbaar via de jwks_uri.
   d) grant_types
      i) Conform dit OAuth-profiel MOET dit de waarde "client_credentials” bevatten.
   e) Als de AS scopes ondersteunt dan MOET de client de te gebruiken scopes bij registratie aangeven.
      i) Deze kunnen (out-of-band van dit OAuth-profiel) ook later aangepast worden.

7) De client MOET bij registratie een identifier (client_id) van de authorization server krijgen.
   a) Deze client_id MOET herleidbaar zijn naar de verwerker (OIN).

8) De client MOET een public/private sleutelpaar hebben voor private_key_jwt client authenticatie bij het token endpoint van de authorization server.
   a) De client MOET een PKIo of private CA certificaten voor het JWT gebruiken.
   b) De authorization server gebruikt de publieke sleutel van het client certificaat om de JWT in stap #1 te valideren.
      i) De authorization server MOET voor validatiedoeleinden over de publieke sleutel kunnen beschikken.
      ii) Er MOET in een proces worden voorzien dat certificaten (probleemloos) kunnen worden vervangen.
         (1) Omdat certificaten over tijd kunnen wijzigen MOET de client (meta)data rond de publieke sleutel(s) beschikbaar stellen. Dit MOET via het registreren van een jwks_uri met verwijzing naar het JSON Web Key (JWK) Set (RFC7517) document (zie voorbeeld hieronder).
         (2) Als de client (issuer) een certificaat vervangt dan moet de publieke sleutel in de JSON Key Set met een nieuwe kid toegevoegd worden. Partijen moeten daarom regelmatig dit document updaten.


```
{
   "keys": [{
      "e": "AQAB",
      "use": "sig",
      "alg": "RS256",
      "kty": "RSA",
      "n": "oNqXxxWuX7LlovO5reRNau5f96K_o3DJx-wK7lrjBmp0qKwNszbbp8MvfrlVs-
oYXfj1rzqAeY6GJF5BETViDTT0i2fEz37J0HGAeTrO7Z5zI5Ure9Cb0lulLOZj1hF8piZzW
W_z_set2NyhafoZ-IG1NSe61mqHu7mTjuHYST84igz-
bPKhkJAVlmPPjHTO51hG9T_roVIkjXnvgqd2dCaJ0ExT2bR96jcyausbkdDNfPtJdfSCA
WYXGQnt0PmIysOHPtCkyFqv5ez8KXT7Q4CAYd7nxwfWNOFRHyLAyF__cYEJlBEK
GyJniSIPtkGBWrbXUQhKF6TEFa-RRRl8Dw==",
      "kid": "1516918956_0"
   }, {
      "e": "AQAB",
      "use": "sig",
      "alg": "RS256",
      "kty": "RSA",
      "n": "kMfHwTp2dIYybtvU-xzF2E3dRJBNBtNbb-d3-
Rm6nRUraxnTwZ6Fr1YpFBd1pnWzLzdtMv7ofCd28nx-
1mfYZ6qhqPWF1RpGe2vVOSTmcu-QpA9h-
```

```
rouqRKlv3jvXPn623Z2U1Wml0EIxyIzD3WLu7NkWEKSIcBzeY1TctpO5FSU3EyyCX1U
oIMuvYBP9tiZlc74yIZvky-
qT8Ej3S8L0JqhvD583E_uGMoLowguOl2yYr9zhubiqOxT3VsxvpJCu04TWmvf4XX34IQ
RyAHcPJFQ2QiBfLWvWyc6iP3JJYJvyapwc5vVEismryXnngyBX8NXHZaarMi6g5kTQi8
itw==",
        "kid": "1516918956_1"
    }]
}
```


4.5. Interacties

Verzoek naar Token Endpoint (STAP #1)

Het verzoek voor een access token dat de client naar het token endpoint van de authorization server stuurt MOET aan de volgende eisen voldoen:

6) Het verzoek MOET een HTTP POST over mTLS zijn conform het REST-profiel[17]:
   a) authenticatie van de verwerker op basis de certificaat hiërarchie;
   b) identificatie (verwerker) op basis van OIN.

7) Het verzoek MOET een in de querystring een routeringskenmerk bevatten conform het REST-profiel:
   a) Op basis van het OIN en het routeringskenmerk MOET het mandaat gecontroleerd worden conform het REST-profiel[18].

8) Het verzoek MOET de a parameter bevatten met de waarde “client_credentials”.

9) Het verzoek MOET de **client_assertion_type** parameter bevatten met de waarde “urn:ietf:params:oauth:client-assertion-type:jwt-bearer”.

10) Het verzoek MOET de **client_assertion** parameter bevatten met de JWT voor client authenticatie.
    a) De client MOET voor ieder verzoek van een bepaalde context[19] een nieuwe JWT genereren.
    b) De client MOET de JWT assertion ondertekenen met de private sleutel van de client.
       i) Het client certificaat MOET een PKIo of een certificaat van een private CA gebruiken voor de JWT.
       ii) De JWT voor client authenticatie MOET voldoen aan RFC7523 (Using JWTs for Client Authentication.
    c) De JWT payload bevat de volgende claims:
       i) iss: een unieke identifier van de uitgever van de verklaring.
          (1) In het verzoek naar de authorization server token endpoint MOET de iss claim gevuld worden met de "client_id".
       ii) sub: een unieke identifier van de entiteit die geauthenticeerd is.
          (1) In het verzoek naar de authorization server token endpoint MOET de sub claim gevuld worden met de "client_id".
       iii) aud: een unieke identifier van de ontvanger(s) van de verklaring
          (1) In het verzoek naar de authorization server token endpoint MOET de aud claim een waarde bevatten die de authorization server identificeerd. Dit MAG het token endpoint URL zijn.
       iv) iat: een timestamp van het moment waarop de JWT is gecreëerd.
       v) exp: een timestamp van het moment waarop de JWT verloopt.

Met opmerkingen [BD11]: A parameter: is hier tekst weggevallen of is dit de juiste benaming van deze parameter?

[17] Het REST profiel verwijst hiervoor naar het UBV TLS Edukoppeling profiel.
[18] Het REST profiel verwijst hiervoor naar het MDX protocol.
[19] Als er het om gegevens van een bepaalde edu_org_id gaat.

vi) jti: een unieke identifier voor de gecreëerde JWT.
vii) Optioneel:
(1) scope: De client MAG een Access Token aanvragen met een specifieke scope. Het wordt AANBEVOLEN om de scope conform de scope naming conventions van dit profiel toe te passen. Deze zorgt ervoor dat er een logische referentie naar de protected resource is i.c.m uit te voeren actie(s)).
(a) Het AT is betekenisvol voor AS en RS en betekenisloos voor de client. Op basis hiervan zou kunnen worden aangenomen dat de scope dus niet relevant is voor de client en dus nooit opgenomen hoeft te worden in dit verzoek. Het wordt echter AANBEVOLEN om waar relevant een client meerdere scopes te geven en dat de client een specifieke scope in aanvraag opneemt. Dit is wenselijk gezien het bearer token betreft en het token minimale autorisaties moet bevatten.
(2) edu_org_id: De client MAG een Access Token aanvragen in de context van een onderwijsorganisatie. De client MOET in dat geval de edu_org_id in het verzoek opnemen. Hiermee kan een meer fijnmazige (data)autorisatie[20] ondersteund worden die door een bepaalde onderwijsorganisatie voor een client is geregistreerd. Hoe een partij dit als onderdeel van de authorization server inricht en hoe de interactie met de onderwijsorganisatie (admin) plaatsvindt is buiten de scope van dit OAuth-profiel.

d) De JWT header bevat de volgende (JSON Web Signature RFC7515) gegevens:
i) typ: The "typ" (type) Header Parameter is used by JWS applications to declare the media type [IANA.MediaTypes] of this complete JWS.
ii) alg: The "alg" (algorithm) Header Parameter identifies the cryptographic algorithm used to secure the JWS.
iii) kid: The "kid" (key ID) Header Parameter is a hint indicating which key was used to secure the JWS.

```
{
   "typ" : "JWT",
   "alg" : "RS256"
   "kid" : “1516918956_0”
}
```

Een voorbeeld van het verzoek (stap #1) wordt weergegeven in het onderstaande Figuur 4.

```
POST /token?edu-to=0000000700099AA00005&edu-from=.. HTTP/1.1
Host: https://as.verwerker.com:443
Content-Length: <length>
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_assertion_type=urn%3Aietf%3Apa
rams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
&client_assertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOi
J0b29sLmNvbSIsInN1YiI6Ind3dy5leGFtcGxlLmNvbSIsImF1ZCI6Imh0dHB
zOi8vd3d3LmV4YW1wbGUuY29tL2x0aS9hdXRoL3R
va2VuIiwiaWF0IjoiMTQ4NTkwNzIwMCIsImV4cCI6IjE0ODU5MDc1MDAiL
CJqdGkiOiIyOWY5MGMwNDdhNDRiMmVjZTczZDAwYTA5MzY0ZDQ5YiJ
9.liArqLDlF-xGcCu8ythy0HIzntxwZ90AYTnwH-daCQQ
```

Figuur 4 - Stap 1: Verzoek van client aan authorization server token endpoint

Met opmerkingen [ER12]: De assertion puur op authenticatie inrichten of ook bijvoorbeeld scope opnemen? In de uit te geven access token wordt de scope ook opgenomen. Als beide ondertekend zijn hebben we voor beide integriteit en onweerlegbaarheid geregeld. Hoe belangrijk vinden we dit?

Met opmerkingen [ER13]: Er zijn meerdere mogelijkheden om de tot de juiste publieke sleutel te komen. In dit OAuth profiel kan de AS op basis van de jwks_uri de certficaten in cache houden en met de kid in header van het JWT kan het actuele cert gevonden worden

iGOV NL: Het PKIoverheid-certificaat MOET worden opgenomen als x5c- of x5u-parameter, volgens (RFC7517). Het wordt AANBEVOLEN om de opname van het certificaat als x5c-parameter te ondersteunen. Partijen MOGEN ook x5u ondersteunen.

Met opmerkingen [ER14R13]: 1edTech: Message Client JWTs *SHOULD NOT use the JWS x5u, x5c, jku, or jwk Header Parameter fields. Instead, Platforms and Tools should communicate the keys to use for Message JWS Tokens during registration.*

Met opmerkingen [ER15R13]: EK OAUTH: Wij kiezen (voorlopig) de optie waarbij certs te vinden zijn via jwks_uri en vias de kid in de header kan het juiste certificaat gevonden worden. AS heeft client certs (in cache) en kunnen (dagelijks) worden geupate met jwks_uri.

[20] Consent door onderwijsorganisatie voor uitwisseling op gegevensniveau.

Verwerking v/h client verzoek bij het AS Token Endpoint

Het verzoek dat in stap #1 is opgesteld wordt door de authorization server gevalideerd.

5) De authorization server MOET controleren of het verzoek via een mTLS verbinding conform het REST-profiel[21] verloopt.
6) De authorization server MOET controleren of het verzoek een querystring met het routeringskenmerk conform het REST-profiel heeft.
7) De authorization server MOET controleren of de client (verwerker OIN/routeringskenmerk) een mandaat heeft voor de betreffende ketensamenwerking conform het REST-profiel[22].
8) De authorization server MOET controleren of het request de volgende parameters bevat:
   e) grant_type parameter met de waarde “client_credentials”;
      i) Komt de authenticatie methode overeen met de geregistreerde authenitcatiemethode (token_endpoint_auth_method) voor deze client.
   f) client_assertion_type parameter bevatten met de waarde “urn:ietf:params:oauth:client-assertion-type:jwt-bearer”;
   g) client_assertion met een JWT;
      i) De authorization server MOET controleren of de JWT voldoet aan de eisen van de standaarden (RFC7519/RFC7523/RFC7521).
      ii) De authorization server MOET de waarden van de iss, sub, exp, aud and jti claims controleren gevolgd door een verificatie van de ondertekening.
         (1) De authorization server MOET controleren of de JWT ondertekend met het certificaat dat voor deze client is geregistreerd.
   h) als de JWT een scope bevat dan MOET de authorization server controleren of deze voor de client geregistreerd is;
         (1) Als dit niet het geval is dan is het resultaat een AT op basis van een default scope of een foutmelding.

Met opmerkingen [ER16]: Is dit redundant? Zijn in principe de eisen die bij het verzoek gelden (stap #1)

Hangt er ook vanaf of we naast de interacties ook de eisen rond de rollen apart willen definieren

Antwoord van Token Endpoint (STAP #2): verwerking succesvol

Als de autorization server request #1 succesvol heeft gevalideerd, MOET deze een HTTP 200 OK status code respons geven conform RFC6749 sectie 5.1.

4) Het respons media type MAG application/at+jwt" zijn
5) Het access token wordt in de access_token parameter opgenomen
   a) Het access token conformeert aan RFC9068.
   b) Het access token MOET ondertekend zijn volgens een algoritme uit het UBV TLS Edukoppeling profiel.

```
Header:

  {"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}

 Claims:

  {
   "iss": "https://authorization-server.example.com/",
   "sub": "5ba552d67",
   "aud":  "https://rs.example.com/",
   "exp": 1639528912,
   "iat": 1618354090,
   "jti" : "dbe39bf3a3ba4238a513f51d6e1691c4",
   "client_id":"s6BhdRkqt3",

  }
```

Figuur 5 - Niet base64-encoded en niet ondertekent Access Token

[21] Het MDX Secure API REST profiel stelt eisen aan TLS verbinding conform het Edustandaard UBV TLS Edukoppeling profiel
[22] Het MDX Secure API REST profiel stelt eisen aan het mandaat conform het MDX Secure API protocol

6) De respons MOET NIET een refresh token bevatten

Antwoord van Token Endpoint (STAP #2): foutmelding RFC6749 (§ 5.2)

Als de autorisatie service request #1 niet succesvol heeft gevalideerd, MOET deze een HTTP 400 Bad Request status code respons geven. In de body is een foutmelding opgenomen conform sectie 5.2 in RFC6749.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
"error" : "invalid_scope"
}
```

Interactie met protected resource (STAP #3)

De client heeft van de authorization server een access token ontvangen. Het access token is voor de client betekenisloos en moet deze niet analyseren.

Het verzoek naar de protected resource MOET aan de volgende eisen voldoen:

4) De client MOET het servercertificaat van de resource server controleren;
   a) Deze MOET voldoen aan de eisen van het REST-profiel;

5) Het verzoek MAG een in de querystring een routeringskenmerk bevatten conform het REST-profiel;

6) Het wordt AANBEVOLEN dat de client het access token opneemt in de authorization request header (RFC2617). Hiervoor MAG ook de RFC6750 form-parameter gebruikt worden.

```
GET /resource/1 HTTP/1.1
Host: provider.example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
```

Figuur 6 - Access token in Authorization request header

De resource server MOET het volgende controleren:

4) De resource server MOET controleren of het verzoek via een mTLS verbinding verloopt conform het REST-profiel [23];
   a) authenticatie van de verwerker op basis de certificaat hiërarchie;
   b) identificatie (verwerker) op basis van OIN.

5) Het verzoek MOET het access token JWT in de authorization request header bevatten (zie Figuur 6) of als form-parameter de parameter confrom RFC6750;

6) Het Access token wordt gevalideerd conform RFC9068 (H4 Validating JWT Access Tokens).

Met opmerkingen [ER17]: Bij kennisplatform API's wordt de interatie met protected resource als 1 stap (#3) gedefinieerd. We hebben ook de plaat van kennisplatform API's overgenomen. Wellicht wordt in een volgende versie dit gesplitst in het verzoek naar RS (#3a) en de controle van het verzoek door RS (#3b)

Met opmerkingen [ER18]: We gaan er vanuit dat 1 partij de AS en RS beheert. Bij uitgifte van AT is het mandaat al gecontroleerd en hoeft dus niet nog een keer plaats te vinden bij de call naar de RS

[23] Het REST profiel verwijst hiervoor naar het UBV TLS Edukoppeling profiel

edustandaard

5. Bijlage B: Overwegingen & acties

5.1. Overwegingen

We moeten nog een eerste versie vaststellen, maar we onderkennen nu al een aantal punten die we nu nog niet meenemen, maar wel op de roadmap zetten voor een volgende versie. Het betreft het volgende:

1. OAuth versie 2.1
2. Routeringskenmerk o.b.v. een JWT
3. Scope naming notation convention
4. …

OAuth versie 2.1

Er is een nieuwe OAuth versie in ontwikkeling https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-07

This specification replaces and obsoletes the OAuth 2.0 Authorization Framework described in RFC 6749 and the Bearer Token Usage in RFC 6750.

Impact moet bepaald worden.

Routeringskenmerk o.b.v. een JWT

Bij het REST profiel hebben we destijds nog niet voor een JWT gekozen gezien de extra implementatie effort die dit introduceert. We hebben toen ervoor gekozen om het routeringskenmerk op te nemen in query string met het uitgangspunt dat het point-2-pooint verbindingen betreft (geen intermediairs) en de respons bevat impliciet een routeringskenmerk voor dezelfde eindorganisatie.

Voor dit OAuth-profiel hebben we eerder besloten om zowel in het request als de respons een routeringskenmerk in een JWT te definiëren. In deze versie van het OAuth-profiel wordt echter op dit punt afgeweken om beter op het huidige REST profiel te kunnen aansluiten. We gaan dus ook in dit OAuth-profiel uit van een point-2-point uitwisseling waarbij slechts een routeringskenmerk in het request volstaat. Hiermee hebben we een meer vereenvoudigde eerste versie van het OAuth-profiel.

Het opnemen van het routeringskenmerk in een JWT in request en respons waarbij de respons een andere eindorganisatie kan bevatten dan het request kunnen we in een toekomstige versie gaan ondersteunen (eventueel ook in het REST-profiel).

Scope Naming Convention

Willen we een naming convention specificeren voor scopes? Een scopes kan gebruikt worden als een grofmazige access control attribuut. De scope naming convention zou bijvoorbeeld een logische verwijzing naar de resource van een bepaald referentiecomponent kunnen zijn en dit combineren met de toegestane actie:

https://edukoppeling.edustandaard.nl/[RC]/[version]/scope/[ketensamenwerking].[action]

waarbij geldt:

- [RC] het referentiecomponent en resource identificeert e.g. las-student.

Met opmerkingen [ER19]: Je zou hier ook een should van kunnen maken. Een client hoort niet meer op te vragen dan waar hij voor geautoriseerd is. Als de client netjes is gebouwd dan is de foutmelding nooit een probleem.
Echter het kan zijn dat een onderdeel van de automrisatie runtime ingetrokken wordt, zodner dat de client daar al op aangepast had kunnen worden. Je kunt dan wel verder met beperkte functionaliteit. Dat is ook wel weer mooi.
Zo overdenkend zou ik kiezen voor het teruggeven van de autorisaties die je wel hebt ipv een foutmelding. Dus eigenlijk ook geen should... ;-)

- [version] de versie van de service specificatie v1p0, v2p1, etc.
- [ketensamenwerking] de keten waarvoor de autorisatie geldt, bijv eindtoets
- [action] de toegestane actie, bijvoorbeeld:
  - readonly for a set of endpoints that permit read only using the GET verb
  - createpost for a set of endpoints that permit creation using the POST verb
  - update for a set of endpoints that permit changing of an established resource
  - delete for a set of endpoints that permit delete only
  - all for access to all of endpoints supported for the version of the identified specification

5.2. Acties

1. In overige documenten de volgende begrippen wijzigen
   - SaaS-profiel wordt Mandated Data eXchange (MDX) Secure API profiel
   - School en onderwijsinstelling wordt onderwijsorganisatie
2. Begrippen: https://rosa.wikixl.nl/index.php/Speciaal:GegevensBekijken/Begrippen?_single&Begrippenset=Edukoppeling

Met opmerkingen [BD20]: Nog controleren of de begrippen in dit document allemaal conform de eerder vastgestelde begrippenset zijn en of er misschien nieuwe begrippen zijn die moeten worden toegevoegd.

6. Bijlage C: Uitgangspunten discussiestuk versie 0.3

1. SaaS-Context en SaaS profielen zijn begrippen die we in de communicatie kunnen blijven gebruiken, maar we veranderen de naam van de profielen in ‘Secure API-profielen’. We onderkennen de volgende Secure API profielen[24]:
    a. Secure API WUS(be, be-S & be-SE)-profiel
    b. Secure API REST-profiel
    c. Secure API OAuth-profiel
2. In de Edukoppeling documentatie en die van OSR hebben we het over een mandatering. Digikoppeling heeft het in een vergelijkbare procesafspraak (bevoegdheid intermediair/SAAS partij door 'machtigen') over een machtiging. In Edukoppeling blijven we het begrip mandatering hanteren.
3. De procesafspraken rond het OSR[25] worden onderdeel van de normatieve voorschriften voor Edukoppeling. De procesafspraken worden beschreven in een Mandated Data eXchange Protocol (zie **Fout! Verwijzingsbron niet gevonden.**).
4. De Secure API profielen worden toegepast bij de uitwisseling van vertrouwelijke gegevens. Dit kunnen persoonsgegevens, maar ook bedrijfskritische gegevens zijn.
    a. Het gaat om vertrouwelijke gegevens en het juridisch kader (AVG) wordt dus niet expliciet binnen het Secure API Protocol opgenomen. De Edukoppeling Secure API Profielen voldoen wel aan algemeen geldende beveiligings- en privacy maatregelen die bij het verwerken van persoonsgegevens verwacht kunnen worden. Het Edustandaard Certificeringsschema geeft aan wanneer deze toegepast dienen te worden.
    b. We definiëren eigen Edukoppeling begrippen (eindorganisatie en verwerker) en sluiten dus niet direct aan op het juridisch (AVG) kader. Wel is het handig als we voor het begrip ‘verwerker’ nog een ander begrip te gaan gebruiken om verwarring te voorkomen. En moeten wellicht ook de definitie aanscherpen.
5. Het functionele toepassingsgebied van Secure API profielen blijft ongewijzigd.
6. We gebruiken (voorlopig[26]) de internationale open standaard OAuth (RFC6749[27] en RFC6750) als vertrekpunt voor de ontwikkeling van het OAuth profiel.
7. Onze use case (toegang tot bulk gegevens van meerdere betrokkenen) past het beste bij het OAuth client credentials en dit OAuth profiel zal als basis dienen. Het gaat om confidential clients die een Access Token krijgen op basis van hun credentials[28]. De client moet zich bij de AS registreren zodat identificatie mogelijk is om de verificatie van het mandaat bij het OSR uit te voeren en te kunnen bepalen of een Access Token geleverd mag worden.
8. In de context van het Secure API OAuth profiel wordt een meer specifieke rolaanduiding wenselijk. In de context van het OAuth profiel kunnen we namelijk spreken van een verwerker die een OAuth AS en RS beheert en een verwerker als client die de protected resource wil verwerken. De relatie tussen bestaande Edukoppeling rollen en nieuwe OAuth rollen wordt weergegeven in **Fout! Verwijzingsbron niet gevonden.**.

Met opmerkingen [KR21]: Een andere begrip kan ook verwarring stichten. Ik vind ' verwerker' juist een heel mooi generiek begrip waarvan iedereen weet wat ermee bedoeld wordt

Met opmerkingen [BD22R21]: https://rosa.wikixl.nl/index.php/F09b4ec1-4f1b-4016-bca6-4f2ea8f7c992, het gaat echter hier om meer dan alleen persoonsgegevens. De vraag is of dat heel erg is. We kunnen in de toelichting van het begrip ook opnemen dat we dit begrip in de Edukoppeling-context ook gebruiken voor bedrijfskritische gegevens

[24] Met Secure API duiden we profielen aan waarbij ook de machtiging aan een verwerker een rol speelt. Als we in de Architectuur meer profielen gaan ondersteunen, bijvoorbeeld API key, dan is het wellicht beter om per profiel met een beveiligingsniveau aanduiding te gaan werken. De WUS be-S & be-SE profielen bieden extra functionaliteit (integriteit en integriteit icm vertrouwelijkheid op data niveau) doordat de data in transport ondertekend of ondertekend en versleuteld kan worden. Het OAuth-profiel biedt extra beveiliging doordat de autorisatie technisch geborgd is. Bij de uitwerking van de architectuur kunnen wellicht nog verschillende beveiligingsniveaus aan de profielen toekennen.
[25] Nader onderzoek moet nog uitwijzen of alle scenario’s door OSR ondersteund kunnen (gaan) worden.
[26] Het Kennisplatform ontwikkelt nog OAuth profielen. Mochten we op een later moment kunnen aansluiten dan nemen we dat in overweging.
[27] https://datatracker.ietf.org/doc/html/rfc6749 (ook OAuth wordt doorontwikkeld: https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/).
[28] Dus niet op basis van toestemming door een gebruiker (Resource Owner).

9. Verwerkers met de AS/RS combi zijn zelf verantwoordelijk voor het registreren van een client.
10. In OAuth (RFC6749) worden een aantal methoden voor client authenticatie[29] onderkend. Een client wordt geauthentiseerd op basis van een JWT.
11. We adviseren het toepassen van beveiliging best practices, OAuth 2.0 threat model and security considerations [RFC6819]. Ook wordt aanbevolen om kennis te nemen van de OAuth 2.0 security best current practices [OAUTH-SBP] en JSON Web Tokens [JSONWT-BP].

[29] https://datatracker.ietf.org/doc/html/rfc6749#section-2.3

7. Bijlage D: Bronnen

[RFC6749]
The OAuth 2.0 Authorization Framework. D. Hardt, Ed.. IETF. October 2012. Proposed Standard. URL: https://datatracker.ietf.org/doc/html/rfc6749

[RFC6750]
The OAuth 2.0 Authorization Framework: Bearer Token Usage. M. Jones; D. Hardt. IETF. October 2012. Proposed Standard. URL: *https://www.rfc-editor.org/rfc/rfc6750*

[RFC6819]
OAuth 2.0 Threat Model and Security Considerations. T. Lodderstedt, Ed.; M. McGloin; P. Hunt. IETF. January 2013. Informational. URL: *https://datatracker.ietf.org/doc/html/rfc6819*

[RFC7518]
JSON Web Algorithms (JWA), **RS256**

[RFC7519]
JSON Web Token (JWT). *M. Jones; J. Bradley; N. Sakimura. IETF. May 2015. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7519*

[RFC7523]
JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants. M. Jones; B. Campbell; C. Mortimore. IETF. May 2015. Proposed Standard. URL: *https://www.rfc-editor.org/rfc/rfc7523*

[RFC7591]
OAuth 2.0 Dynamic Client Registration Protocol*. J. Richer, Ed.; M. Jones; J. Bradley; M. Machulak; P. Hunt. IETF. July 2015. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7591*

[RFC7662]
OAuth 2.0 Token Introspection. J. Richer, Ed.. IETF. October 2015. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7662

[RFC7800]
Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs). M. Jones; J. Bradley; H. Tschofenig. IETF. April 2016. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7800

[RFC8414]
OAuth 2.0 Authorization Server Metadata. M. Jones; N. Sakimura; J. Bradley. IETF. June 2018. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc8414

[RFC9068]
JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens, V. Bertocci, 2021-10-21 , Proposed Standard URL *https://datatracker.ietf.org/doc/rfc9068/*

Met opmerkingen [ER23]: Sinds het profiel is opgesteld, heeft IETF de RFC 9068 uitgebracht. RFC9068 beschrijft een standaard JWT formaat voor access tokens. Deze RFC meenemen in het profiel kan overwogen worden voor interoperabiliteit.
Concreet wordt met name sectie 3.2.1 van het profiel geraakt. Om dit in lijn met sectie 2.2 van RFC9068 te brengen, zullen de claims 'iat' en 'azp' als verplicht moeten worden toegevoegd. Verder moet de claim 'sub' ook verplicht worden gesteld, dit is lijn met issue #7.
Daarnaast is de claim 'client_id' toegevoegd, die zou overeenkomen met 'azp'. Hoe hiermee om te gaan zal nader uitgewerkt moeten worden.
Daarnaast is er ook een wijziging in the JWT header. De JWT header hoort 'typ' (content-type) 'at+jwt' te krijgen, conform RFC9068 sectie 2.1. Ook hiervoor zal nader uitgewerkt moeten worden hoe hiermee om te gaan.
Bovenstaande is geen volledige analyse, er kan dus nog verdere impact zijn.

8. Bijlage E: Begrippen

3. Kandidaten om toe te voegen aan:
   https://rosa.wikixl.nl/index.php/Speciaal:GegevensBekijken/Begrippen?_single&Begrippenset=Edukoppeling

authorization server (bron RFC6749): The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

bearer tokens (RFC6750)
: A security token with the property that any party in possession of the token (a "bearer") can use the token in any way that any other party in possession of it can. Using a bearer token does not require a bearer to prove possession of cryptographic key material (proof-of-possession).

Claim (1edTech): Piece of information asserted about an entity.

Client (RFC6749): An application making protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices). (bron RFC6749)

confidential clients (RFC6749): Clients capable of maintaining the confidentiality of their credentials (e.g., client implemented on a secure server with restricted access to the client credentials), or capable of secure client authentication using other means. (bron RFC6749)

credentials (W3C VC): A credential is a set of one or more claims made by the same entity. Credentials might also include an identifier and metadata to describe properties of the credential, such as the issuer, the expiry date and time, a representative image, a public key to use for verification purposes, the revocation mechanism, and so on. The metadata might be signed by the issuer. A verifiable credential is a set of tamper-evident claims and metadata that cryptographically prove who issued it.

resource owner (bron RFC6749): An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.

resource server (bron RFC6749) : The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.