Ministerie van Binnenlandse Zaken
Koninkrijksrelaties

eID Koppelvlakspecificaties - OpenID Connect

Programma eID
Eerste draftversie / 11 januari 2019

Inhoudsopgave

Routeringsvoorziening OIDC Koppelvlak

Inleiding koppelvlak DV-RD

Aanleiding

Partijen binnen de overheid of partijen met een taak in het publieke domein (hierna te noemen: dienstverleners) met elektronische dienstverlening zien zich gesteld voor een aantal wettelijke verplichtingen aangaande authenticatie:

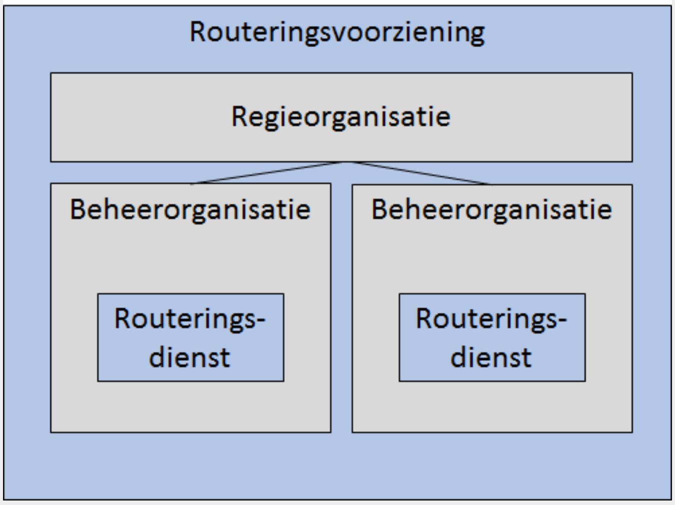
- De Europese eIDAS-verordening legt deze partijen per september 2018 de verplichting op om (onder voorwaarden) ook inlogmiddelen uit andere lidstaten te accepteren. De eerder daarvoor gekozen oplossing is dat dit loopt via de zogenaamde (Herkennings)makelaar in het stelsel Elektronische Toegangsdiensten (eHerkenning), waardoor dienstverleners hierop een aansluiting moeten organiseren.
- De wet Digitale Overheid, welke in voorbereiding is, verplicht dienstverleners om de toegelaten burger-authenticatiemiddelen te ondersteunen, zowel publiek als privaat.

In de consultatie van de eerdere conceptwet hebben de dienstverleners aangegeven ontzorgd te willen worden voor deze verplichtingen, zodat zij niet op meerdere partijen moeten aansluiten, om daarmee aan alle verplichtingen te kunnen voldoen. Zij willen bij voorkeur één koppelvlak, één contract en één factuur. Dienstverleners willen eenvoudig, praktisch en tijdig ontsloten worden voor bovenstaande verplichtingen, tegen acceptabele kosten. Daarop is in de huidige versie van de wet Digtale Overheid de verantwoordelijkheid om te voorzien in een voorziening hiervoor belegd bij de Minister van Binnenlandse Zaken. Doel van deze centrale voorziening is om dienstverleners te voorzien van één koppelvlak, één aanspreekpunt, één contract en één factuur. Deze partij heeft koppelingen met alle relevante authenticatielandschappen en 'vertaalt' deze naar één koppelvlak voor de dienstverlener, en vice versa, conform een algemeen geldende standaard. Deze centrale partij wordt Routeringsvoorziening genoemd.

Definitie

Met *Routeringsvoorziening* wordt het geheel van technische en organisatorische componenten ter ontzorging van publieke dienstverleners bedoeld, zoals beoogd in de wet Digitale Overheid (artikel 5, lid 1, sub c). De Routeringsvoorziening bestaat in de hier gehanteerde Doelsituatie uit twee *Routeringsdiensten* onder regie van een publieke *Regieorganisatie* (beoogd: Logius).

Met *Routeringsdienst* wordt bedoeld: de dienst die onder regie en auspiciën van de Regieorganisatie wordt geleverd aan de dienstverleners door één der beide Routeringsdiensten. Hoewel deze onderling grote overeenkomsten vertonen (met name vanwege interoperabiliteit ten behoeve van dienstverleners) kunnen er verschillen zijn tussen de twee Routeringsdiensten.



Dit document

De Dienstverlener sluit aldus een (*juridische*) overeenkomst met de Regieorganisatie, maar heeft een *technische* relatie met de Routeringsdienst welke in dit document beschreven wordt. Dit document beschrijft de koppelvlakken van de Dienstverlener met de Routeringsdienst op technisch vlak. Voor het organisatorische en juridische perspectief zijn andere documenten beschikbaar. De Routeringsdienst biedt de volgende koppelvlakken aan:

1. Het *DigiD SAML koppelvlak* wordt aangeboden voor Dienstverleners die een stapsgewijze migratie naar de Routeringsdienst wensen. Deze dienstverleners kunnen zo voorlopig gebruik blijven maken van hun huidige koppelvlak, maar beschikken met dit koppelvlak niet over de complete functionaliteit die eID biedt.
2. Een koppelvlak gebaseerd op *Open ID Connect.* Dit koppelvlak biedt de complete eID-functionaliteit inclusief pseudonimisering, attribuutverstrekking en machtigingen.

Leeswijzer

Als eerste worden de use cases beschreven vanuit meer functioneel perspectief. Hierna volgen de technische koppelvlakspecificaties.

Uitgangspunten

Benodigde specificaties

- berichten (functioneel, technisch)
- flow / interfaces / interactie
- bijbehorende processing rules
- metadata specificaties

Principes/aspecten om te bewaken

- security
- "HBCT"
- privacy
- eenvoud voor DV
    - Uitgangspunt is wel een library/framework/toolkit/... Home-grown and selfmade mag, maar kent zeker risico's en is voor eigen rekening.
    - KISS; default makkelijk, moeilijk als nodig voor geavanceerdere use cases
- UX: geen onnodige stappen voor gebruiker
- Sleutelmigraties (waaronder service verhuizen en org-renames) faciliteren

Ontwerpkeuzen

- DV mag gebruiker al optie bieden om een IDP te selecteren (UX optimalisatie).
- Uitzonderingen binnen RV/RD filteren; niet in reguliere aansluit proces oplossen (vertegenwoordiging uitsluiten moet een uitzondering zijn en blijven, ivm recht van gebruiker; KISS-principle).
- Geen dynamische LoA uitvragen; oplossen mbv aparte diensten en dienstensets (complexiteit reductie, varianten inperken, hergebruik mechanisme-dat-al-nodig-is).
- Dynamic registration toepassen voor (technisch) aansluitproces; fallback naar handmatig/self-service-portaal. Self-service-portaal waarschijnlijk sowieso nodig voor meer specifieke use cases (hergebruik standaarden en ondersteuning in bestaande tools).
- Authentication code flow gebruiken; hybrid flow brengt ID-token langs browser, onnodig en ongewenst (levert kleine optimalisatie, introduceert wel meer risico's die met authentication flow (ID-token via backchannel) niet spelen).
-

- Berichtenspecificatie voor het opvragen via artifact binding van oorspronkelijke assertions

Use cases Routeringsvoorziening

Use Cases

- DV-perspectief
    - authentication for single service (uitwerking)
    - authentication for service set (dienstensets) (uitwerking)
    - authentication with multiple recipients (~~geclusterde koppeling~~, portals, misbruikbestrijding, (dienstbemiddeling, meervoudige dienstbemiddeling/dienstenset)) (uitwerking)
    - authentication with attributes (uitwerking)
        - verplicht/optioneel van attributen
        - authentication without identity/pseudonym, just attributes (e.g. 18+)
    - (mogelijkheid) authentication with IDP preselection (en/of MR preselectie ?) (uitwerking)
    - follow-up authentication (re-authentication, ~~step-up~~, switch to (other) representation, switch to another service(set)) ?
    - details original/backing ID/respresentation tokens/assertions
    - authentication at differing LoA
        - Voorkeur: vastleggen in DC met sets, daarmee varieren, niet verder runtime.
    - dynamic & explicit consent (/ signatures ?)
- User-perspectief
    - authentication for self
    - authentication with representation (uitwerking)
    - remember IDP selection; bij DV ok, bij RV ok, over DVs heen het misschien-ooit-een-keer-lijstje
    - authenticate using browser or app from IDP; app-2-app waar mogelijk ondersteunen
- IT/applicatie-architectuur perspectief
    - web-based applicatie
    - native app
    - refresh token
    - m2m / dienstbemiddeling; wanneer mogelijk
    - "groepsaansluiting" = geclusterde koppeling
    - *end-to-end security properties*

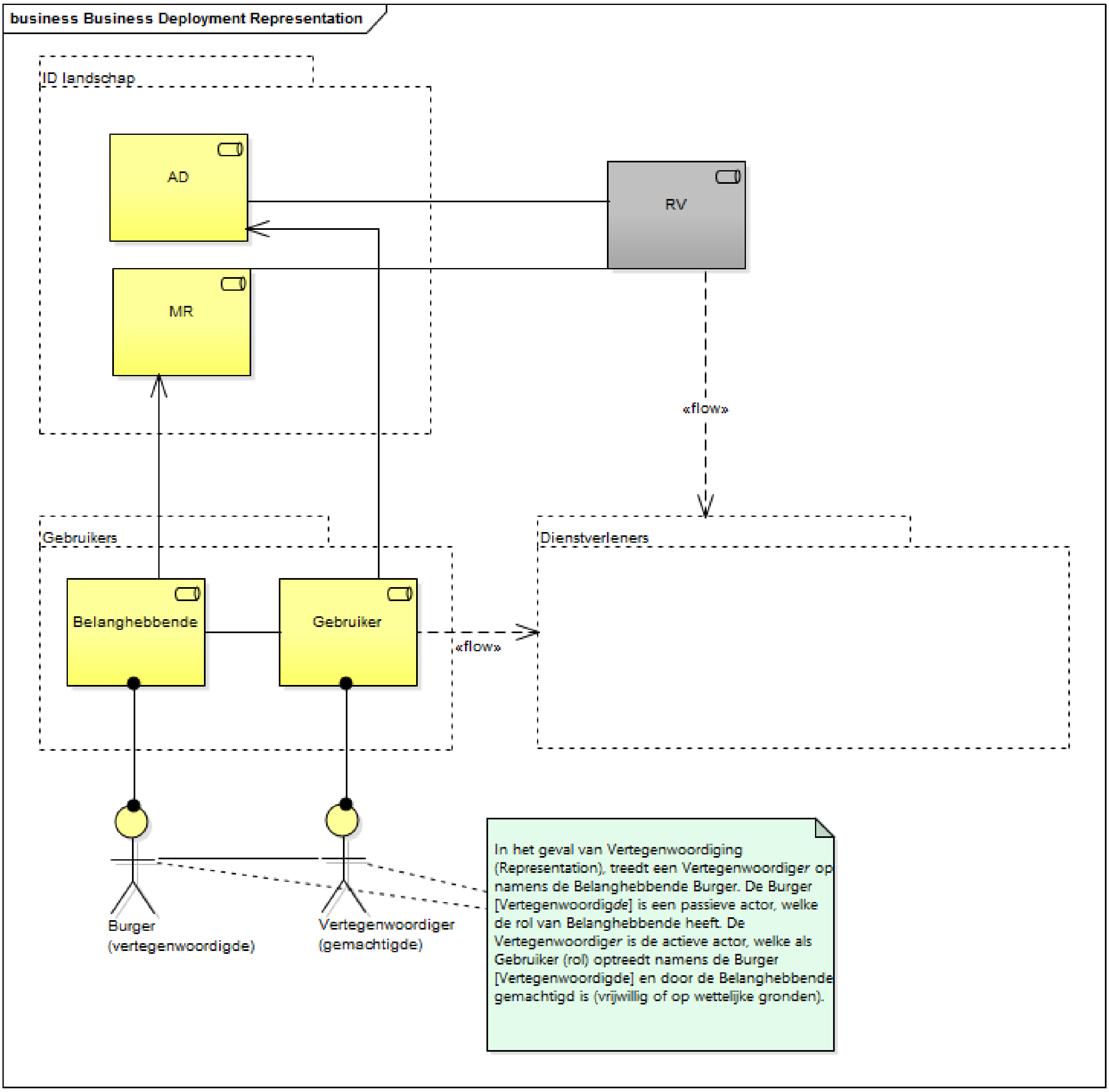
Use case guide

Deze "use case guide" beschrijft verschillende use cases. Deze zijn op verschillende niveaus in de architectuur te plaatsen. Diverse use cases zijn te combineren, het is dus geen kwestie van *de* use case selecteren aangezien *meerdere* use cases *gelijktijdig* van toepassingen kunnen zijn.

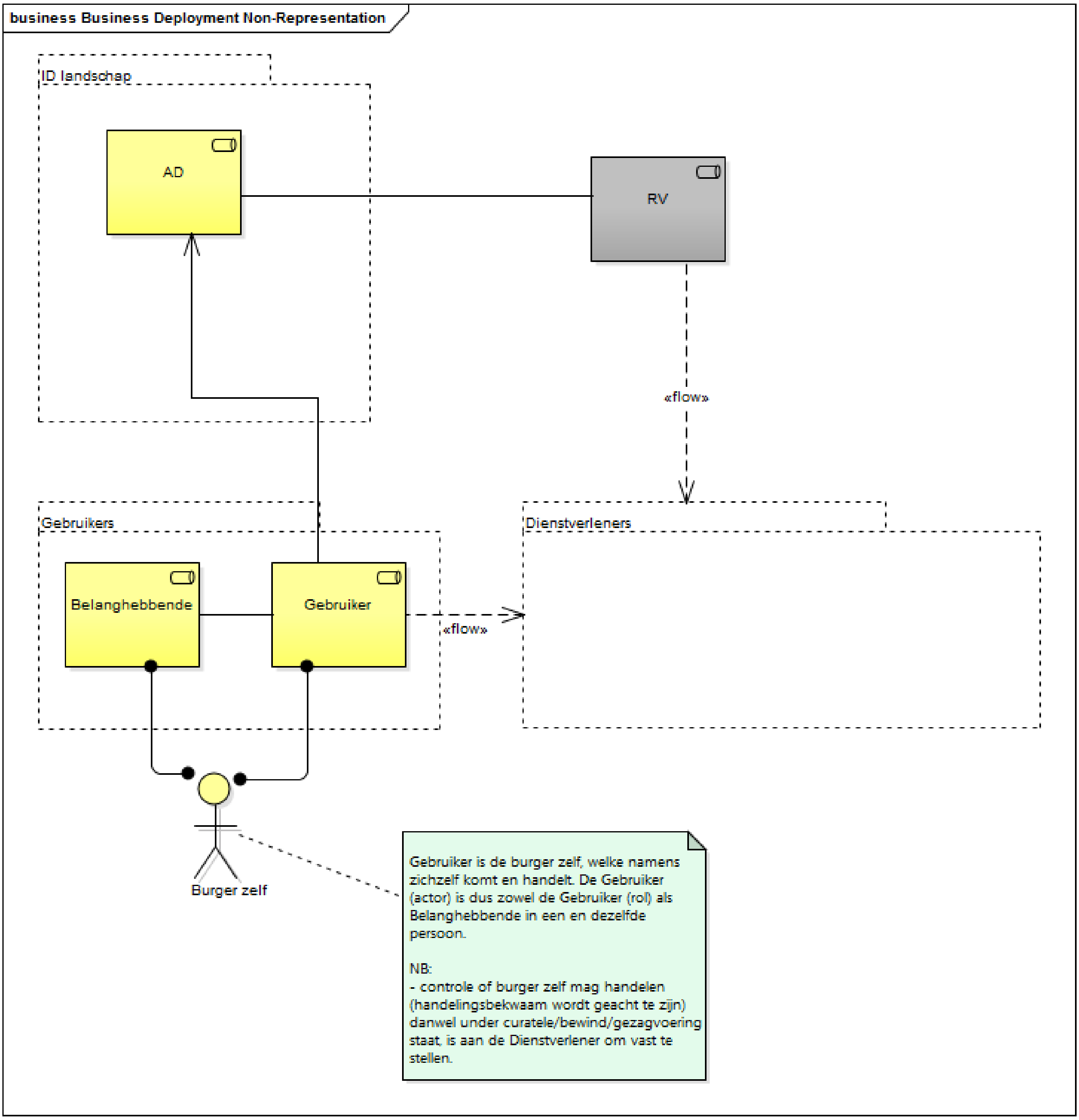
- Vertegenwoordiging (Representation)
- Variabel subject identifier type
- Attribuut verstrekking
  - "Anonieme" attribuut verstrekking
- Dienstbemiddeling (Service Intermediation)
  - Dienstbemiddeling organisatorisch
  - Dienstbemiddeling technisch
- Clusteraansluiting
- Web- versus native applications
- Request by reference

Vertegenwoordiging (*Representation*)

Vertegenwoordiging is een use case op het niveau van de *business layer*.



In het geval van Vertegenwoordiging (Representation), treedt een Vertegenwoordiger op namens de Belanghebbende Burger. De Burger [Vertegenwoordigde] is een passieve actor, welke de rol van Belanghebbende heeft. De Vertegenwoordiger is de actieve actor, welke als Gebruiker (rol) optreedt namens de Burger [Vertegenwoordigde] en door de Belanghebbende gemachtigd is (vrijwillig of op wettelijke gronden).



Gebruiker is de burger zelf, welke namens zichzelf komt en handelt. De Gebruiker (actor) is dus zowel de Gebruiker (rol) als Belanghebbende in een en dezelfde persoon.

NB:
- controle of burger zelf mag handelen (handelingsbekwaam wordt geacht te zijn) danwel under curatele/bewind/gezagvoering staat, is aan de Dienstverlener om vast te stellen.

Variabel subject identifier type

Variabel subject identifier type is een use case op het functionele niveau van de *information layer*. In het geval van een variabel subject identifier type, wordt er een van een mogelijk aantal typen identifiers geleverd.

Doel van een federatief identificatie stelsel, is het leveren van een identiteit van de gebruiker en/of belanghebbende aan vertrouwende partijen. Het type identiteit kan echter wisselen, afhankelijk van het type dienstafnemer of de ondersteunde doelgroep en behoeften van de vertrouwende partij.

Zo kan een vertegenwoordiger zowel een natuurlijk persoon als een rechtspersoon betreffen. Ook kan een dienst aan zowel gebruikers worden aangeboden die al in de BRP staan en al een BSN hebben, als aan gebruikers die nog niet ingeschreven staan (en dus enkele een pseudoniem zullen hebben). Voor Natuurlijke Personen en Rechtspersonen worden echter verschillende typen identifiers gebruikt.

Bij registratie van een Dienst kan worden vastgelegd dat het subject met type *variabel* wordt gevraagd, waarbij de ondersteunde varianten ook worden geregistreerd. Na authenticatie wordt dan het van toepassing zijnde type identifier geleverd en bevat de verklaring ook het type identifier dat geleverd is.

Attribuut verstrekking

Attribuut verstrekking is een use case op het functionele niveau van de *information layer*. In het geval van attribuut verstrekking, worden er naast een identificatie en authenticatie ook een of meer attributen over het Subject geleverd.

Doordat attributen een bepaalde verificatie hebben doorlopen, kan door de ontvangende partij een bepaalde betrouwbaarheid worden veronderstelt. Deze zal doorgaans hoger zijn dan attributen welke direct door de gebruiker aan de ontvangende partij worden verstrekt.

Om reden van privacy dient een gebruiker consent te verlenen voordat zijn of haar attributen worden verstrekt. Hiervoor en vanuit transparantie (ook een privacy maatregel) is het daarom nodig om voorafgaand aan de vraag om attribuut verstrekking de gevraagde attributen, inclusief een doelbinding en privacy statement, te registreren voor de betreffende dienst.

Attributen kunnen als 'verplicht' of als 'optioneel' worden gevraagd. Indien een gebruiker geen consent geeft of de betreffende attributen niet geregistreerd zijn voor de gebruiker, worden attributen niet geleverd. Bij optionele attributen zal de authenticatie worden geleverd zonder deze attributen, het is dan aan de ontvangende partij om toegangsverlening of dienstverlening eventueel aan te passen op de ontbrekende attributen. Ook als het gaat om als 'verplicht' gevraagde attributen, zal de authenticatie worden geleverd *zonder* deze attributen. Het is dan aan de ontvangende partij om toegangsverlening of dienstverlening te weigeren en aan de gebruiker uit te leggen waarom dit geweigerd wordt en eventuele hulp te geven hoe verder te handelen.

"Anonieme" attribuut verstrekking

Een variant op de use case van attribuut verstrekking, is "anonieme" attribuut verstrekking. Dit is een vergelijkbare met reguliere attribuut verstrekking waarbij er wel authenticatie wordt toegepast, maar geen identiteit wordt verstrekt.

Bij "anonieme" attribuut verstrekking wordt geen identifier verstrekt, enkel attributen. Hiermee wordt effectief Attribute Based Authenticatie (of attribute based credentials) bereikt.

Om binnen bestaande protocollen te blijven, kan een "*transient*" identifier worden verstrekt. Een dergelijke "*transient*" identifier is vluchtig en zal iedere keer wisselen, waardoor een terugkerende gebruiker niet te koppelen is aan een individu of zelfs een eerder bezoek.

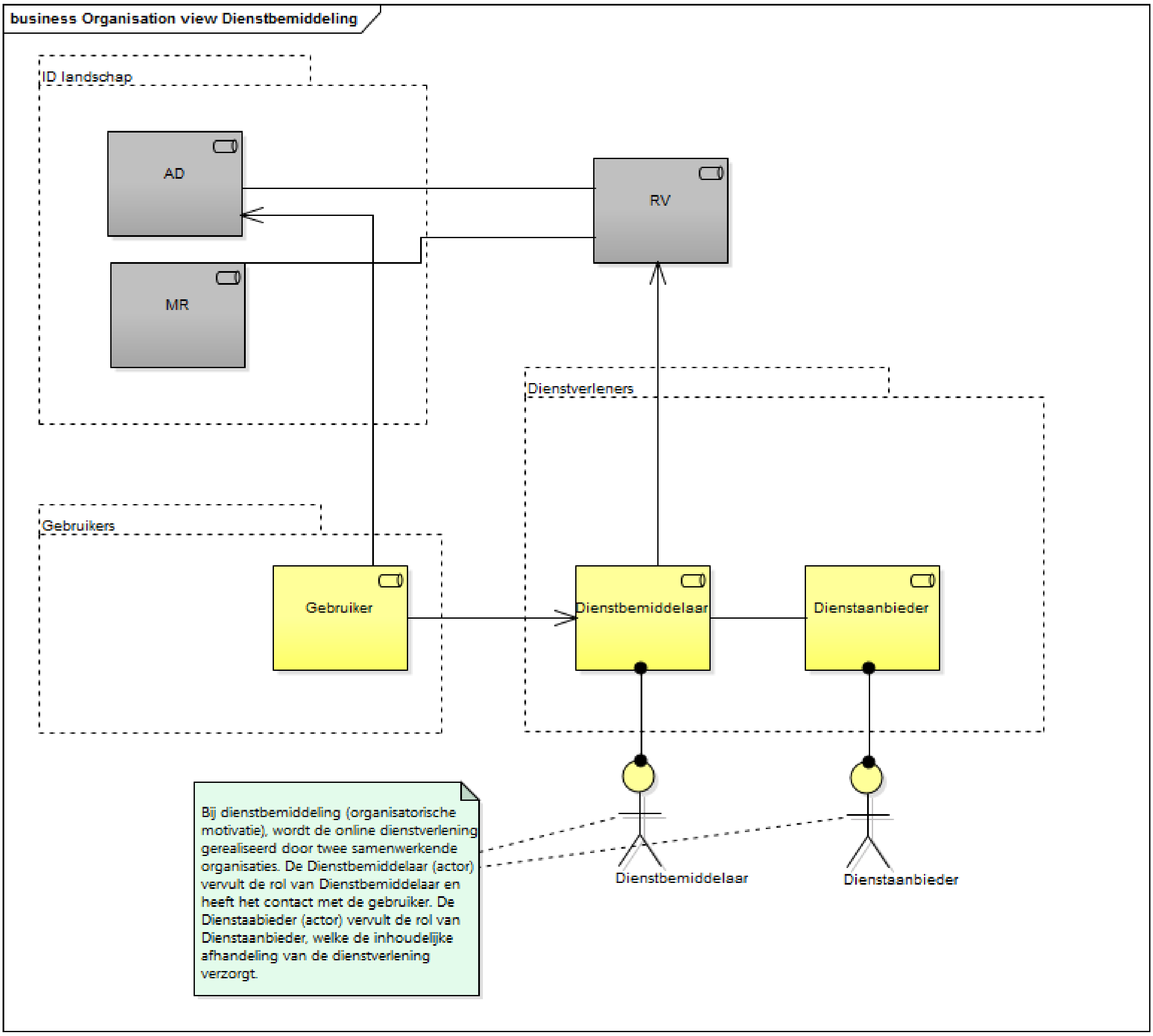
> ⓘ Hoewel de naamgeving van deze use case "anoniem" bevat, hangt de mate van anonimiteit af van het gevraagde attribuut of de combinatie van attributen. Sommige (combinaties van) attributen kunnen herleidbaar tot individuen zijn. Er dient daarom altijd terughoudend te worden omgegaan met attribuut vragen en zo min mogelijk en generiek mogelijke attributen uit te vragen.

Dienstbemiddeling (*Service Intermediation*)

Dienstbemiddeling is een use case op het niveau van de *business layer*. Deze treed op als een dienst door een samenwerking van twee (gescheiden) organisaties wordt geleverd.

Dienstbemiddeling is daarnaast ook in te zetten als technische (beveiligings)maatregel op het niveau van de *application layer*.

Dienstbemiddeling organisatorisch

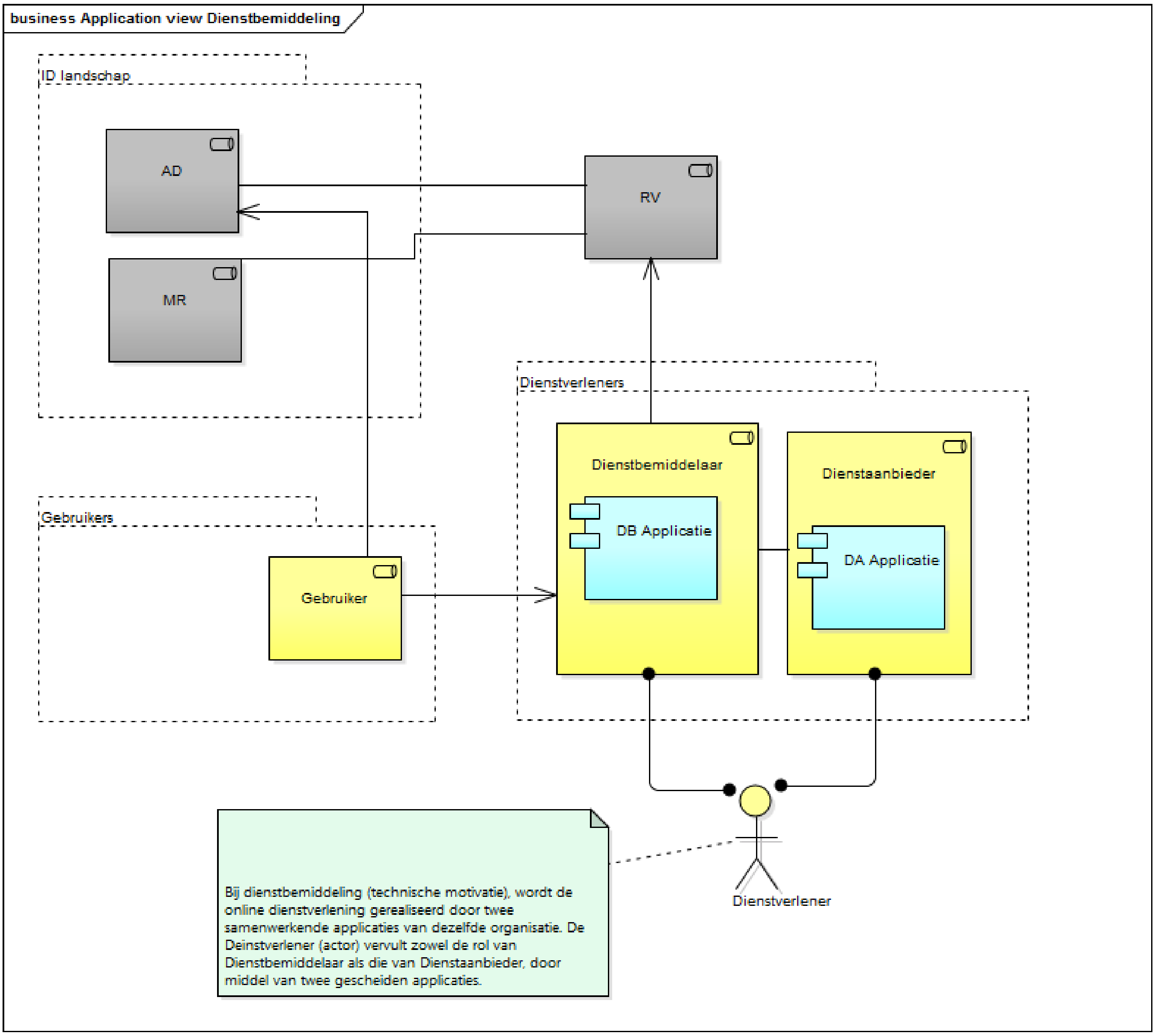


Bij dienstbemiddeling (organisatorische motivatie), wordt de online dienstverlening gerealiseerd door twee samenwerkende organisaties. De Dienstbemiddelaar (actor) vervult de rol van Dienstbemiddelaar en heeft het contact met de gebruiker. De Dienstaabieder (actor) vervult de rol van Dienstaanbieder, welke de inhoudelijke afhandeling van de dienstverlening verzorgt.

De Dienstbemiddelaar en Dienstaanbieder zijn in deze gescheiden organisaties. Door het gebruik van dienstbemiddeling worden een aantal doelen behaald:

- De samenwerking wordt expliciet zichtbaar gemaakt voor de Gebruiker zodat deze geïnformeerd wordt over de beide organisaties die betrokken zijn bij de dienstverlening, waarvoor de Gebruiker ook om consent wordt gevraagd.
- De Dienstaanbieder en Dienstbemiddelaar zijn onafhankelijk opererende organisaties; de Dienstaanbieder is geen onderaannemer/leverancier van de Dienstbemiddelaar of vice versa. Dat sluit niet uit dat er wel een vorm van contractering of mogelijk verwerkersovereenkomst nodig of wenselijk kan zijn.
- De Dienstaanbieder krijgt een verklaring van de Routeringsvoorziening, waarmee zekerheid over de identiteit (en bevoegdheid) van Gebruiker en Belanghebbende krijgt.
- De Dienstaanbieder hoeft de Dienstbemiddelaar geen "*carte blanche*" te geven; de Dienstbemiddelaar kan alleen met instemming van de Gebruiker de dienstafname faciliteren. Er is geen "*high trust*" relatie tussen DB en DA nodig, waarin De Dienstaanbieder volledig op controle van authenticatie en autorisatie bij de Dienstbemiddelaar moet vertrouwen.
- Ondersteuning van meerdere samenwerkingen van een Dienstbemiddelaar of een Dienstaanbieder is eenvoudig en transparant schaalbaar, doordat niet iedere samenwerking 1-op-1 als aparte dienst vastgelegd hoeft te worden.
- De Dienstbemiddelaar krijgt minder verplichtingen en aansprakelijkheden – en daarmee minder risico's – dan het geval is wanneer geen Dienstbemiddeling wordt toegepast.
- De Gebruiker is verantwoordelijk voor het handelen en afname van de dienst; de Dienstbemiddelaar verzorgt enkel de technische realisatie van de dienstafname (bijvoorbeeld in de vorm van een SaaS oplossing).
- Een Gebruiker kiest – wanneer meerdere Dienstbemiddelaars voor eenzelfde dienst beschikbaar zijn – zelf via welke Dienstbemiddelaar hij/zij de dienst afneemt. Indien de Dienstaanbieder ook direct zelf de dienst aanbiedt, kan een Gebruiker ook kiezen de dienst rechtstreeks af te nemen.
- Een Gebruiker welke geen gebruik wenst te maken van een bepaalde Dienstbemiddelaar, wordt ook niet blootgesteld aan risico's bij die Dienstbemiddelaar; omdat geen consent wordt gegeven, zal de Dienstaanbieder de dienst ook niet voor die Gebruiker via de betreffende Dienstbemiddelaar uitvoeren.

Dienstbemiddeling technisch



Bij dienstbemiddeling (technische motivatie), wordt de online dienstverlening gerealiseerd door twee samenwerkende applicaties van dezelfde organisatie. De Deinstverlener (actor) vervult zowel de rol van Dienstbemiddelaar als die van Dienstaanbieder, door middel van twee gescheiden applicaties.

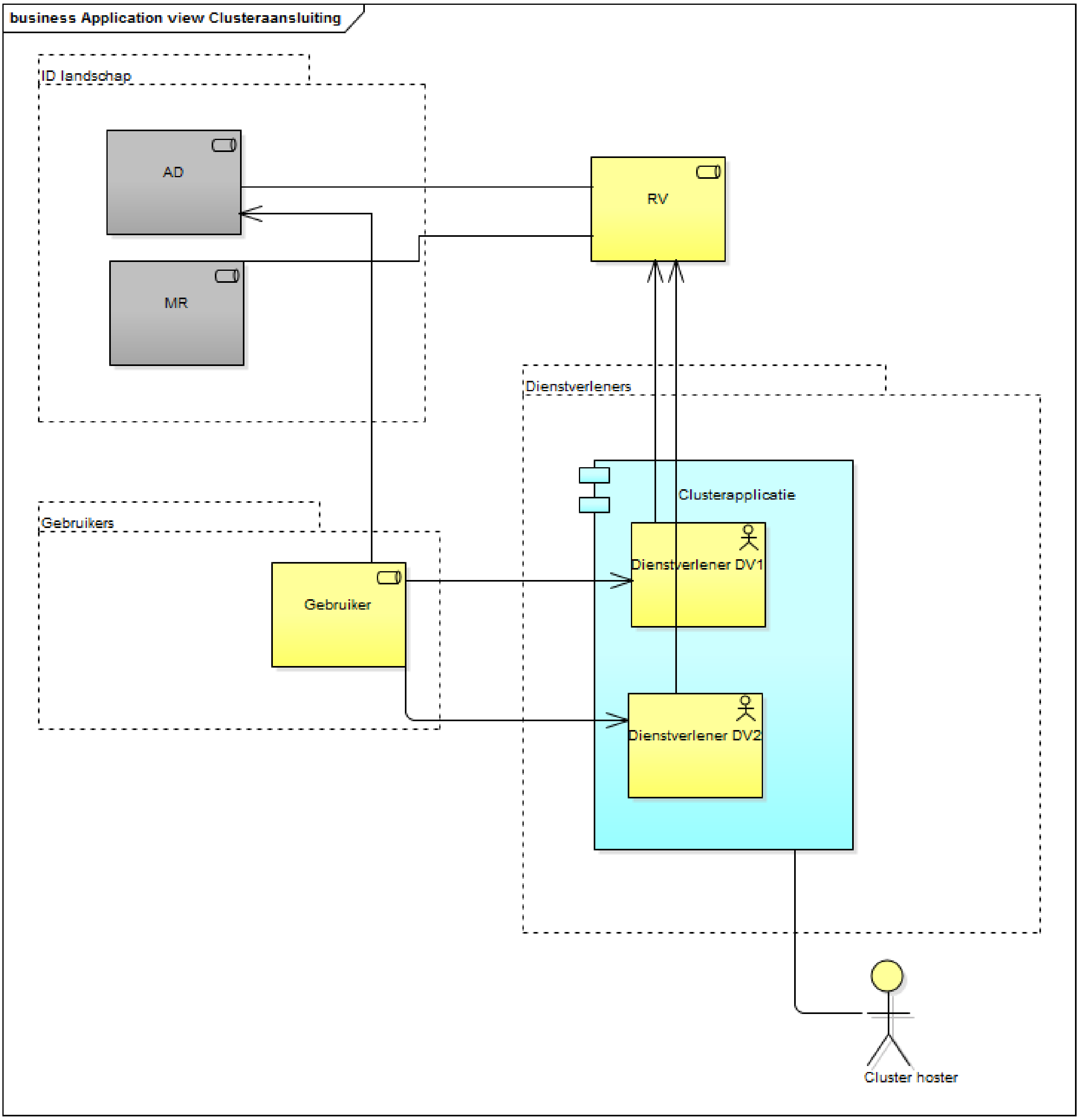
De Dienstbemiddelaar en Dienstverlener zijn in deze een en dezelfde organisatie. Reden om in dit geval Dienstbemiddeling toe te passen zijn:

- Security-by-design: bij scheiding van de *front-end* en *back-end*, biedt Dienstbemiddeling nadrukkelijk de mogelijkheid voor de back-end om authenticatie/authorizatie te controleren. Dit voorkomt een "*high trust*" afhankelijkheid van de *back-end* op de *front-end*.
- multi-channel: wanneer gewerkt wordt met verschillende *front-ends* voor verschillende kanalen (tbv verschillende doelgroepen of type applicaties), kan met behulp van Dienstbemiddeling de authenticatie /authorizatie op een wijze in de back-end worden gecontroleerd. Ook indien de *back-end* direct als API toegankelijk wordt gemaakt voor derden (= organisatorische Dienstbemiddeling) of standalone applicaties.

Dienstbemiddeling kan zowel in organisatorisch als in technische vorm worden toegepast. Ook het gelijktijdig toepassen van Dienstbemiddeling vanuit zowel organisatorische als technische motivatie is mogelijk.

Clusteraansluiting

Een clusteraansluiting is een use case op het niveau van de application layer, welke echter uit een organisatorische oorzaak (business layer) volgt. Een clusteraansluiting is een vorm waarin dienstverlening van meerdere organisaties (meerdere actoren Dienstverlener) in een enkel technische systeem worden gehost. Dit concept staat ook wel bekend als " *multi-tenant*".



Om technische en beheersmatige redenen kan het wenselijk zijn de clusterapplicatie expliciet te erkennen en betrekken in de technische realisatie. Een clusteraansluiting heeft de volgende kenmerken:

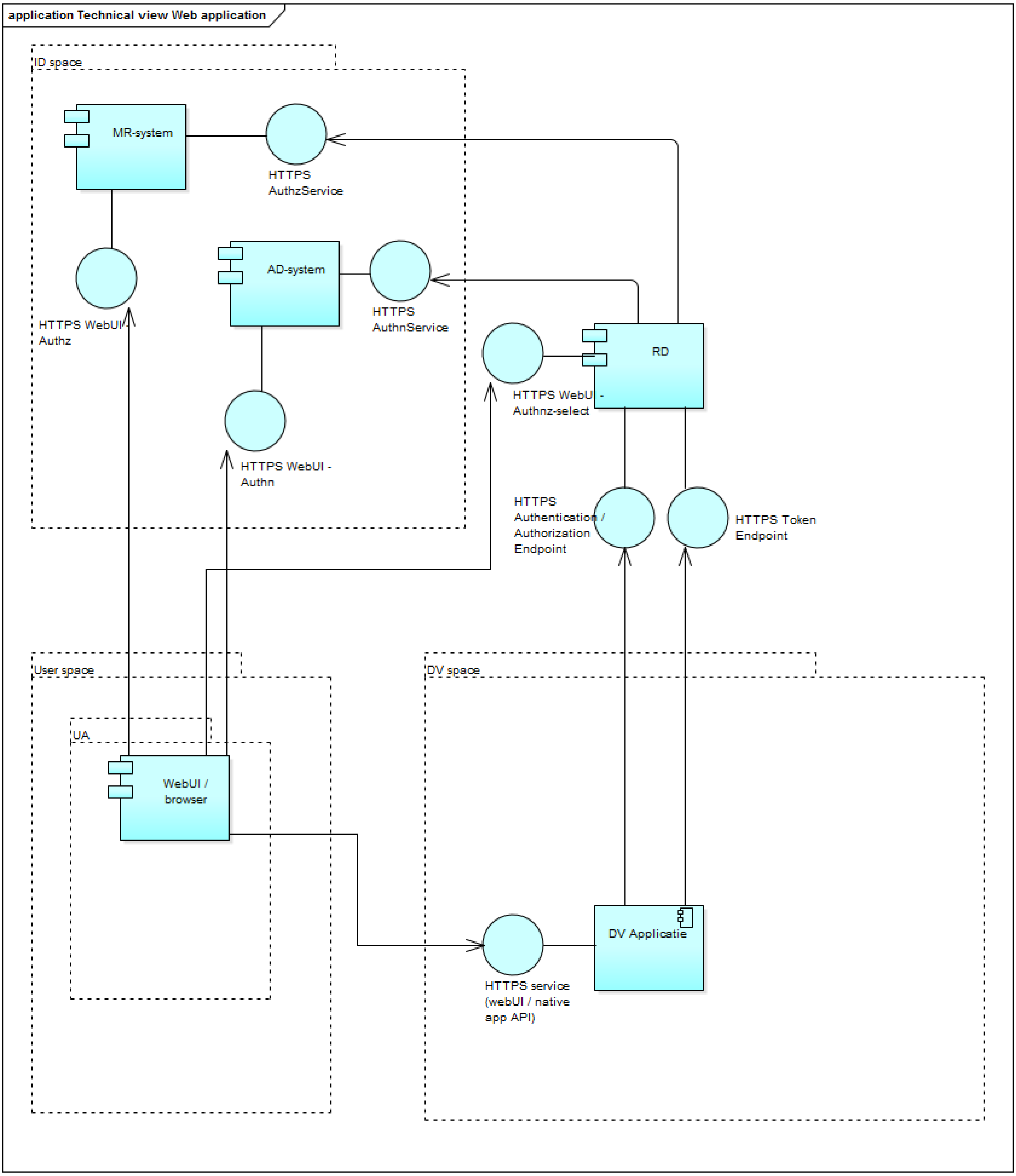
- Het Clustersysteem *host* een of meer organisatorisch gescheiden Dienstverleners.
    - Het Clustersysteem MOET minstens een logische scheiding aanbrengen tussen verschillende Dienstverleners binnen een systeem, voor wat betreft de data en verwerking en beheer van die data.
- De Cluster hoster is een aannemer/leverancier van de Dienstverlener(s).
- De Dienstverlener is geheel verantwoordelijk voor de dienstverlening, waaronder ook voor informatiebeveiliging.
- De Dienstverlener heeft een contract en verwerkersovereenkomst met de Cluster hoster.
- De Cluster hoster is geen actieve partij in de dienstverlening. Voor de Gebruiker is het dan ook geheel transparant of er wel of niet gebruik wordt gemaakt van een Clusteraansluiting, de Cluster hoster is niet zichtbaar voor de Gebruiker.

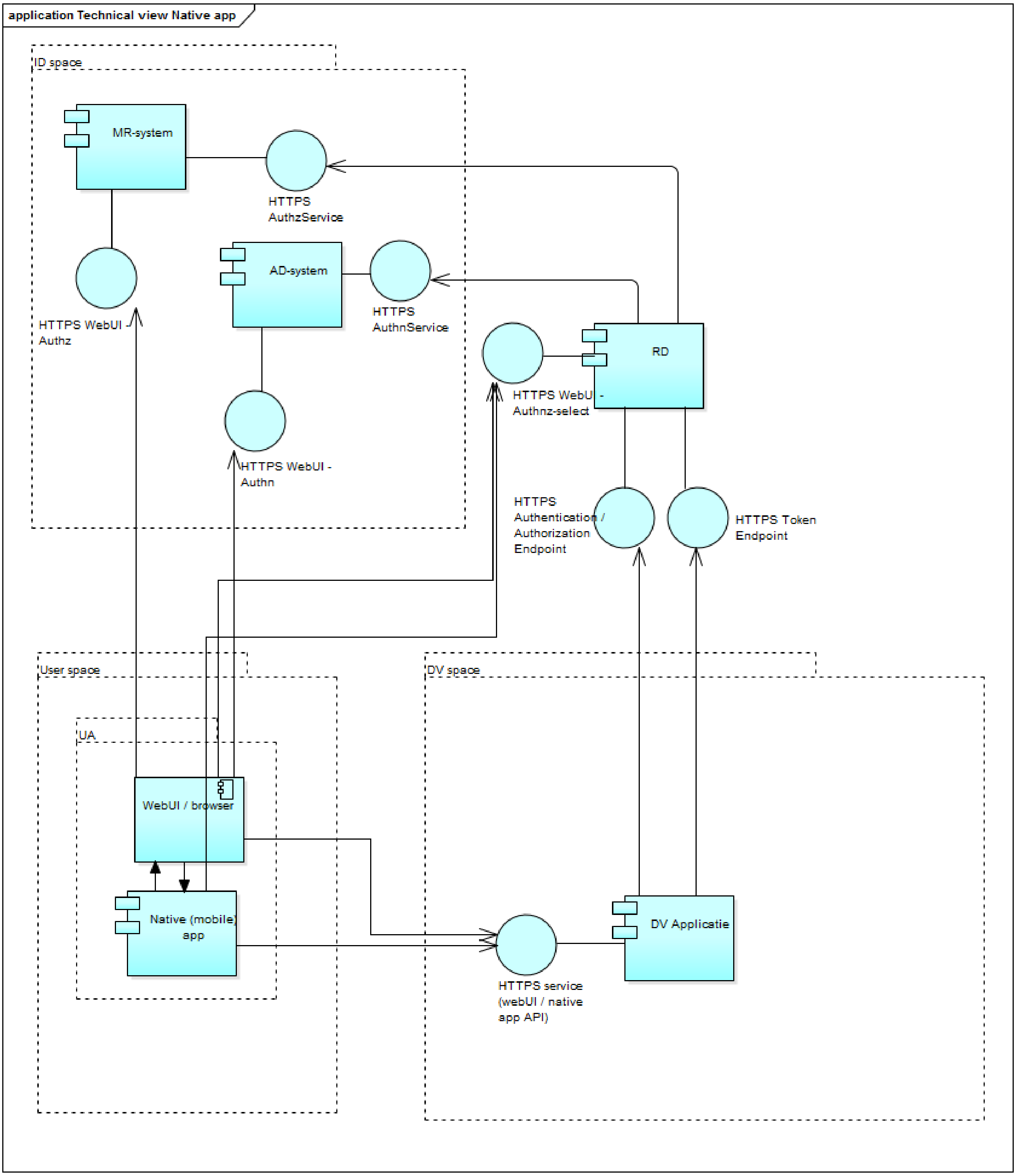
De Dienstverleners welke gehost worden binnen een Clustersysteem kunnen zowel als Dienstverlener (rol) als Dienstbemiddelaar (rol) en/of Dienstaanbieder (rol) optreden.

Web- versus native applications

De use cases van een *web-* of een *native application* (ook wel *mobile app* genoemd) is een onderscheid op het niveau van de technical / application layer. Het concept van een web-applicatie is dat deze met behulp van een browser wordt gebruikt, terwijl bij een *native* of *mobile app* de software direct op het *end-user device* draait. Beiden zijn instanties van een zogeheten *user-agent.*

Hoewel het zuiver gezien een verschil van technische aard is, kunnen gebruikers het wel als functioneel verschillend ervaren. Daarnaast kan het zijn dat er om technische reden daadwerkelijk functionele verschillen zijn, bijvoorbeeld in de interactie met mogelijkheden van het gebruikte *end-user device*, zoals interactie met andere *apps* of aanwezige hardware, bijvoorbeeld een camera. Tenslotte is het onderscheid relevant vanuit het oogpunt van informatiebeveiliging.


application Technical view Web application
ID space
MR-system
HTTPS AuthzService
HTTPS WebUI - Authz
AD-system
HTTPS AuthnService
HTTPS WebUI - Authn
RD
HTTPS WebUI - Authnz-select
HTTPS Authentication / Authorization Endpoint
HTTPS Token Endpoint
User space
UA
WebUI / browser
DV space
HTTPS service (webUI / native app API)
DV Applicatie



Een web-applicatie heeft – in het algemeen gesproken in de context van dit profiel – de volgende kenmerken:

- De Gebruiker heeft een browser, waarmee hij de website van de Dienstverlener bezoekt en de web-applicatie start.

- De Gebruiker installeert geen specifieke software op zijn *end-user device*, maar gebruikt een generieke (web) browser.
- Een browser spreekt de DV applicatie als webserver aan, met gangbare web-protocollen en -technologieën (HTTP, HTML, CSS).
- Een browser kan tegelijkertijd meerdere web-applicaties uitvoeren en optioneel interacteren met *native apps* geinstalleerd op het *end-user device*.
- Een web-applicatie is doorgaans alleen *online* te gebruiken.
- Een web-applicatie is vaak *statefull*, waarbij de browser en webserver (DV Applicatie) een sessie onderhouden (bijvoorbeeld dmv cookies).

Een native app – in het algemeen gesproken in de context van dit profiel – heeft de volgende kenmerken:

- De Gebruiker installeert software (de *app*) op zijn *end-user device*.
- De *native app* kan zowel *online* als *offline* bruikbaar zijn. In de context van dit profiel is alleen *online* gebruik relevant, aangezien authenticatie/autorisatie van Gebruiker naar de Dienstverlener wordt bewerkstelligd. Authenticatie voor gebruik lokaal op het end-user device zelf is nadrukkelijk buiten de doelstelling van dit profiel.
- Een *native app* kan zowel gangbare als *proprietary* protocollen en technieken toepassen.
- Een *native app* is doorgaans specialistisch en kan interacteren met andere apps voor andere toepassing welke zijn geinstalleerd op hetzelfde *end-user device*, waaronder ook een generieke browser.
- Een *native app* kan zowel door de Dienstverlener als door derden beschikbaar worden gesteld.
- Een *native app* kan zowel *statefull* als *stateless* werken.

TODO: eenduidig benoemen wanneer "in case of native app" van toepassing is.

⚠ NB: het onderscheid tussen een web- en native app is niet altijd even duidelijk. Ook zijn beide voortdurend in verdere ontwikkeling, . Hierdoor zijn de bovengenoemde criteria niet altijd eenduidig en aan verandering onderhevig.

Zo zijn er applicaties die in een web/browser draaien maar zich in steeds verdere mate als een *native app* gedragen en krijgen browsers steeds meer mogelijkheden voor het gebruik van hardware mogelijkheden van het *end-user device*. Andersom zijn er native apps die praktisch gezien een browser *embedden* en er eigenlijk sprake is een web-applicatie, deze variant wordt ook wel *hybrid app* genoemd (RFC 8252).

Mede hierom is de interactie van de *user-agent* met de DV Applicatie weergegeven als één interface. Deze kan zowel een web-applicatie, een API voor *native apps*, als gemengde of hybride varianten hiervan bedienen.

ⓘ NB: Er wordt vaak gesproken van *mobile apps*. Omdat *apps* niet gebonden zijn aan het gebruik van *mobile devices* , maar ook steeds meer worden toegepast op allerlei verschijningsvormen, wordt er in dit profiel gesproken over *native apps*. Of dit op een smartphone, tablet, laptop, desktop, mainframe, smart-tv, auto of IoT-device draait, is verder in de context van dit profiel niet relevant. Het gebruik van *native app* als naam komt overeen met gangbare standaarden, bijvoorbeeld RFC 8252.

Request by reference

Technical - TODO.

Applicable:

- keus DV
- cluster verplicht

- native app verplicht?

Use cases uitgewerkt

Let op. Voor de definities van de use cases, zie PSA routeringsvoorziening. Deze is autoritatief tav definitie use cases.

- Authentication for single service (GUC01, AUC02)
- Authentication for service set (AUC03)
- Authentication with attributes (AUC04)
- Authentication with representation (AUC06)
- Authentication with IDP preselection (AUC01)

Authentication for single service (GUC01, AUC02)

Achtergrond

Een burger wil via de webdienst van een (semi-)overheidsdienstverlener een dienst afnemen waarvoor hij/zij zich dient te authenticeren.

Actoren

Actor	Rolbeschrijving
Gebruiker	Wenst gebruik te maken van dienstverlening
Dienstverlener	(Semi-) overheidsdienstverlener welke dienst levert aan geauthenticeerde burgers
Routeringsvoorziening	Voorziening welke de dienstverlener in staat stelt aan te sluiten op meerdere authenticatiediensten
Authenticatiedienst	Publieke of Private partij welke authenticatie-verklaringen afgeeft

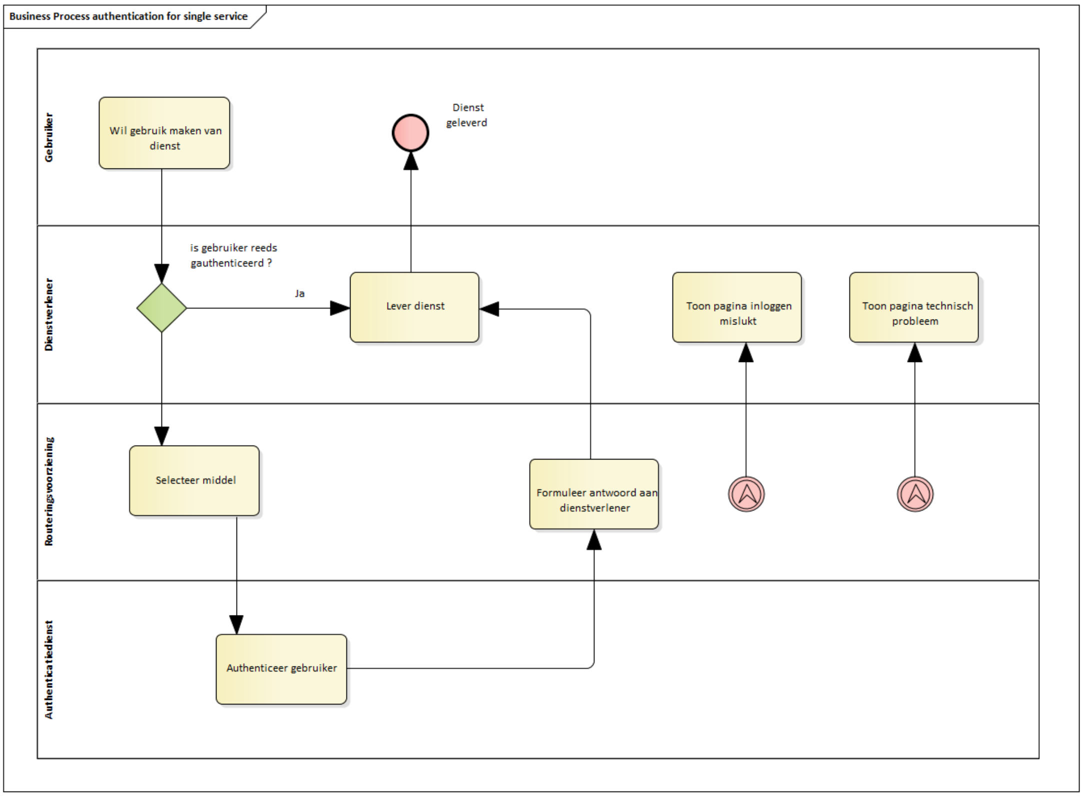
Functionele beschrijving

De meest basale use case is wanneer een burger gebruik wil maken van elektronische dienstverlening bij een overheidsdienst ten bate van zichzelf.

Hierbij bezoekt deze de webdienst van de dienstverlener waar hij/zij al dan niet stappen neemt om kenbaar te maken dat deze een dienst wenst af te nemen waarvoor authenticatie vereist is.

De dienstverlener zal vervolgens de burger doorsturen naar de routeringsvoorziening, alwaar deze kenbaar maakt via welke authenticatiedienst hij/zij wenst te authenticeren en/of deze als gemachtigde wenst te handelen namens een ander.

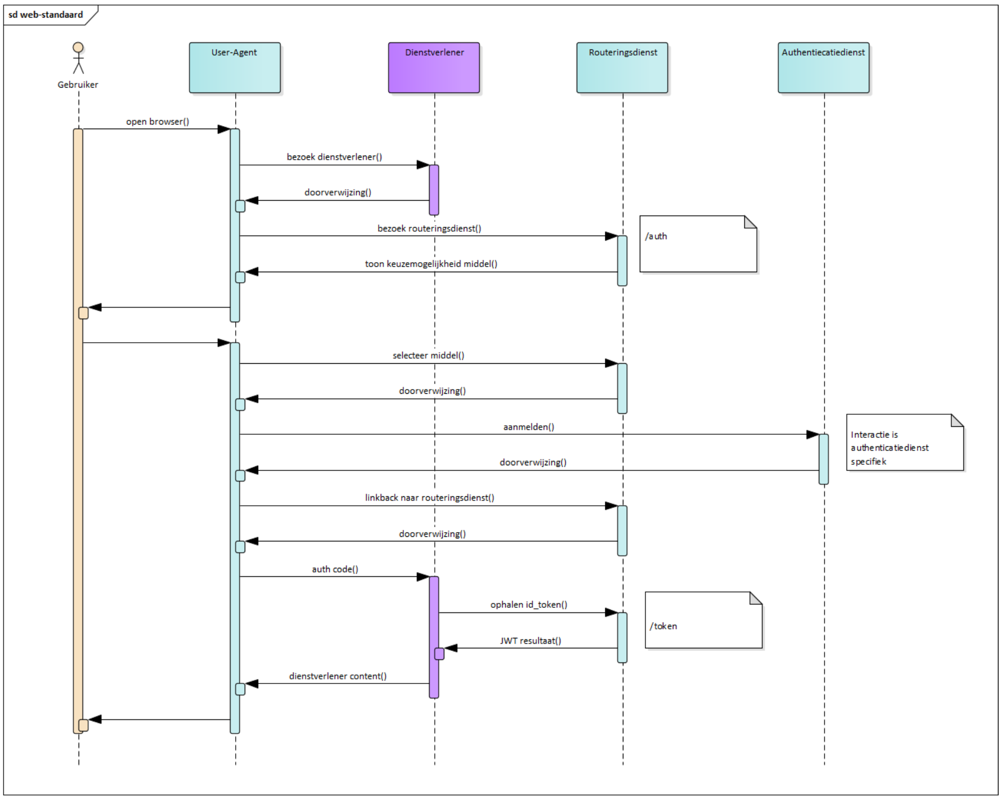
Vervolgens wordt de gebruiker naar de betreffende authenticatiedienst verwezen waar deze zich kan authenticeren. Na succesvolle authenticatie zorgt de authenticatiedienst ervoor dat de burger weer terugverwezen wordt naar de routeringsvoorziening welke de authenticatieverklaring in het vigerende koppelvlak opneemt en aan de dienstverlener retourneert.


Business Process authentication for single service
Gebruiker
Dienstverlener
Routeringsvoorziening
Authenticatiedienst
Wil gebruik maken van dienst
Dienst geleverd
is gebruiker reeds gauthenticeerd ?
Ja
Lever dienst
Toon pagina inloggen mislukt
Toon pagina technisch probleem
Selecteer middel
Formuleer antwoord aan dienstverlener
Authenticeer gebruiker

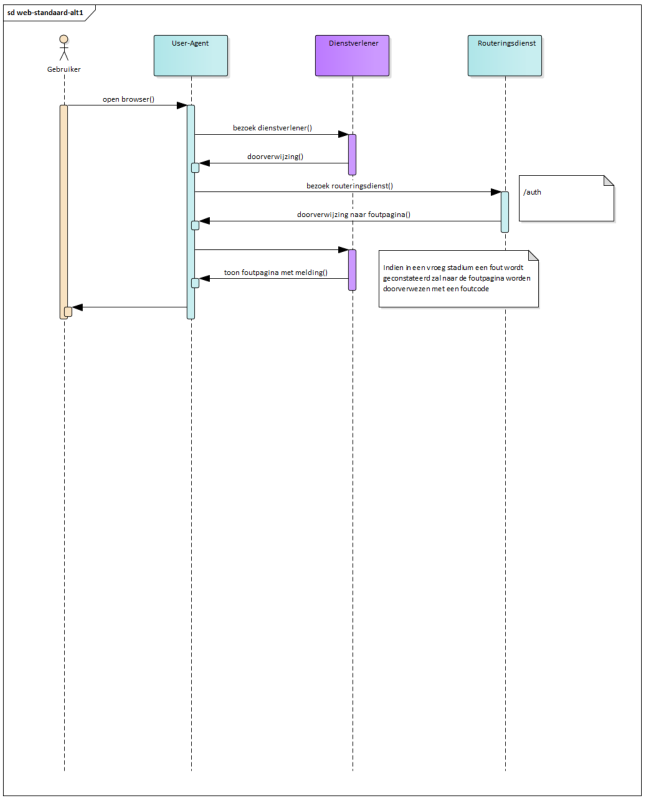
Koppelvlakbeschrijving

Er bestaan twee plaatsen in de gehele authenticatie sequence waar de dienstverlener technisch betrokken is. Daar waar de gebruiker bij het bezoeken van de dienst wordt doorverwezen voor authenticatie richting de routeringsvoorziening, en daar waar de gebruiker na authenticatie wordt terug verwezen naar de dienstverlener. Deze zal op dat moment via de back-channel (het *token-endpoint* bij openID Connect, of de *ResolveArtifact* functie bij SAML) de routeringsvoorziening uitvragen om de daadwerkelijke verklaringen te kunnen verwerken. In onderstaande diagram zijn de technische raakvlakken met dienstverlener in het paars weegegeven.

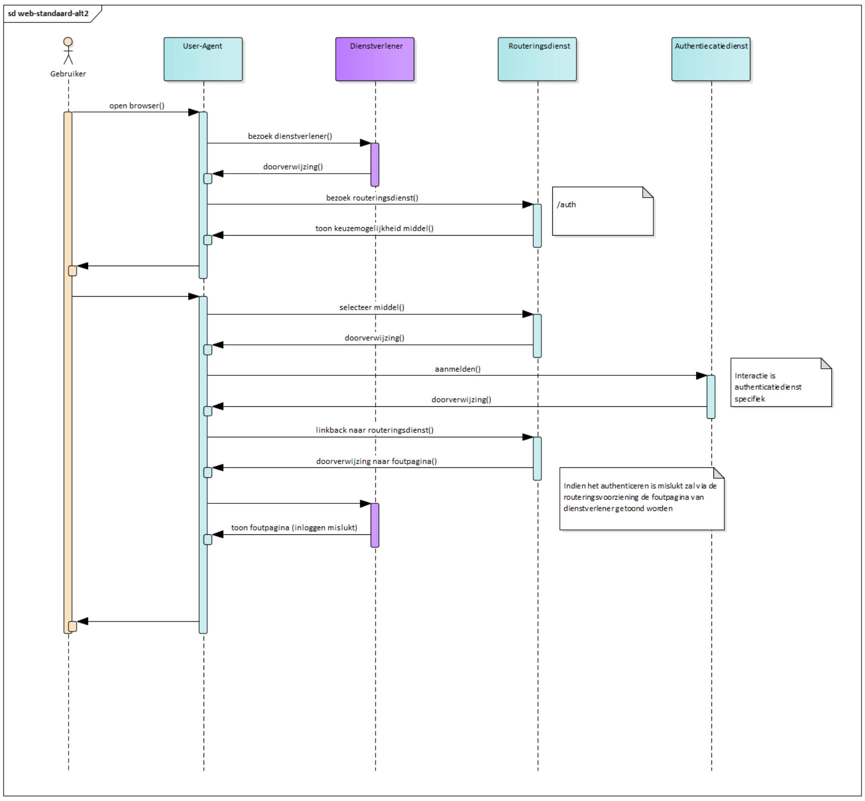
Daarnaast zijn er nog de alternatieve scenario's waarbij het authenticeren door authenticatiedienst is mislukt of door technische problemen de keten niet voltooid kan worden. In die gevallen zal, waar mogelijk, de metadata geraadpleegd worden om de dienstverlenerpagina's die hiervoor zijn aangewezen te tonen.



Alternatief scenario 1 (Technisch probleem)


sd web-standaard-alt1
Gebruiker
User-Agent
Dienstverlener
Routeringsdienst
open browser()
bezoek dienstverlener()
doorverwijzing()
bezoek routeringsdienst()
/auth
doorverwijzing naar foutpagina()
toon foutpagina met melding()
Indien in een vroeg stadium een fout wordt geconstateerd zal naar de foutpagina worden doorverwezen met een foutcode

Alternatief scenario 2 (Inloggen mislukt)


sd web-standaard-alt2
Gebruiker
User-Agent
Dienstverlener
Routeringsdienst
Authentiecatiedienst
open browser()
bezoek dienstverlener()
doorverwijzing()
bezoek routeringsdienst()
/auth
toon keuzemogelijkheid middel()
selecteer middel()
doorverwijzing()
aanmelden()
Interactie is authenticatiedienst specifiek
doorverwijzing()
linkback naar routeringsdienst()
doorverwijzing naar foutpagina()
Indien het authenticeren is mislukt zal via de routeringsvoorziening de foutpagina van dienstverlener getoond worden
toon foutpagina (inloggen mislukt)

Authentication for service set (AUC03)

Achtergrond

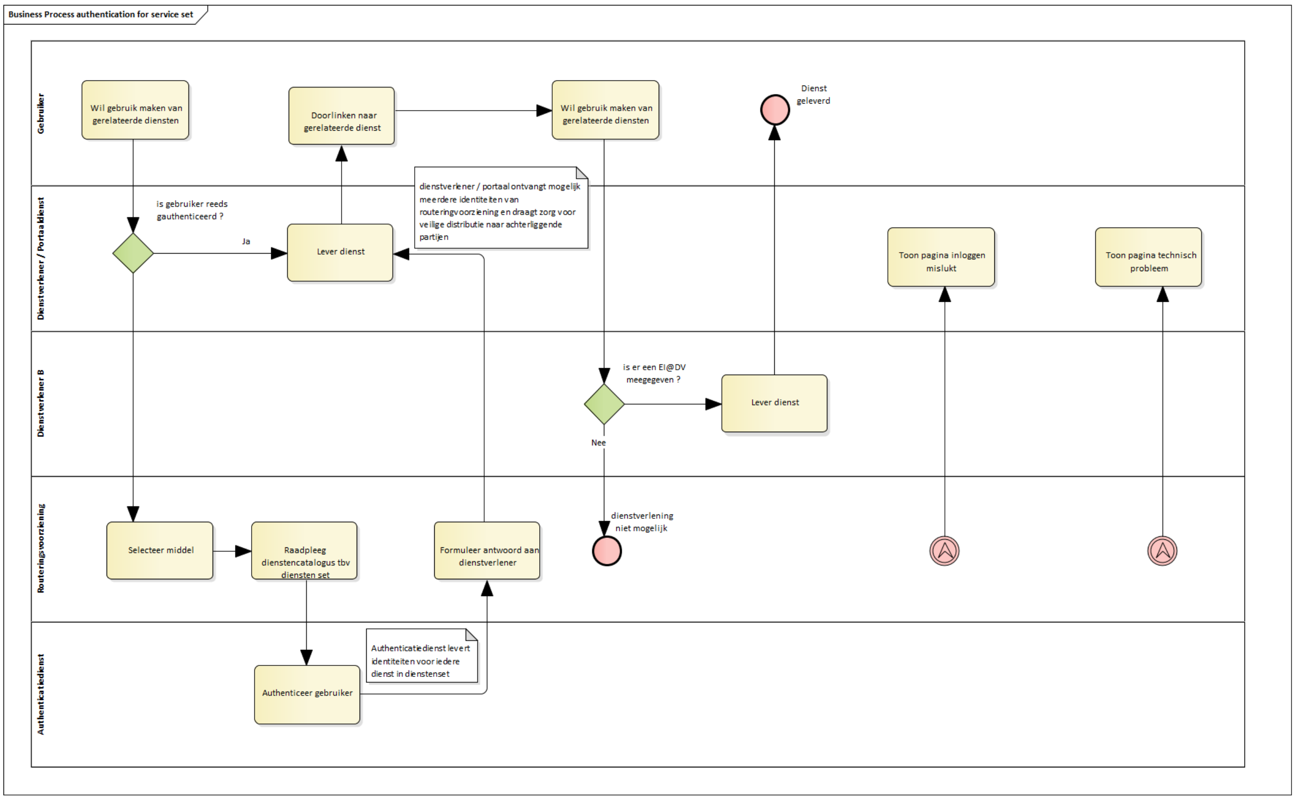
Deze use case geldt voor de situaties waar meerdere dienstverleners een samenhangende dienst leveren. Denk hierbij aan een portaalvoorziening waarin bijvoorbeeld fragmenten uit verschillende bronnen in samenhang getoond worden. Niet alle portaal onderdelen (dienstverleners en mogelijk private partijen) hebben dezelfde gegevens nodig over de gebruiker. Deze krijgen de gebruikergegevens aangeleverd die noodzakelijk zijn voor hun specifieke deel-dienstverlening en de authenticatieverklaring zodat zij zeker kunnen zijn dat de gebruiker succesvol is geauthenticeerd.

Actoren

Actor	Rolbeschrijving
Gebruiker	Wenst gebruik te maken van dienstverlening
Dienstverlener/Portaal	(Semi-) overheidsdienstverlener welke dienst levert aan geauthenticeerde burgers en binnen het kader van een samenghangende dienstverlening als hoofd-dienstverlener aangemerkt kan worden.
Dienstverlener B	(Semi-) overheidsdienstverlener welke dienst levert aan geauthenticeerde burgers
Routeringsvoorziening	Voorziening welke de dienstverlener in staat stelt aan te sluiten op meerdere authenticatiediensten
Authenticatiedienst	Publieke of Private partij welke authenticatie-verklaringen afgeeft

Functionele beschrijving

De gebruiker bezoekt de hoofdvoorziening/portaal van de samenhangende dienst en geeft aan zich te willen authenticeren. Vervolgens zal de dienstverlener de gebruiker naar de routeringsvoorziening doorsturen met het identificerende kenmerk voor de samenhangende dienst. Aldaar wordt de gebruiker de keuze geboden om een authenticatiemiddel te selecteren. Nadat de selectie is gemaakt wordt de gebruiker doorverwezen naar de authenticatiedienst samen met de identificerende kenmerken van de afzonderlijke deel-dienstverleners. Voor iedere aangeduidde dienstverlener wordt een identiteit afgegeven door de authenticatiedienst en via de routeringsvoorziening weer aan de dienstverlener teruggeleverd. Vervolgens is het aan de dienstverlener om zorg te dragen voor de distributie van overige identiteiten aan de andere aangesloten dienstverleners.


Business Process authentication for service set
Gebruiker
Dienstverlener / Portaaldienst
Dienstverlener B
Routeringsvoorziening
Authenticatiedienst
Wil gebruik maken van gerelateerde diensten
Doorlinken naar gerelateerde dienst
Wil gebruik maken van gerelateerde diensten
Dienst geleverd
is gebruiker reeds gauthenticeerd ?
Ja
Lever dienst
dienstverlener / portaal ontvangt mogelijk meerdere identiteiten van routeringvoorziening en draagt zorg voor veilige distributie naar achterliggende partijen
Toon pagina inloggen mislukt
Toon pagina technisch probleem
is er een EI@DV meegegeven ?
Lever dienst
Nee
dienstverlening niet mogelijk
Selecteer middel
Raadpleeg dienstencatalogus tbv diensten set
Formuleer antwoord aan dienstverlener
Authenticatiedienst levert identiteiten voor iedere dienst in dienstenset
Authenticeer gebruiker

…

Authentication with attributes (AUC04)

Achtergrond

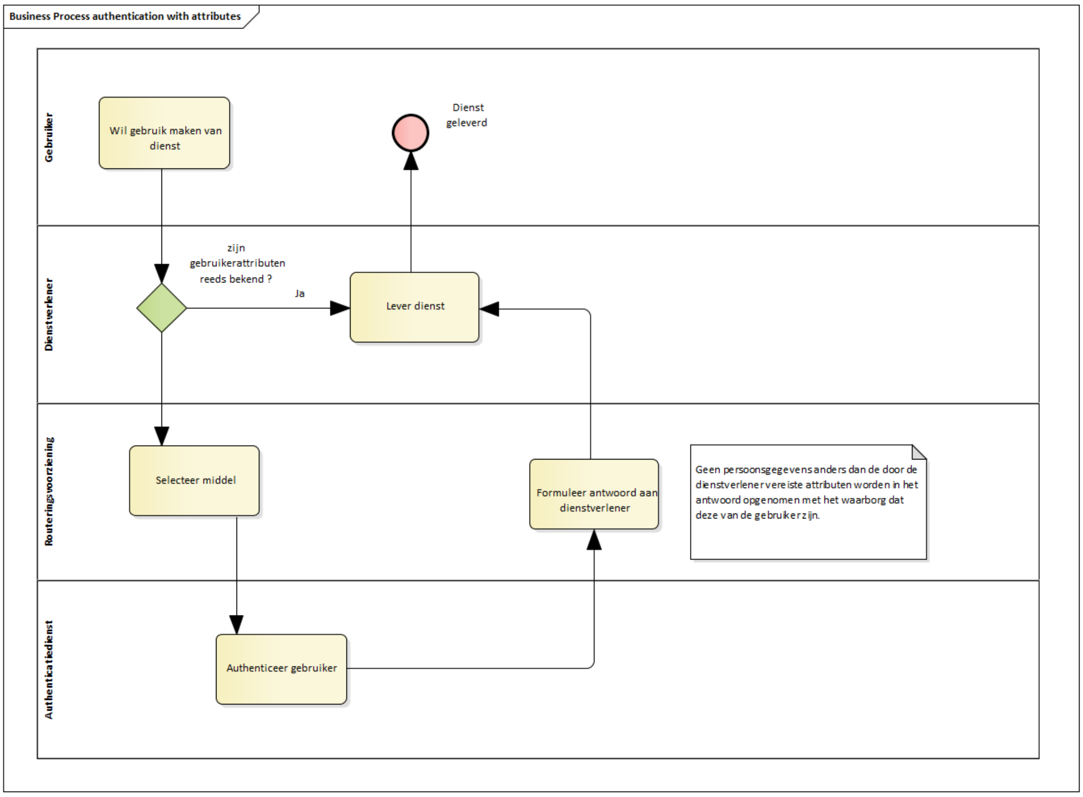
Deze use case is een variatie op de use case *authentication for single service*. In dit geval heeft de dienstverlener geen noodzaak om het BSN te verwerken. Deze is slechts geinteresseerd in een of meerdere attributen die met zekerheid een facet van de gebruiker weerspiegelen. Hierbij kan gedacht worden aan het vaststellen van iemands leeftijd (18+ ?) of ....

Actoren

Gebruiker	Wenst gebruik te maken van dienstverlening
Dienstverlener	(Semi-) overheidsdienstverlener welke dienst levert aan geauthenticeerde burgers
Routeringsvoorziening	Voorziening welke de dienstverlener in staat stelt aan te sluiten op meerdere authenticatiediensten
Authenticatiedienst	Publieke of Private partij welke authenticatie-verklaringen afgeeft

Functionele beschrijving

Voor deze use case geldt de functionele beschrijving uit de use case *authentication for single service*. In deze specifieke use case wordt echter niet de identiteit van de gebruiker geleverd door de authenticatie dienst (na het succesvol authenticeren) maar slechts die set van attributen die door de dienstverlener vereist worden.


Business Process authentication with attributes
Gebruiker
Wil gebruik maken van dienst
Dienst geleverd
Dienstverlener
zijn gebruikerattributen reeds bekend ?
Ja
Lever dienst
Routeringsvoorziening
Selecteer middel
Formuleer antwoord aan dienstverlener
Geen persoonsgegevens anders dan de door de dienstverlener vereiste attributen worden in het antwoord opgenomen met het waarborg dat deze van de gebruiker zijn.
Authenticatiedienst
Authenticeer gebruiker

Koppelvlakbeschrijving

Authentication with IDP preselection (AUC01)

Achtergrond

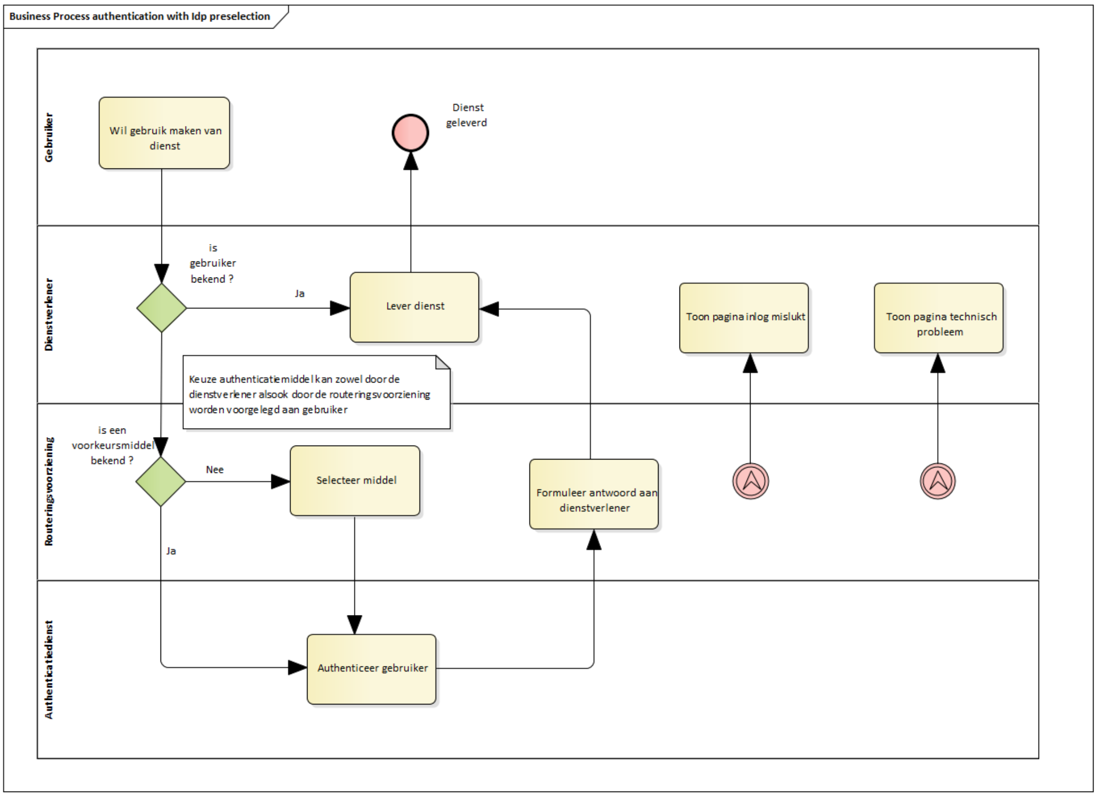
Deze use case is een uitbreding op use case authenticate for single service. In plaats van het kiezen van een authenticatiedienst wanneer de gebruiker bij de routeringsdienst aankomt wordt de keuze voor de gewenste authenticatiedienst reeds eerder in het proces gemaakt. In het kader van een betere gebruikerservaring is het hiermee mogelijk om deze keuze door de dienstverlener te laten voorleggen aan de gebruiker.

Actoren

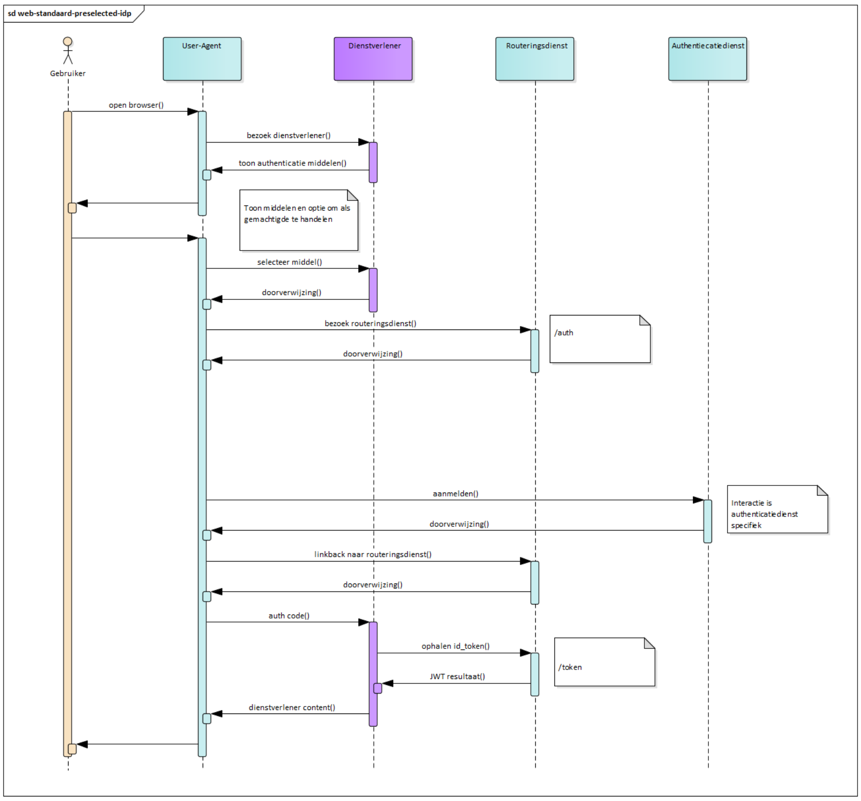
Gebruiker	Wenst gebruik te maken van dienstverlening
Dienstverlener	(Semi-) overheidsdienstverlener welke dienst levert aan geauthenticeerde burgers
Routeringsvoorziening	Voorziening welke de dienstverlener ontzorgt door de gebruiker door te sturen naar de door dienstverlener gespecificeerde (keuze gebruiker) authenticatiedienst
Authenticatiedienst	Publieke of Private partij welke authenticatie-verklaringen afgeeft

Functionele beschrijving

De gebruiker zal bij de dienstverlener kenbaar maken dat deze gebruik wil maken van dienstverlening waarvoor authenticatie vereist is. De dienstverlener zal de gebruiker *voordat* deze doorverwezen wordt naar de routeringsvoorziening de vraag stellen van welke authenticatiedienst hij/zij gebruik wil maken. Tegelijk met de keuze voor de gewenste authenticatiedienst zal de keuze voorgelegd worden aan de gebruiker of deze als gemachtigde wenst te handelen. Nadat een keuze gemaakt is worden de gegevens met betrekking tot keuze voor authenticatiedienst en het al-dan-niet namens een ander handelen doorgegeven aan de routeringsvoorziening. In het geval dat de gebruiker namens een anders wenst te handelen zal hem/haar in een later stadium de keuze geboden worden welke persoon vertegenwoordigd wordt. Eventueel kan door de dienstverlener de keuzemogelijkheid geboden worden aan de gebruiker om via eHerkenning in te loggen en zodoende een burger te vertegenwoordigen.


Business Process authentication with Idp preselection
Gebruiker
Dienstverlener
Routeringsvoorziening
Authenticatiedienst
Wil gebruik maken van dienst
Dienst geleverd
is gebruiker bekend ?
Ja
Lever dienst
Toon pagina inlog mislukt
Toon pagina technisch probleem
Keuze authenticatiemiddel kan zowel door de dienstverlener alsook door de routeringsvoorziening worden voorgelegd aan gebruiker
is een voorkeursmiddel bekend ?
Nee
Selecteer middel
Formuleer antwoord aan dienstverlener
Ja
Authenticeer gebruiker

Koppelvlak beschrijving


sd web-standaard-preselected-idp
Gebruiker
User-Agent
Dienstverlener
Routeringsdienst
Authentiecatiedienst
open browser()
bezoek dienstverlener()
toon authenticatie middelen()
Toon middelen en optie om als gemachtigde te handelen
selecteer middel()
doorverwijzing()
bezoek routeringsdienst()
/auth
doorverwijzing()
aanmelden()
Interactie is authenticatiedienst specifiek
doorverwijzing()
linkback naar routeringsdienst()
doorverwijzing()
auth code()
ophalen id_token()
/token
JWT resultaat()
dienstverlener content()

Authentication with multiple recipients (AUC05)

Achtergrond

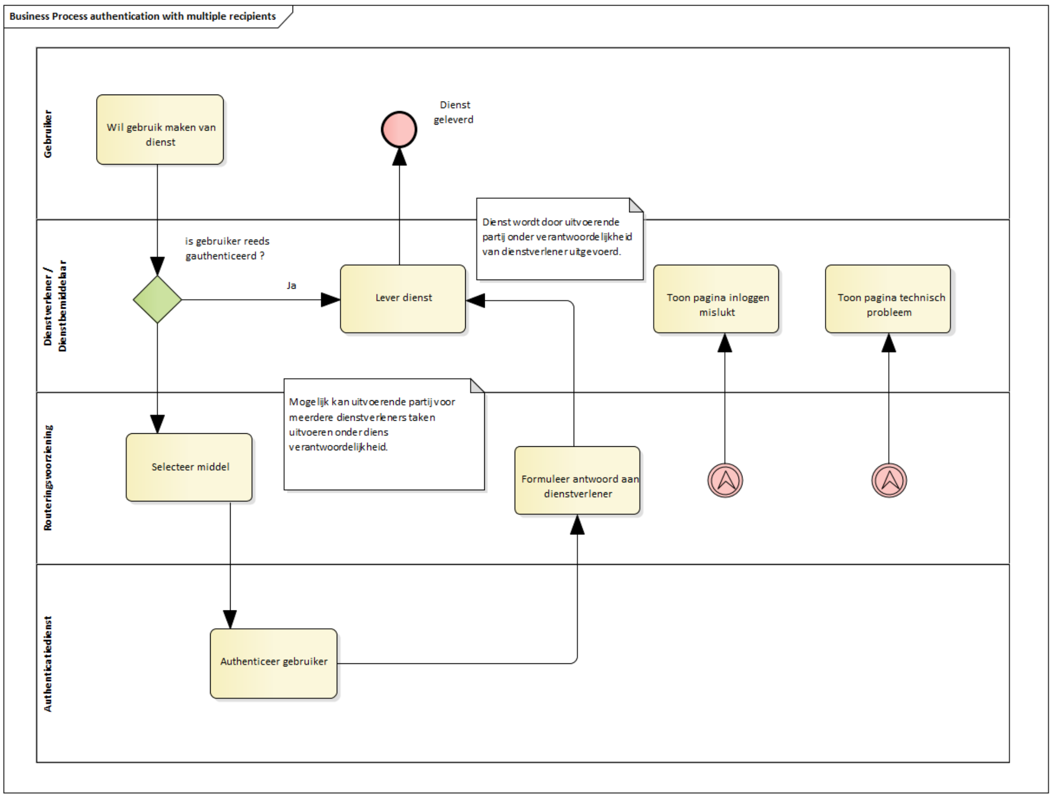
Een burger wil via de webdienst van een (semi-)overheidsdienstverlener een dienst afnemen waarvoor hij/zij zich dient te authenticeren. De dienstverlener wordt vertegenwoordigd door een dienstbemiddelaar.

Actoren

Gebruiker	Wenst gebruik te maken van dienstverlening
Dienstverlener / Dienstbemiddelaar	(Semi-) overheidsdienstverlener welke dienst levert aan geauthenticeerde burgers en het zorg dragen voor authenticatie heeft uitbesteed aan een dienstbemiddelaar
Routeringsvoorziening	Voorziening welke de dienstverlener in staat stelt aan te sluiten op meerdere authenticatiediensten
Authenticatiedienst	Publieke of Private partij welke authenticatie-verklaringen afgeeft

Functionele beschrijving

Deze use case is functioneel gezien gelijk aan de use case authentication for single service met de uitzondering dat de dienstverlener in dit geval gebruik maakt van een dienstbemiddelaar.


Business Process authentication with multiple recipients
Gebruiker
Dienstverlener / Dienstbemiddelaar
Routeringsvoorziening
Authenticatiedienst
Wil gebruik maken van dienst
Dienst geleverd
is gebruiker reeds gauthenticeerd ?
Ja
Lever dienst
Dienst wordt door uitvoerende partij onder verantwoordelijkheid van dienstverlener uitgevoerd.
Toon pagina inloggen mislukt
Toon pagina technisch probleem
Selecteer middel
Mogelijk kan uitvoerende partij voor meerdere dienstverleners taken uitvoeren onder diens verantwoordelijkheid.
Formuleer antwoord aan dienstverlener
Authenticeer gebruiker

Koppelvlakbeschrijving

Authentication with representation (AUC06)

Achtergrond

Deze use case beschrijft de situatie waarbij een belanghebbende door en ander persoon wordt vertegenwoordigd. Het gaat hierbij om vertegenwoordiging *van* een natuurlijk persoon of rechtspersoon *door* een ander natuurlijk persoon of rechtspersoon. Ook aaneengeschakelde ketens van dergelijke vertegenwoordiging vallen hieronder.

Daarnaast wordt in de context van deze specificaties onder vertegenwoordiging alle bestaande, gangbare (juridische) varianten verstaan. Hieronder vallen dus onder andere (niet limitatief): vrijwillige machtiging, wettelijke vertegenwoordiging (curatele, bewindvoering), gevolmachtigd (bestuurder, directeur), (beperkt) tekenbevoegd, etc...

Iedere vertegenwoordigingsrelatie kent een vertegenwoordigde (belanghebbende) en een vertegenwoordiger (gemachtigde). De vertegenwoordiger handelt namens de vertegenwoordigde.

De vertegenwoordiger kan zijn zowel een andere burger zijn alsmede een medewerker/functionaris van een organisatie. In het laatste geval zal de medewerker/functionaris zich middels eHerkenning moeten authenticeren.

Actoren

Gebruiker	Wenst gebruik te maken van dienstverlening, namens de Belanghebbende.
Belanghebbende	Wenst de dienst af te nemen, laat dit uitvoeren door de Gebruiker. Hiervoor is een vertegenwoordigingsrelatie vastgelegd.
Dienstverlener	(Semi-) overheidsdienstverlener welke dienst levert aan geauthenticeerde dienstafnemers (burgers of organisaties).
Routeringsvoorziening	Voorziening welke de Dienstverlener in staat stelt aan te sluiten op meerdere authenticatiediensten
Authenticatiedienst	Publieke of Private partij welke authenticatie-verklaringen afgeeft
Machtigingsregister	Publiek of Privaat machtigingsregister, welke wordt geraadpleegd voor geldende vertegenwoordigingsrelaties.

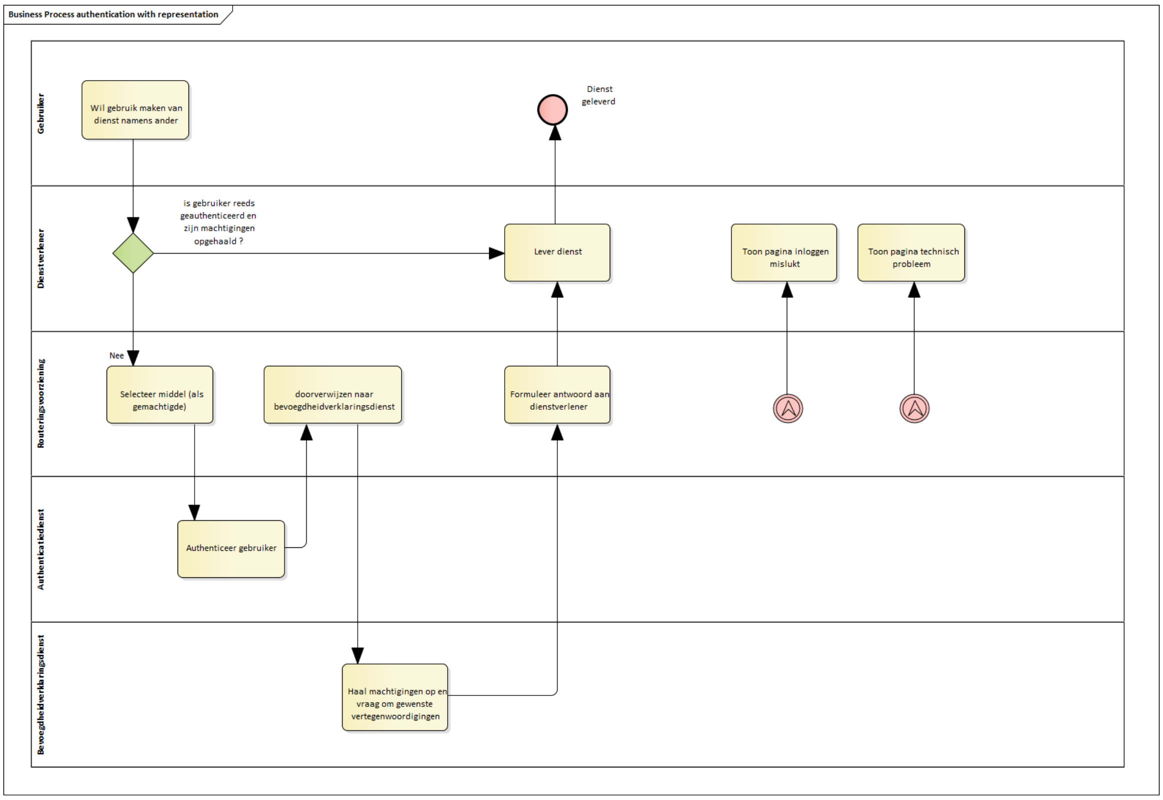
Functionele beschrijving

Meerdere varianten:

- Burger vertegenwoordigt burger
- Functionaris/organisatie vertegenwoordigt burger
- Medewerker/functionaris vertegenwoordigt organisatie
- Medewerker/functionaris vertegenwoordigt functionaris (enkel van toepassing voor beroepen welke gelden op persoonlijk titel)
- Combinaties van bovenstaanden, welke leiden tot een keten van vertegenwoordigingsrelaties.

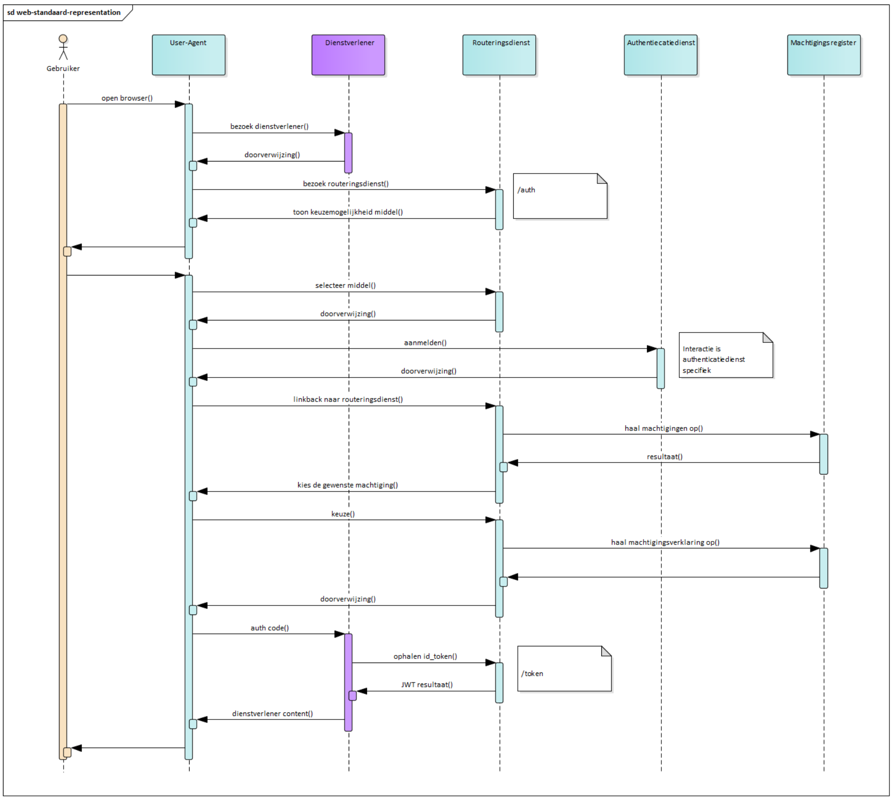
Burger vertegenwoordigt burger:

In dit geval wordt een burger door een andere burger vertegenwoordigd. De gemachtigde zal na het authenticeren zoals beschreven in use case *authentication for single service* kenbaar maken dat hij/zij een ander wil vertegenwoordigen. Deze selectie wordt gemaakt in het webportaal van de machtigingsvoorziening waar de routeringsvoorziening de gebruiker naartoe stuurt nadat deze succesvol is geauthenticeerd. Na een selectie van de vertegenwoordigden gemaakt te hebben kan de gemachtigde burger de door dienstverlener geboden dienst namens de vertegenwoordigde burger(s) gebruiken. Dienstverlener dient rekening te houden met de mogelijkheid dat de geauthenticeerde gebruiker meerdere machtigingen op haar/zijn naam heeft staan.


Business Process authentication with representation
Gebruiker
Wil gebruik maken van dienst namens ander
Dienst geleverd
Dienstverlener
is gebruiker reeds geauthenticeerd en zijn machtigingen opgehaald ?
Lever dienst
Toon pagina inloggen mislukt
Toon pagina technisch probleem
Routeringsvoorziening
Nee
Selecteer middel (als gemachtigde)
doorverwijzen naar bevoegdheidverklaringsdienst
Formuleer antwoord aan dienstverlener
Authenticatiedienst
Authenticeer gebruiker
Bevoegdheidverklaringsdienst
Haal machtigingen op en vraag om gewenste vertegenwoordigingen

TODO uitwerken andere vertegenwoordigingrelaties dan burger-burger.

Koppelvlakbeschrijving


sd web-standaard-representation
Gebruiker
User-Agent
Dienstverlener
Routeringsdienst
Authentiecatiedienst
Machtigingsregister
open browser()
bezoek dienstverlener()
doorverwijzing()
bezoek routeringsdienst()
/auth
toon keuzemogelijkheid middel()
selecteer middel()
doorverwijzing()
aanmelden()
Interactie is authenticatiedienst specifiek
doorverwijzing()
linkback naar routeringsdienst()
haal machtigingen op()
resultaat()
kies de gewenste machtiging()
keuze()
haal machtigingsverklaring op()
doorverwijzing()
auth code()
ophalen id_token()
/token
JWT resultaat()
dienstverlener content()

Datamodel Services

Each Authentication / Authorization Request using this interface effectively is a request of a User to access a Service. This Service is an abstract concept, it can be nearly anything.

A Service as used within this interface is a description of various aspects of where access is requested for. This includes information on the Service Provider, the Level of Assurance requested, a description of the service itself, etc.... Below is a high-level specification of how a Service is defined with these interface specifications.

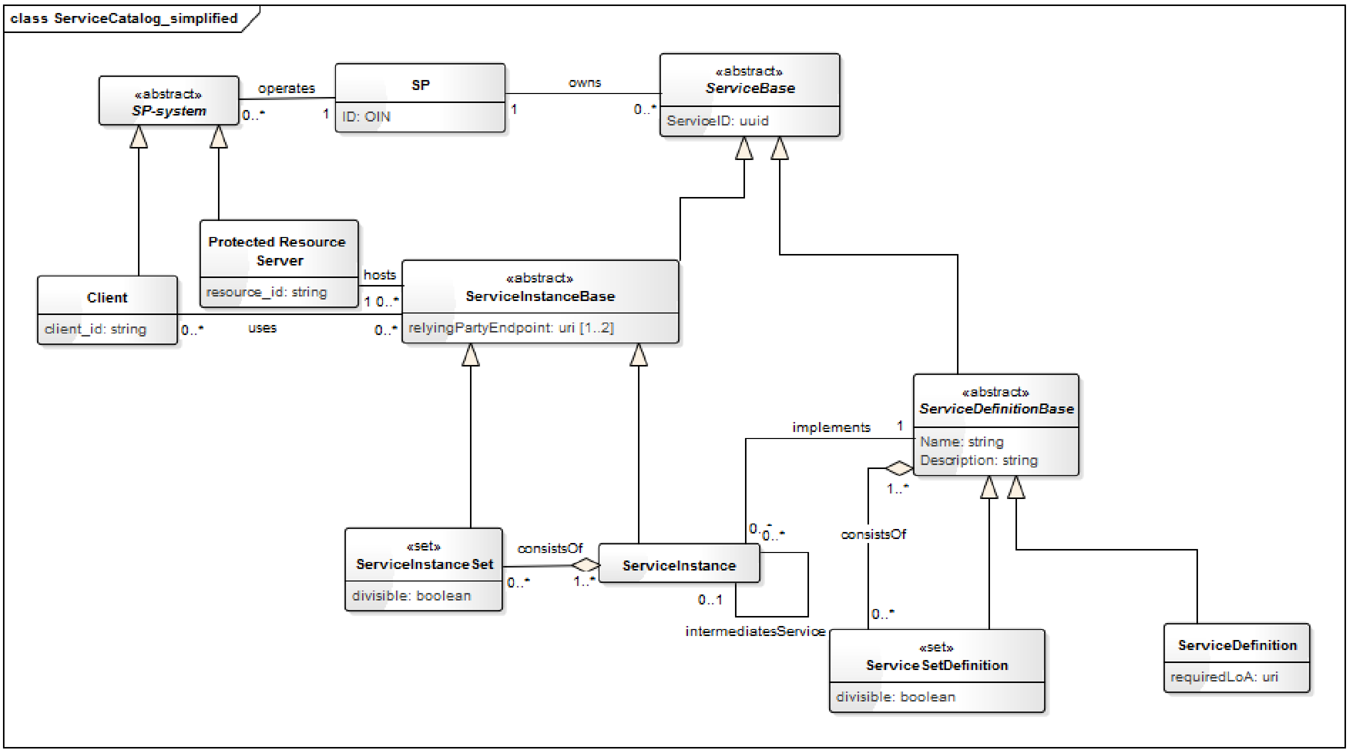
Service types

In order to facilitate various use cases, these specifications uses several appearances of Services. A Service can be one of the following:

- *Service*: any of the 4 types defined below.
- *Service Definition*: a (generic) functional, semantic definition of a clearly delimited Service. For example: "requesting a parking permit".
- *Service Definition Set*: a set of Service Definitions as defined above. For example "transport related municipality services".
- *Service Instance*: an individual, concrete, (specific) technical realization of a Service Definition or Service Definition Set. For example "requesting a parking permit in The Hague".
- *Service Instance Set*: a set of Service Instances as defined above. For example "all services at the municipality of The Hague".

This model allows reuse of often used Services, allows for prior registration of (delegation) authorizations. It provides a balanced, layered approach for dealing with fine-grained control where applicable, yet at the same time practically manageable and usable structures.

Data Model Services



Each Service, Service Definition, Service Definition Set, Service Instance or Service Instance Set is assigned a UUID, as ServiceID. The UUID uniquely identifies the specific service description. Through the UUID, the service and all of its details can be referenced in a globally and temporally unique manner. The temporal uniqueness provides for a way to reference the exact description in a historical way.

A more detailed data model for the Service Catalog can be found as data model Service Catalog.

TODO add description detail-level diagram

Important concepts

The data model has a few concepts that are used throughout these interface specifications and are therefore quite important for understanding. These are explained below.

- *ServiceID*: Each service (definition, instance or a set) is assigned an ID. The ID is of the type UUID, making it both globally and temporal unique and identifying a particular service.
- *ServiceMnemonicURL*: A Service Instance or Service Instance Set MAY have a mnemonic URI. This allows to use "readable" human URLs to reference a specific Service Instance (Set).

TODO eleborate on other important concepts

Both Set concepts are recursive. Hierarchical constructs can be created by referencing Sets in a Set. Creating loops is not allowed and restrictions on the depth of service hierarchies and recursions may be imposed to limit complexity and for practical reasons.

A Service Instance may intermediate another Service Instance, but only if the referenced Service Instance does not reference another Service Instance (in other words Service intermediation is non-recursive).

Usage of datamodel Services

Usage of (reference to) the Service(s) concepts

A Relying Party is requesting authentication/authorization for a specific purpose, defined through the Service. This definition of a Service is used for multiple reasons:

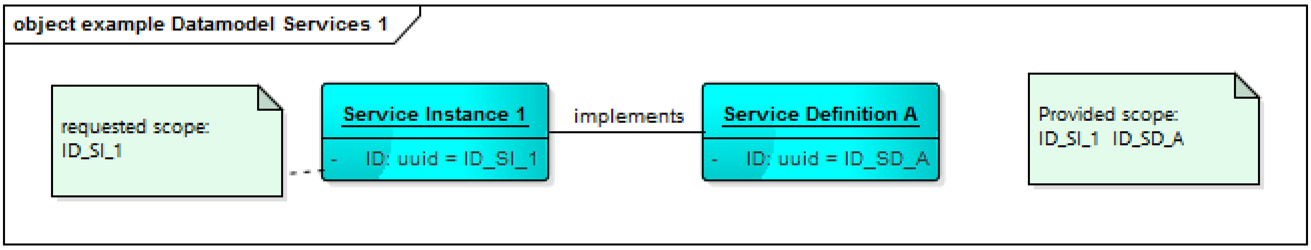
- Used to reference a Service in a request in a way fitting within the OAuth2/OIDC standards and efficiently referencing common data.
- Ask a User for consent; the consent is documented in the service description and sets the semantics.
- Enable registration of specific representation relationships, prior to actual utilizing such registered authorization.
- Derive technical details necessary for handling a specific request.
- Allow flexibility through forming sets of services, facilitating both fine-grained control as well as practical usage.

Usage as reference for Authentication / Authorization Request

In a OIDC Authentication / Authorization Request, one or more Service Instance(s) or Service Instance Set(s) must be referenced to define the access requested with the request. The Service Instance (Set) defines what is requested, both functionally as well as technically.

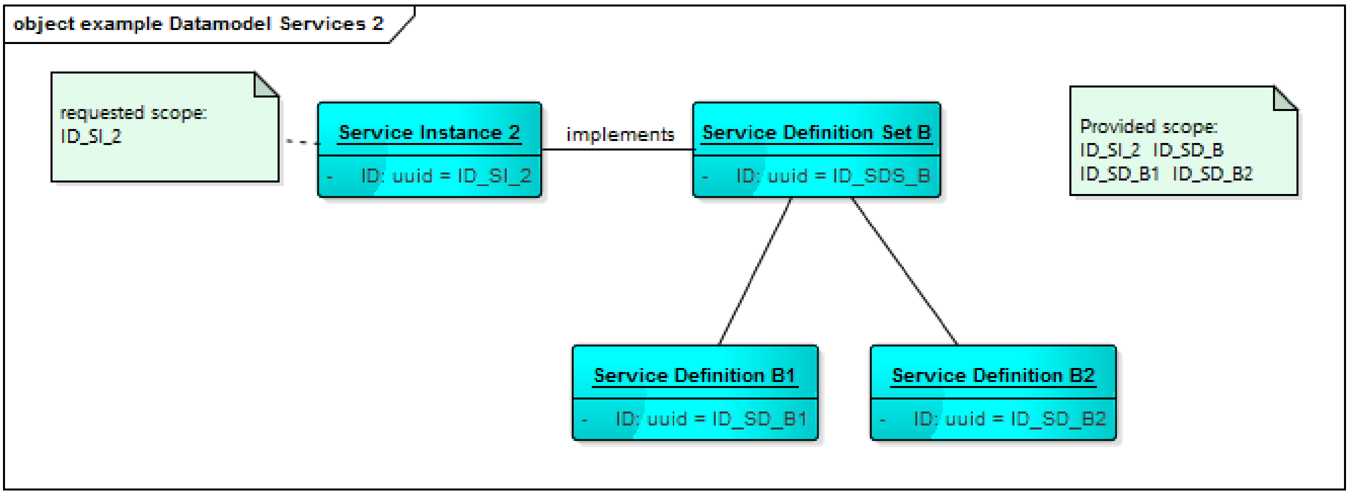
The functionality requested is defined in the Service Definition (Set), referenced by the Service Instance. The other information included in a Service Instance is used for technical reasons.

Service Instance



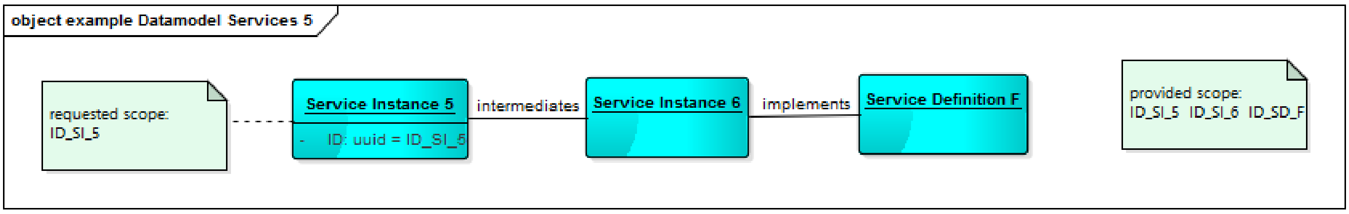
In case a Service Instance references a Service Definition, functionally a single Service is requested. The semantics of the access request are defined by the Service Definition.

Service Set



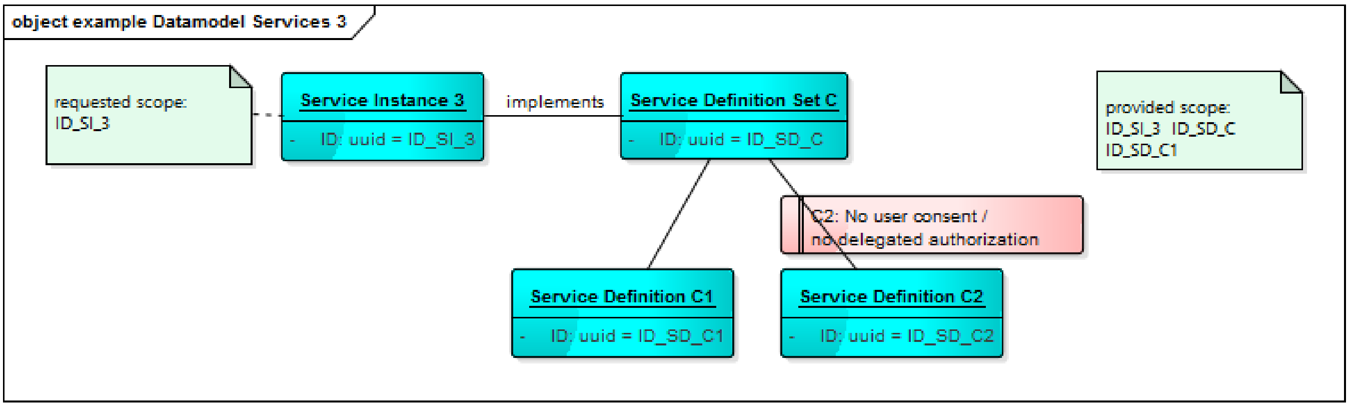
In case a Service Instance references a Service Definition Set in the request, a combination of multiple Services is requested. The access request effectively consists of the combined semantics of each of the Service Definitions in the reference Service Definition Set.

Service Intermediation



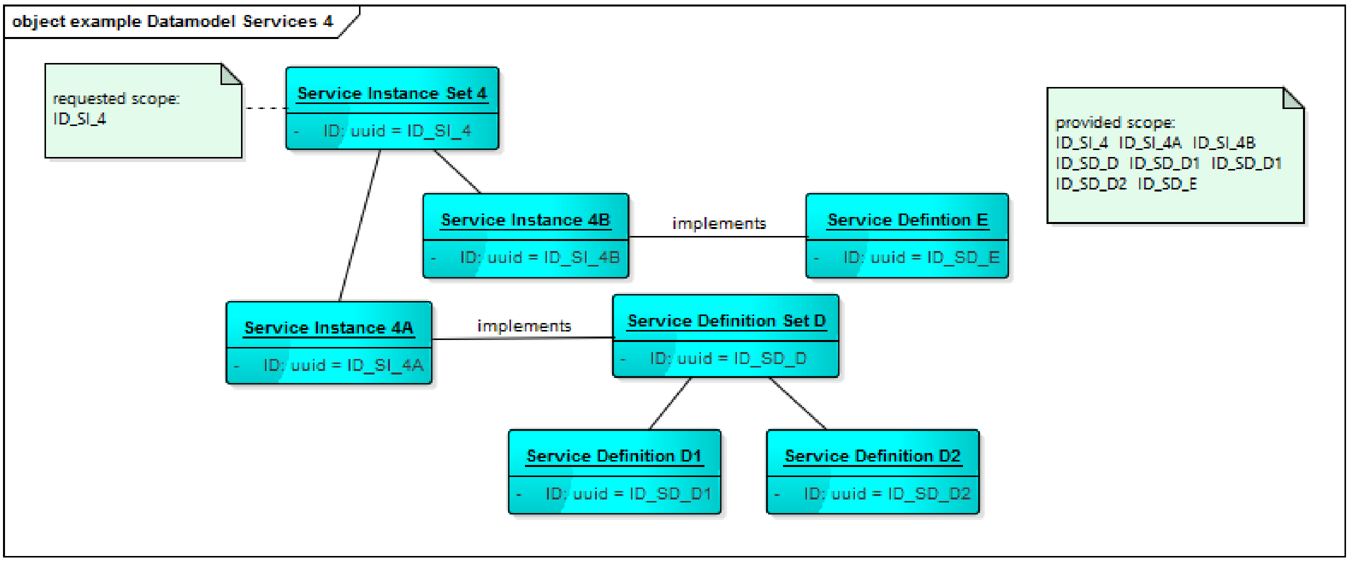
In case a Service Instance references another Service Instance, this identifies a case of "dienstbemiddeling"; the first service provider intermediates the service instance of the , using the semantics

Service Set – divisible



A Service Definition Set can be marked as "divisible". If a set *is not* "divisible", a User must consent with each Service included or have a registered authorization for each Service. If a set *is* "divisible", a User may consent to only a subset of the requested set or have a registered authorization for a subset of the requested set.

Service Instance Set



In case a Service Instance Set is requested, this is equal to a request for multiple Service Instances and as a result functionally referencing multiple Service Definitions. As with Service Definition Sets can be marked "divisible" with the same semantics. The differences between a Service Definition Set and Service Instance Set are:

- Ownership: A Service Instance Set can be defined by any Service Provider (DV). A Service Definition Set can be restricted, so the Service Provider owning a Service Definition (Set) can control who may include the Service Definition (Set) into a Service Definition Set.
- Representation relationship: Service Definition Sets can be used to register authorizations (delegation, mandate, other type of legal form of representation). Service Instance(Set)s cannot be used to register authorizations.

Provided scope

Both an OIDC Token Response and an OIDC Access Token contain the scope of the access granted. The scope returned describes the actual scope for which access is granted. All identifiers (UUIDs) for each Service Instance, Service Definition or Set of these directly or indirectly referenced from the request – provided consent and authorization are granted – will be included in the response.

In case a set is marked as "divisible", the scope returned can be a subset due to only partial consent or lack of registered authorization(s). A set marked as "divisible" thus can be returned without certain Services listed in the applicable scope. As a consequence, a Service Provider requesting a "divisible" Service (Set) MUST check the returned scopes and take scopes (Services) *not* provided into account in its access decision.

Note: a Service Instance can be identified through the combination of Service Definition and Audience (aud as defined in OAuth2 and these specifications. Combinations of Service IDs from a scope combined with the audience may result in a broader reach than intended. As a result, the actual service referenced by the scope is preferred and deriving the granted access from the scope and the audience must be treated with caution.

OIDC specs

This section contains the (technical) specifications of the interface between a Service Provider (SP, or *Dienstverlener,* DV) and the OpenID Provider (a.k.a. Routeringsdienst, RD).

OIDC Introduction

The interface between a *DV* and the RD, is based on OpenID Connect. OpenID Connect is an identity layer on top of OAuth2.

OpenID Connect is specified by the OpenID Foundation. OAuth2 is specified by IETF.

This interface specification is based on the OpenID Connect and OAuth2 specifications.

Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Terminology

Terminology specific to these specifications:

- RV = *Routeringsvoorziening*. A central service, orchestrating federated identification, authentication and authorization for supported Service Providers.
- RD = *Routeringsdienst*. One of the technical implementations for the RV. Also referenced as OpenID Provider (OP).
- DV = *Dienstverlener*. Service Provider or SP.
- AD = *Authenticatiedienst*. Identity Provider or IDP.
- MR = *Machtigingenregister*. (Delegated) Authorization Information Provider.
- TODO

For other terminology, see OAuth2 and OpenID Connect specifications.

OIDC Common requirements

This part contains requirements on anyone implementing these interface specifications.

OIDC Generic security requirements

All parties:

- MUST ensure their private key and/or any `client_secret` and other symmetric/secret keys are securely stored, used and managed as per best practices.
- MUST ensure only cryptographically sound random is used for random values and during key generation.
    - For any value that has to be random (non-guessable) that value MUST be generated with at least 128-bit entropy, as in https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.1.
- MUST verify (PKIoverheid) certificates according to best practices (validate hierarchy, CRL/OSCP, time, key usage, etc...) before usage.
- MUST ensure the session with the user-agent of the subject is securely managed as per best practices (e.g. https://www.owasp.org/index.php/Session_Management_Cheat_Sheet).
- MUST use NTP synchronized clocks.

OIDC Transport interoperability and security requirements

DNSSEC requirements

These specifications utilize DNSSEC (https://tools.ietf.org/html/rfc4033) as one of the security measures. For DNSSEC the following applies.

Anyone:

- SHOULD verify DNSSEC records when communicating with peers.

A (requesting) DV-system / Client:

- SHOULD publish DNSSEC records for their domains registered and used in any eID(-related) context.

A (responding) RD / OP:

- MUST publish DNSSEC records for their domains registered for use within any eID context.

HTTPS and TLS requirements

All communication between peers in these specifications is based on HTTP. All communication MUST be secured using Transport Layer Security, TLS. As a result, all communication MUST be transported over HTTPS (https://tools.ietf.org/html/rfc2818).

HTTPS and TLS MUST be applied as specified in BSNk Technical Specifications, release R3 or above. See Secure connection (TLS) (BSNk Technical Specifications). The following (additional) requirements apply:

- Connections between servers (a.k.a. back-channel connections):
    - MUST only apply TLS version 1.2 (https://tools.ietf.org/html/rfc5246) or above;
    - MUST only use ciphers listed as 'good';
    - MUST use other options as indicated on Secure connection (TLS).
- Connections between a user-agent / end user device and the DV-system / Client, RD / OP and other related components:
    - MUST apply TLS version 1.0 (https://tools.ietf.org/html/rfc2246) or above;
        - SHOULD apply only TLS 1.2 or above, as implementations not supporting TLS 1.2 are outdated and can be expected to be insecure;
    - MUST only apply ciphers listed as either 'adequate' or 'good' and SHOULD only apply ciphers listed as 'good';
    - MUST use other options as indicated on Secure connection (TLS).

As HTTP itself is stateless, implementations are free to choose a method of maintaining state or sessions with a User-agent when applicable. The following applies for any HTTP state/session mechanism:

- HTTP servers MUST ensure session and state information is secured and User-agents are properly instructed with relevant security settings (e.g. proper cookie lifetime, Secure setting for cookies, CORS headers and similar).
- Any HTTP session or state tracking mechanisms MUST be implemented using current best practices to avoid session hijacking and other attacks. For more information, see for instance https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md.

HTTP Status codes for redirects

These specifications are built on top of OAuth2 and OpenID Connect. Both underlying protocols rely heavily on HTTP redirects.

As per OAuth2 (https://tools.ietf.org/html/rfc6749#section-1.7), any method to accomplish a redirect MAY be used. However, implementations:

- MUST apply proper `Pragma` and `Cache-control` HTTP headers, as per OIDC (https://openid.net/specs/openid-connect-core-1_0.html#TokenResponse) and OAuth2 (https://tools.ietf.org/html/rfc6749#section-5.1).
- MUST NOT use a HTTP 307 redirect if any user specific information, such as credentials, or privacy sensitive information is included in the request being redirected and SHOULD use a HTTP 303 redirect, as per https://tools.ietf.org/html/draft-ietf-oauth-security-topics-10#section-3.10.
- SHOULD avoid using methods that result in sensitive information in the URL that can end up residing in browser history, on proxy servers, in log files or HTTP Refer(r)er headers, etc....
- SHOULD avoid relying exclusively on support for JavaScript in the user-agent to accomplish redirects, as JavaScript might be disabled which would result in user experience / usability issues.

Common methods of implementing a HTTP redirect include:

- Sending a HTTP 302 (https://tools.ietf.org/html/rfc7231#section-6.4.3) or HTTP 303 (https://tools.ietf.org/html/rfc7231#section-6.4.4) response including a `Location` header containing the destination URL including the parameters.
- Using an html form and JavaScript or a button to trigger a POST request to the destination. See https://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html#FormPostResponseExample for an example of this method.

OIDC Message and protocol security requirements

Anyone:

- MUST validate any received message before usage.

A (requesting) DV-system / Client:

- MUST have pre-registered its service(s) with the RV / RD / OP.
- MUST initiate and maintain a secure session with the user-agent of the subject to be identified/authenticated /authorized (user).
- MUST enable a user to "log off" at any time, terminating the session with the user-agent of the subject (user). This is a form of "local" log off, affecting the session(s) between the user-agent and the DV-system / Client.
- MUST register any `redirect_uri` used and MUST ensure these cannot be (ab)used for "open redirect" requests.
    - SHOULD NOT have multiple `redirect_uri`s on multiple domains (as per https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.2.1).
- MUST NOT accept unsolicited login initiation; that is the DV-system / Client MUST initiate the protocol itself, triggered by interaction from a user-agent of an not-yet-known User.
- MUST only interact and accept protocol messages from known and trusted RD / OPs.
- TODO
- ...

A (responding) RD / OP:

- MUST NOT register or accept `redirect_uri` using non-absolute URIs and MUST NOT apply pattern matching when verifying any `redirect_uri` but use exact URI matching instead, as per https://tools.ietf.org/html/draft-ietf-oauth-security-topics-07#section-3.1.3 and https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.1.8.
- MUST enable a user to "cancel" and return to the requesting DV at any time.
- TODO
- ...

A Resource Server:

- MUST apply appropriate security and privacy controls to any information received originating from an RD / OP.
- MUST only accept (access) tokens from known and trusted RD / OPs, as per https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.4.3.
- SHOULD only accept requests from known and trusted DV-system / Clients.
- TODO
- ...

OIDC Client Authentication

The Client, the DV's system, MUST be able to authenticate. Client authentication is applied upon sending requests to the OpendID Provider's direct endpoints, the Token Endpoint and the Introspection Endpoint.

Where Client Authentication is applicable, one of the following methods MUST be used, in order of preference:

1. Mutual authenticated TLS, using a client TLS certificate matching the registered subject (see OIDC Client registration Request, `tls_client_auth_subject_dn`).
2. `private_key_jwt` : signed JWT, as defined at http://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication.

HTTP Basic (`client_secret_basic`), HTTP Post (`client_secret_post`) and Client secret JWT (`client_secret_jwt`) authentication methods MUST NOT be used.

See https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication (for `private_key_jwt`) and https://tools.ietf.org/html/draft-ietf-oauth-mtls-08#section-2 (for Mutual TLS).

A combination of the above methods MAY be used. Practically, this means mutual authenticated TLS in combination with a private key JWT.

In case of a "clusteraansluiting":

See the Use case guide for more information on "clusteraansluiting".

⚠ Note: authentication of a "cluster" implementation is preliminary and may be subject to change in future updates of these specifications!

Authentication Request

The authentication request MUST be made using a signed request object (signed by the PKIoverheid certificate of the service provider, Dienstverlener) which contains the following REQUIRED claim that will be used to identify the cluster system and verify the relationship between the cluster system and the service provider:

cluster_assertion : This is a nested, signed JWT object (JWS), signed by the cluster system's PKIoverheid certificate and MUST contain the parameters as listed below under Private key JWT.

Request objects SHOULD be passed by value as described at http://openid.net/specs/openid-connect-core-1_0.html#RequestObject and MAY be passed by reference, using the `request_uri` parameter instead of the `request` parameter, as described under OIDC Authentication / Authorization interaction.

Token Request (and any other requests that require oauth client authentication)

Requests to the Token Endpoint by the cluster system MUST apply mutual authenticated TLS using the cluster system's PKIoverheid certificate combined with authentication using `private_key_jwt` client authentication method where the client assertion is signed using the PKIoverheid certificate of the requesting Dienstverlener.

Authentication methods

Mutual authenticated TLS

A client can authenticate using `tls_client_auth`, by applying client authentication during a TLS handshake so that mutual authentication is established. The recipient can then verify the client's certificate matches the registered expected identifier (`subjectDN`), this allows for certificate renewal without the need for explicit key management.

The `subjectDN` of the certificate used by the client MUST be pre-registered (see OIDC Client registration Request). See https://tools.ietf.org/html/draft-ietf-oauth-mtls-12#section-2.1.

Private key JWT

A client can authenticate using `private_key_jwt`, by generating a JWT with contents as specified below and signing it with its private key. The private key will correspond to the client's PKI certificate. This method is specified in OIDC, see https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication.

The contents of the JWT is as follows:

parameter	0..n	description	comment	example	documentation references
iss	1	REQUIRED. Issuer. This MUST contain the client_id of the Client or the identifier of the cluster in case this private key JWT is used as "cluster_assertion".			https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.2
sub	1	REQUIRED. Subject. This MUST contain the client_id of the Client or the identifier of the cluster in case this private key JWT is used as "cluster_assertion".			https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.2
aud	1	REQUIRED. Audience.	The Audience of the client authentication is the RD / OP.		https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.2
jti	1	REQUIRED. JWT ID. A unique identifier for the token, which can be used to prevent reuse of the token. MUST have enough entropy to guarantee JWTs are unique for at least 12 months.			https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.2
exp	1	REQUIRED. Expiration time on or after which the JWT MUST NOT be accepted for processing.	This `private_key_jwt` MUST NOT have a lifetime exceeding *<client-authentication-jwt.max-lifetime>*		https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication

					https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.2
iat	0..1	REQUIRED. Time at which the JWT was issued.	Although optional in OIDC, required in the iGov profile.		https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.2

The following is a non-normative example of usage of the `private_key_jwt` client authentication method in a OIDC Token Request (only the http request body). Please note that the `client_assertion` parameter contains the signed JWT and `client_assertion_type` parameter is equal to `urn:ietf:params:oauth:client-assertion-type:jwt-bearer` (encoded using application/x-www-form-urlencoded) as specified in https://tools.ietf.org/html/rfc7522#section-2.2 :

```
grant_type=authorization_code&
code=AzNx2xXGF5Tq6j4swOUY4T1kh4utXPBfMwRoGVMd&
redirect_uri=https%3A%2F%2Flocalhost%3A8443%2Ftest%2Fa%2Fpkjwt_2_signedreq%2Fcallback&
client_assertion=eyJraWQiOiJwa2p3dF8xX2sxIiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiJwa2p3dF8yX3
NpZ25lZHJlcSIsImF1ZCI6Imh0dHBzOlwvXC8xOTIuMTY4LjEwMS4xMzA6OTAzMVwvYXNcL3Rva2VuLm9hdXRo
MiIsImlzcyI6InBrand0XzJfc2lnbmVkcmVxIiwiZXhwIjoxNTQ0NDUwNTc3LCJpYXQiOjE1NDQ0NTA1MTcsIm
p0aSI6ImpnYWl1ZUtLZnBvdjBsSUprWHp6In0.bAcg3hXoTQvEkE8QBVpecFqkxTLtPqPgti5IKhmWkQD8bbQb
WFnVM_xepD_VttWxQataNtRPnuE_9QUO8B72uPPgEDE0u4aGt3UiApX_jzCxdO6MuZAYU8OpVYIaZDCAjQl-XR
ZBz_LiVLo-JlwyEKI8w4tLqntLoi5myO0EK3OuMpIHnCumEersiACN1mu86sBmWhl4bFIJWpBiL17D2e7wRsTV
cIzuwFsQJFk-KOBP3yh2x9QYdwlbMJt6kZTF0_8cqSDA_A8pQoO9w7MySP6YbEW4A_rb-VLFHEX3gTJ4ztTCKd
81vL3iIgqeBO37WFL7uqI9PFxlRg635GBVfA&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
```

Decoding the above client_assertion gives:

```
{
  "kid": "pkjwt_1_k1",
  "alg": "RS256"
}
{
  "sub": "pkjwt_2_signedreq",
  "aud": "https://192.168.101.130:9031/as/token.oauth2",
  "iss": "pkjwt_2_signedreq",
  "exp": 1544450577,
  "iat": 1544450517,
  "jti": "jgaiueKKfpov0lIJkXzz"
}
```

where `https://192.168.101.130:9031/as/token.oauth2` is the token endpoint address and `pkjwt_2_signedreq` is the `client_id`.

Processing rules

An authenticating client:

- In case `private_key_jwt` is used:
  - MUST generate a JWT with a unique `jti` of at least 128-bit entropy cryptopgraphic random (as per https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.2.3.2). Tokens and `jti` values MUST be used only once, the same token cannot be used to authenticate more than once.
  - MUST sign the generated JWT using the private key corresponding to its PKI-overheid certificate.
- In case `tls_client_auth` is used:
  - MUST pre-register its client certificate subjectDN as `tls_client_auth_subject_dn` (see OIDC Client registration Request).

A validating server:

- MUST validate the authentication as per specifications and best practices for the applicable method.
- MUST validate the used authentication method is registered for that client.
- In case `private_key_jwt` is used:
  - MUST NOT allow the same token (jwt) to be used more than once to avoid replay attacks, e.g. using the `jti` to perform replay detection.
  - MUST NOT accept the client authentication tokens after the expiration time, the value of `exp`.
- In case `private_key_jwt` or `tls_client_auth` is used: MUST validate the PKI-overheid certificate applicable to the signature, including revocation checking etc as listed under PKI.

OIDC Message notation

All messages in these specifications are OIDC (or OAuth2) messages, unless explicitly stated otherwise. Any OIDC message has to be formatted according to these specifications, in order to ensure interoperability and proper processing.

As per the JSON specifications (https://tools.ietf.org/html/rfc8259#section-8.1), messages MUST be encoded using UTF-8.

Signed messages

In case messages are defined as signed JSON / JWT, the following applies:

- Signing MUST use JWS (https://tools.ietf.org/html/rfc7515)
  - Default is to use the compact serialization format (https://tools.ietf.org/html/rfc7515#section-3.1), as specified in OIDC (https://openid.net/specs/openid-connect-core-1_0.html#rnc), unless explicitly stated otherwise.
- Signing MUST use a private key corresponding with a PKIoverheid certificate for signatures.
- The JWS MUST include a `kid`, referring to the key/certificate as listed in the (referenced) JWKset in the party's metadata that can be used to validate the signature; see OIDC JWKS for more information.
- Signing MUST use an algorithm listed under OIDC Algorithms.
- Signatures MUST be validated as per best practices

Encrypted messages

In case messages are defined as encrypted JSON / JWT, the following applies:

- Encryption MUST use JWE (https://tools.ietf.org/html/rfc7516).
  - Default is to use the compact serialization format (https://tools.ietf.org/html/rfc7515#section-3.1), as specified in OIDC (https://openid.net/specs/openid-connect-core-1_0.html#rnc), unless explicitly stated otherwise.
- Encryption MUST be addressed to the intended recipient, encrypting to the recipient's PKIoverheid certificate.
- The JWE MUST include a `kid`, referring to the key/certificate as listed in the (referenced) JWKset in the recipient's metadata that can be used to decrypt the message; see OIDC JWKS for more information.
- Encryption MUST use an algorithm listed under OIDC Algorithms.

OIDC JWKS

Messages and signature/encryption contained therein can make use of references to cryptoegraphic keys. Keys are shared in a format known as a JSON Web Key (JWK) or JSON Web Keyset (JWKS or JWK set).

RD / OP

The RD / OP MUST publish the keys/certificates it uses for signing and keys that others can use to encrypt OIDC messages, as a JWK set, by referencing it from its metadata as `jwks_uri`.

DV-system / Client

Any DV-system / Client MUST publish the keys/certificates it uses for signing and keys that others can use to encrypt OIDC messages, as a JWK set, either by referencing it upon registration as `jwks_uri` (RECOMMENDED) or by including it as `jwks` in the registration request.

JWKS transport

The form is a document in JSON Web Key format, containing a key set with all relevant keys used by the publishing party. The keyset can be obtained using a synchronous HTTP GET request to a HTTPS location when referenced as `jwks_uri`. The response MUST have a Content-Type value of `application/jwks-set+json`, as per RFC 7517 §7 (https://tools.ietf.org/html/rfc7517#section-7) and contain a JWKS keyset as defined in RFC 7517 and the contents as specified below.

JWKS contents

The content of the JWKS is as follows, with all except 'keys' as a *per key* set of parameters:

key	0..n	description	comment	example	spec
keys	1..n	array of key objects		`{ "keys:" [ ... ] }`	https://tools.ietf.org/html/rfc7517#section-5.1
kid	0..1	Key identifier, MUST be included. A `kid` can be arbitrarily chosen by a party, but a `kid` MUST be unique, at minimum within a JWKset used by a party and over time; preferably globally unique.			https://tools.ietf.org/html/rfc7517#section-4.5 https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.1.5
kty	1	Key type.	For key types, see RFC7518 (JWA) and https://www.iana.org/assignments/jose/jose.xhtml#web-key-types .	`"RSA"`	https://tools.ietf.org/html/rfc7517#section-4.1

					https://tools.ietf.org/html/rfc7518#section-6.1
n	0..1	In case of a (public) key of type "RSA", the value of the modulus MUST be included as parameter "n".			https://tools.ietf.org/html/rfc7518#section-6.3.1.1
e	0..1	In case of a (public) key of type "RSA", the value of the exponent MUST be included as parameter "e".			https://tools.ietf.org/html/rfc7518#section-6.3.1.2
crv	0..1	In case of a (public) key of type "EC", the value of the curve MUST be included as parameter "crv".			https://tools.ietf.org/html/rfc7518#section-6.2.1.1
x	0..1	In case of a (public) key of type "EC", the value of the x coordinate of the EC point MUST be included as parameter "x".			https://tools.ietf.org/html/rfc7518#section-6.2.1.2
y	0..1	In case of a (public) key of type "EC", the value of the y coordinate of the EC MUST be included as parameter "y".			https://tools.ietf.org/html/rfc7518#section-6.2.1.3
use	0..1	intended usage of a (public) key.	"sig" and "enc"	`"sig"`	https://tools.ietf.org/html/rfc7517#section-4.2
key_ops	0..n	intended operation(s) for usage of a (public) key.		`["wrap_key"]`	https://tools.ietf.org/html/rfc7517#section-4.3
alg	0..**1**	Algorithm to be used with this key, MUST be included.	For algorithm values, see https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms	`"PS256"`	https://tools.ietf.org/html/rfc7517#section-4.4 https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.1.5
x5u	**0**..1	URI to X509 certificate, not to be used.			https://tools.ietf.org/html/rfc7517#section-4.6

x5c	0..**n**	List (array) of X509 certificate(s), as Base64 DER encoded PKIoverheid certificate(s). The first certificate is the RD / OP's certificate, followed by the rest of that certificate's chain.			https://tools.ietf.org/html/rfc7517#section-4.7
x5t	**0**..1	SHA-1 thumbnail of X509 certificate, not to be used.			https://tools.ietf.org/html/rfc7517#section-4.8
x5t#S256	**0**..1	SHA-256 thumbnail of X509 certificate, not to be used.			https://tools.ietf.org/html/rfc7517#section-4.9

JWKS processing

JWKS processing rules for the publishing party :

- MUST publish the JWKS with all relevant keys as used in combination with the specified metadata or registration referencing or including this JWKS for that party.
- In case a `jwks_uri` referencing this JWKS is used, MUST publish only via HTTPS, using a PKIoverheid certificate for the server.
- SHOULD use separate keys for multiple unrelated key usage/operations.
- MUST include the public key parameters with the same values of the corresponding X509/PKI certificate included as `x5c`, as per RFC7517 §4.7 (https://tools.ietf.org/html/rfc7517#section-4.7).
- MAY require authentication of a party requesting this JWKS for authorisation purposes and MAY adapt the content to contain requesting party specific settings.
    - In case symmetric keys are included, the requesting party MUST be authenticated and authorized before provisioning the JWKS and the content MUST be adapted to the requested party.

JWKS processing rules for the consuming party:

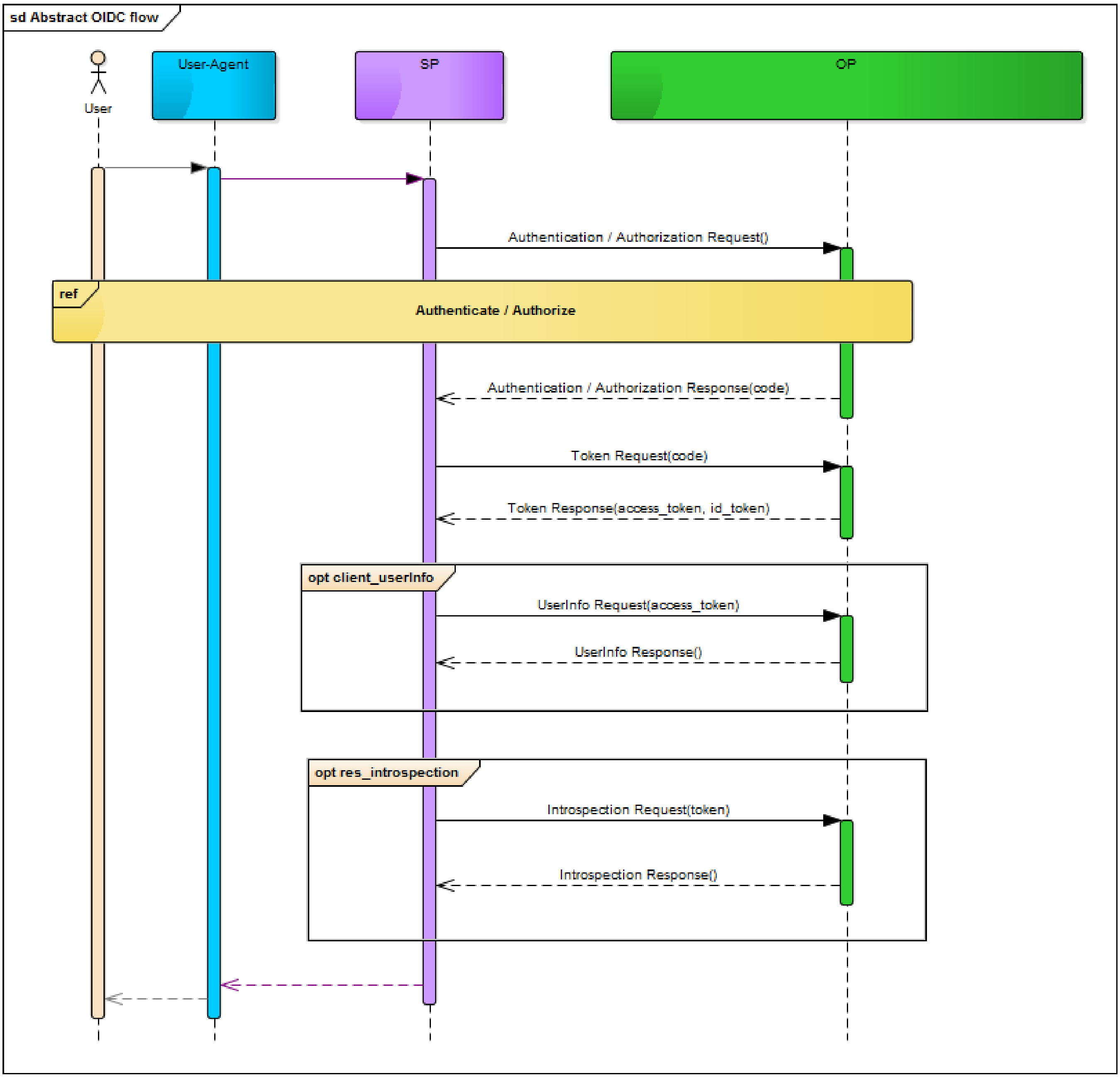
- MUST ensure the JWKS is retrieved from a verified trusted party, only use JWKS acquired from a validated and secure HTTPS connection using a PKIoverheid certificate and using a trustworthy location (URL).
- MUST use the keys and algorithms listed in the JWKs when verifying signatures/MACs or for encrypting/decrypting OIDC messages.
- MUST validate X509 certificates as per PKIoverheid requirements and best practices, before using those certificates.
- SHOULD use the `kid` (in combination with the identity of the party) to identify the relevant key to be used for signing/encrypting/validating messages.

OIDC Flow

The OIDC Flow describes the default use case for authentication and authorization of a User. This is the case where a User wants to consume a Service, using a web-browser (or technically, a User-Agent).

This flow effectively is the Authorization Code Flow from OAuth2 / OpenID Connect.

High-level functional information flow


sd Abstract OIDC flow
User
User-Agent
SP
OP
Authentication / Authorization Request()
ref
Authenticate / Authorize
Authentication / Authorization Response(code)
Token Request(code)
Token Response(access_token, id_token)
opt client_userInfo
UserInfo Request(access_token)
UserInfo Response()
opt res_introspection
Introspection Request(token)
Introspection Response()

Note that the above diagram describes the OIDC flow from an information flow perspective. The technical control and message flow is more detailed and can deviate somewhat from the above abstraction.

When a User wants to consume a Service, he or she does so, by trying to access the Service online using a User-Agent (typically a browser). Before the Service Provider will grant access, it will send an OIDC Authentication / Authorization Request to the RD / OP. The RD / OP will have the User authenticate and, if applicable, apply (delegated) authorization information, before returning a OIDC Authentication / Authorization Response (a.k.a. the Authorization Grant).

With the authorization code thus obtained, the Service Provider can request a Token with the OP using the OIDC Token Request. The resulting OIDC Token Response contains both an OIDC Access Token and an OIDC ID Token. The Access Token can be used to access the actual Service, also known as a Protected Resource hosted by a Resource Server (in OAuth2 /OIDC). The ID Token will convey information about the User to the Service Provider. Further details are provided under OIDC Token interaction.

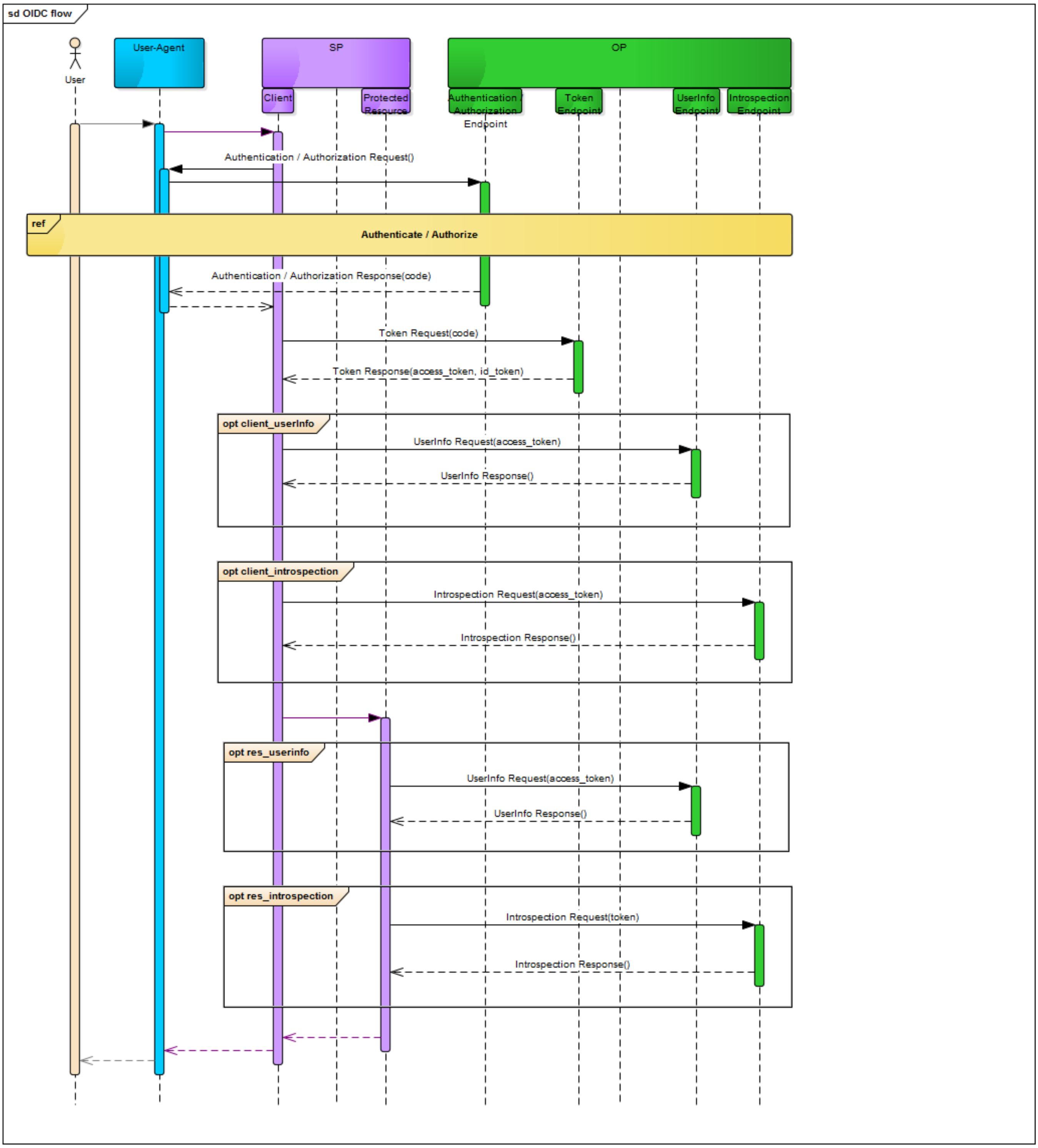
Additional information about the User may be requested through a OIDC UserInfo Request and will be provided in a OIDC UserInfo Response. Details can be found under OIDC Userinfo interaction.

Finally, the Service Provider may inquiry the validity of and more information about an access token, using a Introspection Request to get a Introspection Response. Details about Token Introspection can be found under Introspection interaction.

Although these interactions may seem redundant, this allows them to be split over various roles and thereby systems, as will be described below and elsewhere in these specifications. This has benefits for implementations and may even be a necessity in various more detailed use cases.

Low-level technical flow

A more detailed, technical oriented diagram is shown below.



The steps in the diagram above are identical though more detailed than the first, high-level, diagram. They represent the actual flow of messages and control in the OAuth2 / OpenID connect protocol as applied in these specifications.

Note that the SP and OP are split in order to represent that both fulfill multiple (technical) roles in protocol. Interactions in the diagram above are aligned with the relevant role involved in these interactions.

A first relevant detail, is that the User-Agent of the User is redirected using a HTTP redirect from the SP to the RD / OP for the OIDC Authentication / Authorization Request. In a similar fashion, the OIDC Authentication / Authorization Response is sent using a HTTP redirect as well. By using a redirect, the protocol transfers control via the User-Agent and thus allows user interaction to occur in intermediate steps. This is specifically required in the Authenticate and Authorize process taking place, although details of that process are out of the scope of this specification. More detailed information, including alternative options for sending an Authentication / Authorization Request, can be found under OIDC Authentication / Authorization interaction.

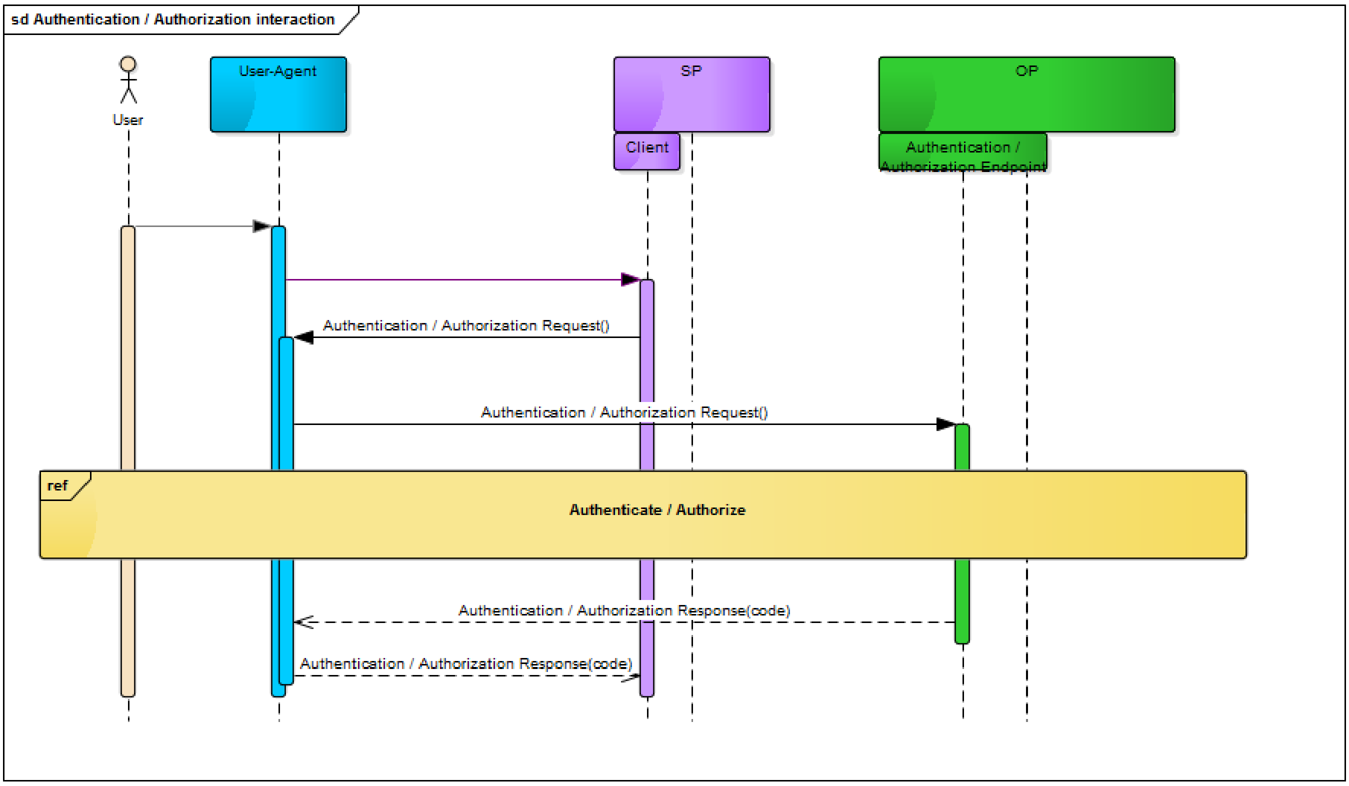
As subsequent calls do not require user interaction, they are direct, back-channel, interactions between the DV-system / Client and the RD / OP. Such interactions are often called *back-channel* calls, as they are hidden from the User's perspective.

The other noticeable difference is that both the DV-system / Client as well as the Resource Server role of the SP may initiate the UserInfo and the Token Introspection interaction. Although both roles are allowed to initiate both of the interactions, the recommended implementation is to use the UserInfo interaction from the DV-system / Client role and the Token Introspection interaction from the Resource Server role. Both interactions are optional in either case and usage depends on applicable use cases and implementation choices at the SP.

OIDC Authentication / Authorization interaction

The Authentication flow starts with a Client that wishes to authenticate and authorize a User. This is an interaction between the Client, the OpenID Provider and the User via his/her User-Agent (browser).

The DV-system / Client initiates the interaction, the RD / OP responds on its Authorization Endpoint. The message is sent via the User-Agent, so that the User can be part of the interaction.


sd Authentication / Authorization interaction
User
User-Agent
SP
Client
OP
Authentication / Authorization Endpoint
Authentication / Authorization Request()
Authentication / Authorization Request()
ref
Authenticate / Authorize
Authentication / Authorization Response(code)
Authentication / Authorization Response(code)

The above sequence diagram shows the Authentication / authorization request and response interaction, in context. This covers the first part of the full OpenID Connect or OAuth2 authorization code flow. Note that the diagram above shows the basic (information) message flow, the technical message flow may slightly deviate, as described below.

First, the User, using his/her User-Agent (browser), will try to access the application(s) provided by the Service Provider (SP) to consume a Service. As the User is not authenticated, the application (the DV-system / Client, the SP) will redirect the User-Agent to the RD / OP. This is the OIDC Authentication / Authorization Request.

In order to answer the request the RD / OP needs to authenticate, and if applicable get (delegated) authorization information for, the User. This is out of scope of these specifications.

The RD / OP then responds with the Authorization Grant, which is the authorization code in this flow, with the OIDC Authentication / Authorization Response. This is again using a redirect to return the User-Agent to the DV-system / Client, who will use this to continue the authorization code flow in the OIDC Token interaction.

Interaction transport

The OIDC Authentication / Authorization Request is sent as an HTTP redirect request, with the contents serialized in one of the three formats specified in OpenID Connect (https://openid.net/specs/openid-connect-core-1_0.html#Serializations), depending on the request mode (GET, POST or request object). The HTTP redirect ensures the User-Agent is part of the request, handing over user interaction to the RD / OP. The OIDC Authentication / Authorization Response is returned in a similar fashion.

Request MUST be sent over TLS, thus effectively applying HTTPS. As the User-Agent is involved in the interaction, the interaction uses HTTP redirects and only TLS server authentication is applicable.

In case a "clusteraansluiting" is not applicable, the DV-system / Client MAY use any of the above methods. In case a "clusteraansluiting" is applicable, the request object form – including a cluster authentication assertion as described at OIDC Client Authentication – MUST be used.

Request objects

Passing Request Objects By Reference Will Not Be Supported Initially

Due to technical reasons, passing request objects by reference will not be supported by the RV implementation initially. It may be supported in a future version (there is no planned date yet). Until that time, only passing request objects by value will be supported.

In case request objects are used, the request object MUST be signed and MAY be encrypted (as per https://openid.net/specs/openid-igov-openid-connect-1_0-03.html#rfc.section.2.4), using pre-registered algorithms. This provides integrity protection, authenticates the request and protects the user(-agent) from having its requests monitored, as per https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-16#section-1. The request object MUST be signed using the DV-system / Client's certificate. Note: in case a "clusteraansluiting" is applicable, that means the request objects MUST NOT be signed using the "cluster" systems' certificate.

In case the request object is encrypted, the request object MUST be signed then encrypted, as per OpenID Connect (https://openid.net/specs/openid-connect-core-1_0.html#RequestObject).

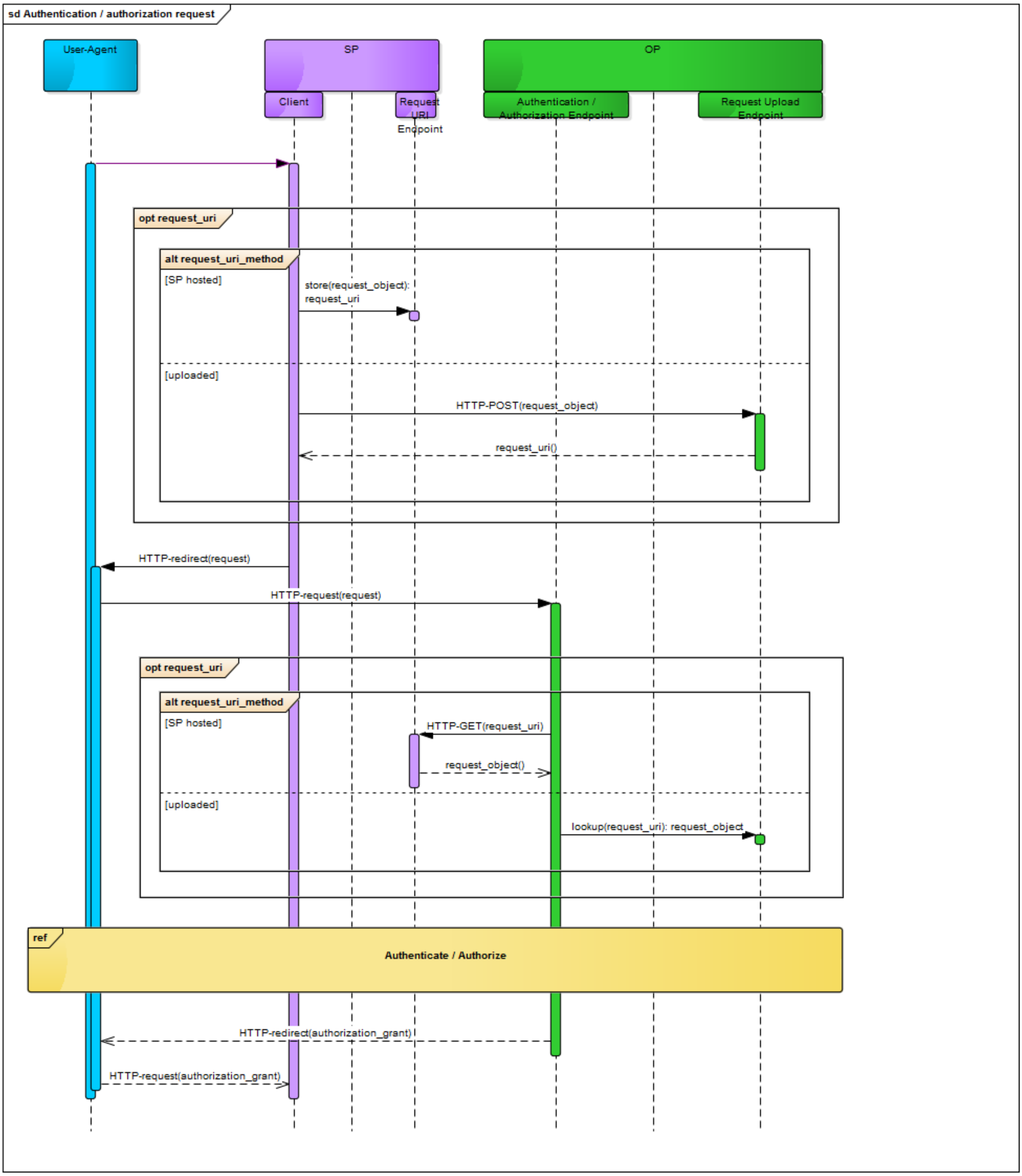
When request object serialization is used, the request object can either be passed by value or by reference. Reasons for passing the request by reference relevant for these specifications are:

- reducing the amount of data sent through the user-agent, and thus latency and improvement for UX (in line with https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-16#section-5.2).
- avoiding limitations in URI size in browsers (in line with https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-16#section-5.2).
- this form allows client authentication for the request, providing additional security (in line with https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-16#section-1).

For interoperability reasons, requests using request objects MUST include the `client_id`, `scope` and `response_type` parameter in the HTTP request with an identical value as in the request object. Other parameters SHOULD NOT be used as these will be ignored, even though this is allowed by OpenID Connect/OAuth2 JWSreq.

Request object transport

A OIDC Authentication / Authorization Request using a request object may be exchanged in one of three methods. In each of these methods the request parameters are encoded as a request object in JSON. The request object is exchanged either as parameter in a HTTP redirect or is transferred via a so-called back-channel.


sd Authentication / authorization request
User-Agent
SP
OP
Client
Request URI Endpoint
Authentication / Authorization Endpoint
Request Upload Endpoint
opt request_uri
alt request_uri_method
[SP hosted]
store(request_object): request_uri
[uploaded]
HTTP-POST(request_object)
request_uri()
HTTP-redirect(request)
HTTP-request(request)
opt request_uri
alt request_uri_method
[SP hosted]
HTTP-GET(request_uri)
request_object()
[uploaded]
lookup(request_uri): request_object
ref
Authenticate / Authorize
HTTP-redirect(authorization_grant)
HTTP-request(authorization_grant)

The diagram above depicts all three methods of exchanging the request object.

ⓘ Note that passing a request object by reference may be subject to change in future updates to these specifications.

Pass by value

In case the request object is passed by value, the request object MUST be serialized using the Compact Serialization notation, as per OIDC (https://openid.net/specs/openid-connect-core-1_0.html#RequestObject). The serialized request object is passed as `request` parameter in a HTTP redirect (see OIDC Transport interoperability and security requirements). Effectively that is typically either a URI-parameter in HTTP GET or in form request serialized form using a HTTP POST redirect. All request parameters (see OIDC Authentication / Authorization Request) are then contained in the request object using the JSON notation.

Authentication requests containing request object passed by value SHOULD use the HTTP POST method (http://openid.net/specs/openid-connect-core-1_0.html#AuthRequest) to avoid URL length limitations imposed by browsers and web servers when using the GET method.

Pass by reference

ⓘ Passing Request Objects By Reference Will Not Be Supported Initially

Due to technical reasons, passing request objects by reference will not be supported by the RV implementation initially. It may be supported in a future version (there is no planned date yet). Until that time, only passing request objects by value will be supported.

Clients MAY use a `request_uri` parameter, passing a request object by reference.

In case the request object is passed by reference, the DV-system / Client MAY either host the request itself or it MAY send the request object to the RD / OP prior to redirecting the User's user-agent (browser) for authentication and authorization, as per https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-17#section-5.2.1.

In case the request object is passed by reference, the location or identifier of the request object MUST be passed as `request_uri` parameter. The (base)URI hosting the request MUST be pre-registered for the requesting DV-system / Client. This URI MUST be unique per request using appropriate entropy to avoid guessing attacks, MUST apply TLS mutual-authentication in case the request_uri is a URL and SHOULD be discarded after it is retrieved. The RD / OP will retrieve the request object from the URI using HTTP GET. The responding server SHOULD respond with a content-type `application/jwt` and the request object as HTTP body (see https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-16#section-5.2.3).

Pass by reference, using uploading

ⓘ Passing Request Objects By Reference Will Not Be Supported Initially

Due to technical reasons, passing request objects by reference will not be supported by the RV implementation initially. It may be supported in a future version (there is no planned date yet). Until that time, only passing request objects by value will be supported.

In case the request object is uploaded prior to the Authentication / Authorization Request to the RD / OP, this MUST be done by sending a HTTP POST with the request object as body (as per https://openid.net/specs/openid-financial-api-part-2.html#request) and a HTTP-header for the Content-type `application/jose` (in line with RFC7515 rather than OIDC FAPI), using a mutual-authenticated TLS connection. The URI for uploading the request object will be shared by the RD / OP through other means.

The server MUST respond with a JSON payload containing the generated value for the `request_uri` parameter, taking the previous paragraph into account. Note that this response is neither encrypted nor signed. This response MUST have the following contents:

parameter	0..1	description	comment	example value	documentation references
request_uri	1	URI to be used as request_uri in the related authentication/ authorization request.			https://openid.net/specs/openid-financial-api-part-2.html#successful-response
iss	1	The identifier of the RD / OP.			https://openid.net/specs/openid-financial-api-part-2.html#successful-response
aud	1	The client_id of the sending requester.			https://openid.net/specs/openid-financial-api-part-2.html#successful-response
exp	1	Expiry time of the request	The request object is not guaranteed to be available after expiry time.	`1493738581`	https://openid.net/specs/openid-financial-api-part-2.html#successful-response

Request processing

A sending DV-system / Client:

- MUST authenticate to the RD / OP when uploading request objects.
- In case of a Clusteraansluiting, MUST authenticate using mutual-authenticated TLS using the cluster's TLS client certificate.
- SHOULD NOT rely on a resulting `request_uri` being valid and available for more than <*request-uri.min-lifetime*> seconds.

A receiving RD / OP:

- MUST authenticate the client before processing or storing a request object.
  - In case of a Clusteraansluiting, MUST authenticate the client with mutual-authenticated TLS using the cluster's client certificate for verification.
- MUST validate a request object before processing it.
- MUST keep a request object available through the returned `request_uri` for at least <*request-uri.min-lifetime*> seconds.

Interaction authentication

Requests are routed via the User-Agent. Therefore, the DV-system / Client cannot be authenticated directly.

In case a signed request object is used, the signature can be used for authentication of the DV-system / Client.

In case a signed request object is passed by reference, the HTTP request used for uploading to or retrieving the signed request object from the `request_uri` can be authenticated to confirm authenticity.

The user-agent will authenticate the RD / OP by its PKIoverheid certificate.

OIDC Authentication / Authorization Request

An Authentication / Authorization Request is the request for authentication and authorization of a User, initiated by the DV-system / Client.

Request encoding

Authentication request is sent from the DV-system / Client to the RD / OP authorization endpoint via the User's user-agent, using a HTTP redirect (see OIDC Transport interoperability and security requirements).

As stated under OIDC Authentication / Authorization interaction, the request can be encoded using one of three methods: HTTP GET, HTTP POST or as request object.

- In case HTTP GET is used, request parameters are encoded as URI query parameters, typically sent as part of the `Location` HTTP header in the HTTP redirect.
- In case HTTP POST is used, request parameters are encoded as form parameters (`application/x-www-form-urlencoded`), combined with an HTML form, auto-submitted using JavaScript or manually submitted by the user, for the HTTP redirect.
- In case a request object is used, authentication request parameters are encoded as JSON then signed, resulting in a JWS (or JWE if encrypted). This request object itself is sent as a HTTP request parameter, named "request", in a HTTP GET or POST redirect.

For example, the DV-system / Client may return a HTTP 302 redirect which will instruct the user-agent to navigate to the authorization endpoint. The following is a non-normative example HTTP 302 redirect response by the DV-system / Client (Lines breaks and formatting added to improve readability):

```
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
```

Upon processing this response from the DV-system / Client, the user-agent will send an HTTP GET request to the authorization endpoint (https://openidprovider.example.com/authorization-endpoint in this example) to initiate the authentication / authorization flow :

```
GET /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 HTTP/1.1
Host: openidprovider.example.com
```

Request contents

This message has the following contents:

parameter	0..n	description	comment	example value	documentation references

client_id	1	DV system id, as assigned by the RD / OP upon Client Registration.	See OIDC Client registration Response.		https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest https://tools.ietf.org/html/rfc6749#section-4.1.1
iss	0..1	SHOULD be included in case a request is sent as a request object. The value MUST be the issuer, which by definition is the client_id of the DV-system / Client.	The issuer is typically redundant with the client_id. As such, the issuer is not mandatory (SHOULD instead). If the issuer (`iss`) is omitted, any receiver MUST fallback to and interpret the client_id as `iss`.	*client_id*	https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-17#section-4
aud	0..1	MUST be included in case a request is sent as a request object. The value MUST be the intended recipient, which by definition is the identifier of the RD / OP.		https://rd.example.nl/op	https://tools.ietf.org/html/draft-ietf-oauth-jwsreq-17#section-4
scope	2..n	Scope of this Authentication / authorization request ("doelbinding"), scope values separated by space characters. As per OpenID Connect, the scope '`openid`' MUST always be included. In addition, one or more scope(s) referencing a registered service instance (set) MUST be included. This defines the actual service(s) for which authentication / authorization is requested and the information registered for the referenced service(s) will be used to obtain a user's consent.	The registered service instance (set) is referenced by a UUID. The scope for this ServiceUUID has the prefix '`urn:nl-gdi-services:`' with the UUID concatenated. See Usage of datamodel Services for the working of service instance(set)s.	`openid urn:nl-gdi-services:ea0eb2b6-9cf5-4a6e-99de-e41da799eb8b`	https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest https://tools.ietf.org/html/rfc6749#section-4.1.1
response_type	1	Type of response		`code`	

		requested. Only authorization code flow is supported hence the only allowed value for this parameter is code.			https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest https://tools.ietf.org/html/rfc6749#section-4.1.1
response_mode	0..1	Optional. Overrides the default response mode type, which is the query mode.	Only `query` and `form_post` are supported. The `fragment` mode MUST NOT be used.	`query`	https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest https://openid.net/specs/oauth-v2-multiple-response-types-1_0.html https://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html
redirect_uri	1	URI where the application expects the user/browser to return once the authentication is completed.	URI MUST be HTTPS and pre-registered.	`https://dv.example.nl/eid/auth`	https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest https://tools.ietf.org/html/rfc6749#section-4.1.1
state	1	State of DV application, allows to bind request and response to the user's session at the DV. This value will be included in the authorization endpoint response.	MUST be used by the DV-system / Client for CSRF, XSRF protection.		https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest https://tools.ietf.org/html/rfc6749#section-4.1.1 https://openid.net/specs/openidigov-openid-connect-1_0-03.html#rfc.section.2
nonce	1	Random value, used only once, to allow the DV-system / Client to apply replay-detection. This value will be included in the ID token.	New, cryptographic random values MUST be used for every message.		https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest
code_challenge	0..1	PKCE code challenge, RECOMMENDED to use. See RFC7636.	A random code (verifier) MUST be generated, as per RFC 7636 §4.1. The challenge is calculated as a SHA-256 digest over the verifier, as per RFC 7636 4.2.		https://tools.ietf.org/html/rfc7636#section-4.3

code_challenge_method	0..1	PKCE code challenge verifier method, MUST be present if `code_challenge` is used.	Only the `S256` method is to be used, for PKCE. The `plain` method MUST NOT be used.	`S256`	https://tools.ietf.org/html/rfc7636#section-4.3 https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.1.7
urn:nl-gdi-eid:1.0:oidc:cluster_assertion	0..1	In case of a "clusteraansluiting", MUST be included. MUST NOT be used in other cases.	See "clusteraansluiting" under OIDC Client Authentication for more information on this cluster authentication assertion.		Custom to these specifications.
claims	0..1	Optional. This parameter is used to request that specific Claims be returned. The value is a JSON object listing the requested Claims. See below.	See below.		https://openid.net/specs/openid-connect-core-1_0.html#ClaimsParameter
id_token_hint	0..1	Optional. MAY be used to pass an ID Token previously issued by the RD / OP, to help optimize the UX. The `id_token_hint` and `id_token_hint_jti` are mutually exclusive.	As per OIDC, the ID Token MUST be decrypted by the Client (if applicable). The original signed ID Token SHOULD be re-encrypted by the Client to the RD / OP. Note: The re-encrypt requirement is more strict than OIDC, which has this optional (MAY instead of SHOULD).	*base64.base64.base64* *base64.base64.base64.base64.base64*	https://openid.net/specs/openid-connect-core-1_0.html#AuthRequest
urn:nl-gdi-eid:1.0:oidc:id_token_hint_jti	0..1	Optional. MAY be used to reference an ID Token previously issued by the RD / OP, to help optimize the UX. The `id_token_hint` and `id_token_hint_jti` are mutually exclusive.	The value of the `jti` parameter of a previously received ID Token MUST be used.		Custom to these specifications.
urn:nl-gdi-eid:1.0:oidc:idp_hint	0..1	Optional, identifier of a valid IDP. MAY be used to indicate a pre-selected	The actual IDP for the authentication may deviate as this	`https://eidas.example.nl/de`	Custom to these specifications.

		IDP (*authenticatie dienst*), in order to optimize the user experience (UX) and indicate the IDP to be used, as selection of the IDP already occured at the DV-system / Client.	only indicates a pre-selection to optimize UX. A DV-system / Client SHOULD verify the ID Token for the actual IDP, in cases where a previous token is provided. Note that use cases requiring specific IDPs are rare and SHOULD NOT be applied.		
urn:nl-gdi-eid:1.0:oidc:mr_hint	0..1	Optional, identifier of a valid MR. MAY be used to indicate a pre-selected Machtigingenregister (machtigingsdienst), in order to optimize the user experience (UX) and indicate the MR to be used, as selection of the MR already occured at the DV-system / Client. In case present but left empty, SHOULD be used by the RD / OP to trigger the flow for the use case of representation.	The actual MR for the authorization may deviate as this only indicates a pre-selection to optimize UX. A DV-system / Client SHOULD verify the ID Token for the actual MR, in cases where a previous ID Token is provided. Note that use cases requiring specific MRs are rare and SHOULD NOT be applied.	`https://mr.example.nl`	Custom to these specifications.
display	0..1	Optional, will be ignored. Always effectively processed as `page` and `touch`.	The RD / OP MUST always be responsive and support a touch interface, and MUST always be full page.		https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest
prompt	**0**..1	SHOULD NOT be used. Contrary to OIDC, the DV-system / Client is not expected to indicate a `prompt`. The `prompt` MUST be ignored if present.	The RD / OP MUST follow level of assurance rules to determine the applicable "prompt".		https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest
max_age	0..1	Optional, will be ignored.	Always effectively processed as per level of assurance rules.		https://openid.net/specs/openidconnect-core-1_0.html#AuthRequest
ui_locales	0..1			`nl-NL en`	https://openid.net/specs/openid

		Optional. MAY be used to pass user's preferred or DV-system / Clientsupported locale (s) for user interaction.			connect-core-1_0. html#AuthRequest
claims_locales	0..1	Optional. MAY be used to pass User's or DV-system / Clientpreferred locale(s) for claims.	Defaults to `"nl"`.	`nl en fr`	https://openid.net/specs/openid connect-core-1_0. html#ClaimsLanguagesAndScrip
login_hint	0..1	Optional, will be ignored.	Not applicable under this profile, and for privacy reasons undesired to provide an identifier for the user.		https://openid.net/specs/openid connect-core-1_0. html#AuthRequest
acr_values	0..1	Optional, will be ignored.	Values for the requested level of assurance (authentication context class referenced) MUST be pre-registered and will be process according pre-registration.		https://openid.net/specs/openid connect-core-1_0. html#AuthRequest
registration	0	MUST NOT be used. Contrary to OIDC, the DV-system / Client MUST NOT support Self-Issued OP's under these specifications.			https://openid.net/specs/openid connect-core-1_0. html#RegistrationParameter
resource	0..n	URI(s) referring to the resource(s) that is/are to be accessed. Each reference MUST be an absolute URI.	Typically these are HTTPS URLs (`https://...`), matching the structure of the protected resource that will be accessed (e.g. API).	`http://example.com/service/read`	https://tools.ietf.org/html/draft-ietf-oauth-resource-indicators-01#section-2 https://tools.ietf.org/html/rfc3986#section-4.3 Used in https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-16#section-2.1 as well
include_granted_scopes	0..1	Optional. Reserved for future updates to these specifications.			https://tools.ietf.org/html/draft-ietf-oauth-incremental-authz-01#section-4
existing_grant	0..1	Optional. Reserved for future updates to these specifications.			https://tools.ietf.org/html/draft-ietf-oauth-incremental-authz-01#section-5

Request processing

Sending DV-system / Client processing rules:

- MUST use the `client_id` as assigned by the RD / OP upon Client Registration (see OIDC Client registration Response).
- MUST include one or more `scope`(s) to be used for this authentication / authorization, in addition to the default scope `openid`. Each `scope` MUST reference a Service Instance or Service Instance Set registered for the `client_id`. See Datamodel Services and Usage of datamodel Services.
- MUST use only pre-registered `redirect_uri`'s, see OIDC Client registration Request.
- MUST use a `state` to protect the request from CSRF-, XSRF-attacks and other type of attacks.
  - MUST use a cryptographically secure random or otherwise non-predictable method to set a `state`; the `state` MUST be unique during at least 12 months (across all systems for the same Client) and MUST bind the request and `state` to the session with the User's user-agent (browser).
- MUST include a `nonce` in each request in order to apply replay-detection.
  - MUST use a cryptographically secure random or otherwise non-predictable method to set a `nonce`; the `nonce` MUST be unique during at least 12 months (across all systems for the same Client) and MUST bind the request and `nonce` to the session with the User's user-agent (browser).
- SHOULD (RECOMMENDED to) use the `state` and/or `nonce` parameter(s) to bind the request to the user's session.
- SHOULD include a PKCE code challenge, to protect against authorization code interception attack.
  - In case a code challenge is included, the PKCE code challenge method `S256` MUST be used in combination with a cryptographic random `code_challenge`, as per https://tools.ietf.org/html/rfc7636#section-4.
- SHOULD use a request object. In case a request object is used:
  - MUST sign the request object
  - MAY encrypt the request object
- MAY request claims (attributes) using the `claims` or alternatively the `scope` parameter (see below).
- MAY include either an `id_token_hint` or an `id_token_hint_jti` to help UX, MAY NOT use both.
  - In case a `id_token_hint` is included, the (unencrypted) signed `id_token_hint` SHOULD be re-encrypted by the Client to the OP.
- MAY use the `idp_hint` parameter to indicate IDP pre-selection, only values supported by the eID network MUST be used.
  - Available values for IDP values can be obtained from the RO RV (TODO how/ref).
  - Note that for "MR" (*machtigingenregister*) pre-selection MUST NOT use the `idp_hint` parameter, but `mr_hint` instead.
- MAY use the `mr_hint` parameter to indicate MR pre-selection, only values supported by the eID network MUST be used.
  - MAY use the `mr_hint` with an empty value, to indicate representation is requested or applicable.
- In case a "clusteraansluiting" is being used, MUST include a `cluster_assertion` for authentication of the cluster.

Receiving RD / OP processing rules:

- MUST only accept requests received over HTTPS.
- MUST only accept request from registered clients
- SHOULD use the value of the `nonce` for duplicate detection.

- In case a request was sent as a request object:
    - MUST validate it is listed as audience in `aud`.
    - MUST validate the `iss` if present to be the validated signer of the request object and matches the `client_id`; otherwise SHOULD use `client_id` as fallback for `iss`.
    - MUST use the value of the `nonce` for duplicate-detection.
- MUST keep the value of the `state`, `nonce` and `code_challenge` parameters in non-modified form for usage in subsequent (token) responses
- MUST validate `redirect_uri` is syntactically valid absolute URL, has HTTPS as scheme and is an exact match of one of the pre-registered `redirect_uri`'s.

    ⓘ
    - The HTTPS scheme requires native apps to claim to be the scheme handler for a HTTPS URI; this is in line with the preferred solution in https://tools.ietf.org/html/rfc8252#section-7.2.
- MUST use the `scope` to obtain the information of the service(s) for which authentication / authorization is requested.
- SHOULD use the `idp_hint` and `mr_hint` parameter to optimize the user experience (UX) and minimize the steps in user interaction.
    - In case the `mr_hint` is present with an empty value, SHOULD trigger the use case for representation.
- MAY use the value of an `id_token_hint` or an `id_token_hint_jti` to help UX.
- MUST ignore any parameter `display`, `prompt`, `max_age`, `login_hint` or `acr_values` in the request.
- MUST reject any request containing a `registration` parameter, using the `registration_not_supported` error code.

Requested Claims

ⓘ Claims parameter will not be supported in the initial RV implementation

Due to technical limitations, support for the claims parameter will not be available in the initial RV implementation.

By default scopes will be used to determine which attributes will be returned in ID tokens and from the userinfo endpoint. For example if attr1 and attr2 are associated with a scope A (in OP configuration), clients will receive attr1 and attr2 when the user grants scope A. The main difference between using scopes and claims parameter is that when using scopes, claims can not be marked as essential, clients must always check that they have received all the required attributes for their use cases (this check must be done even when using claims anyway).

Please let us know if having support for the claims parameter is vital and absolutely required for your use cases or implementation.

A DV-system / Client MAY request specific Claims (attributes) to be provided.

This can be done using the `claims` parameter in the Authentication / Authorization Request. Alternatively, attribute Claims MAY be requested using the `scope` parameter. Claims requested via the `scope` parameter will always be provided via the OIDC UserInfo Response.

ⓘ When using the `scope` parameter, the essential or value restrictions cannot be used; the DV-system / Client has to incorporate that into its own access control decision. ~~Using the `claims` parameter is therefore RECOMMENDED.~~

Specific Claims or attributes, MAY be requested to be provided either under the ID Token (using `id_token`) or the UserInfo (`userinfo`). It is up to the requesting DV-system / Client to choose based on its architecture, configuration and applicable scenarios. Note: for certain use cases, a particular usage can be mandated (TODO).

The value of any parameter under `id_token` or `userinfo` in the `claims` request parameter, MUST have values as described in OpenID Connect 1.0 core §5.5.1 (https://openid.net/specs/openid-connect-core-1_0.

html#IndividualClaimsRequests). This effectively means that a value MUST be used that is either the `null` value or an object that MAY contain any of the `essential`, `value` and `values` sub-parameters.

- In case a requested Claim (attribute) is listed as "essential", this will be used to inform the End-User that the service cannot be consumed unless the attribute is provided. If an "Essential Claim" is not available, authentication will continue and a response will be returned without the Claim (attribute). This as per the OIDC specifications (https://openid.net/specs/openid-connect-core-1_0.html#IndividualClaimsRequests).
- In case one or more values are specified in the request, the Claim (attribute) will only be provided if a specified value can be met. For instance, for the request "`18+`" with the value "`true`", only users with age 18 and above will be able to use the service authenticating for.
- Note that the combination of `value` and `values` can be ambiguous and SHOULD NOT be used.

⚠ Note that any attribute the can be requested for a service MUST be pre-registered with a purpose and a privacy policy MUST have been pre-registered for the service.

Both the `id_token` or `userinfo` parameter under `claims` parameter has the following contents:

param	0..n	description	comment	example value	documentation references
attr_name	0..*n*	Optional. MAY be used to request the attribute *`attr_name`*. Only attributes that have explicitly stated that one or more specific values MAY be requested, support such usage.	See https://openid.net/specs/openid-connect-core-1_0.html#IndividualClaimsRequests for the syntax of requesting Claims (attributes). See OIDC Attribute catalog for available attribute claim names and their semantics.	`"urn:nl-eid-gdi:1.0:attribute:18+": {"essential": true, "value": true}`	https://openid.net/specs/openid-connect-core-1_0.html#IndividualClaimsRequests OIDC Attribute catalog

Sending DV-system / Client processing rules:

- SHOULD request any applicable Claim via the `claims` parameter, MAY use the `scope` parameter instead.
- MUST only request attributes as Claim for attributes listed in OIDC Attribute catalog.
- MUST pre-register any attribute, included its purpose, that can be requested for the service and MUST provide a pre-registered privacy policy.
- TODO MR pre-selection parameters?

Receiving RD / OP processing rules:

- MUST process any attribute Claim requested using the `scope` parameter, as if it were requested as a non-essential claim, without any specified value, to be provided in the UserInfo.
- In case a Claim is requested, MUST ask consent of the User.
- In case a Claim is requested with the `true` for `essential`, MUST inform the User that consent is required for accessing a service.
- In case a Claim is requested with a (list) of value(s), MUST only ask consent

- In case a Claim is requested with a (list) of value(s) and with `true` for `essential` and the actual value for the User does not match, MUST inform the User that he/she does not meet the criteria for the service. Note that authentication still MAY be continued, as per OIDC specifications.
- TODO

OIDC Authentication / Authorization Response

This is the response, from the OpenID Provider, sent to the redirect_uri (from the OIDC Authentication / Authorization Request) as the result of the OIDC Authentication / Authorization Request. Depending on the response_mode, the response will be sent using an HTTP redirect or a form post via the user-agent.

Response encoding

The response is a HTTP redirect to the applicable `redirect_uri`, either sent using URI query or form encoding, depending on the HTP redirect method being applied.

- In case `response_mode` is set to `query`, the response parameters are encoded as URI query parameters added as URI parameters to the `redirect_uri`, typically sent as part of the `Location` HTTP header in the HTTP redirect.
- In case `response_mode` is set to `form_post`, the response parameter are encoded as form parameters (`application/x-www-form-urlencoded`), combined with for instance a auto-submitted HTML form for the HTTP redirect.

Response contents

This message has the following contents:

param	0..1	description	comment	example value	documentation references
code	1	Authorization Code grant from the OpenID provider Authorization Endpoint	This is the value that will be used to obtain tokens from the OP Token Endpoint	`JNL6+mvE3k2uAqrgE/ScOmK92W14fP+cn8+Zu7N0p/k=`	https://openid.net/specs/openid-connect-core-1_0.html#AuthResponse https://tools.ietf.org/html/rfc6749#section-4.1.2
state	1	Exact copy of the state parameter from OIDC Authentication / Authorization Request	The Client MUST validate this value.		https://openid.net/specs/openid-connect-core-1_0.html#AuthResponse https://tools.ietf.org/html/rfc6749#section-4.1.2 https://tools.ietf.org/html/rfc6749#section-10.12

Response processing

Sending RD / OP processing rules:

- MUST use a new cryptographically secure random value for the `code`; alternatively MAY issue a `code` containing information for handling the subsequent Token Request if and only if the `code` value is encrypted and also cannot be replayed or guessed.
- MUST keep the value of the `code`, bound to the `client_id` and issuing time, for processing subsequent (token) request(s).

- MUST ensure that code values have a limited validity period and unused code values are discarded after the validity period expires.
- MUST ensure that code values cannot be used more than once, code values MUST become unusable as soon as the first request is received for a code value.
- MUST ensure no collisions of `code` between clients/requests can occur.
- MUST copy the `state` value exactly from the Authorization Request.
- MUST validate the `redirect_uri` and MUST NOT send any responses to invalid redirect_uris.

Receiving DV-system / Client processing rules:

- MUST keep the value of the `code` parameter for subsequent (token) request(s)
- MUST use the value of the `state` parameter to protect against CSRF/XSRF attacks as described at https://tools.ietf.org/html/rfc6749#section-10.12.

Error handling

In case a RD / OP cannot process a (request) message or has to reject a request, an error response is sent instead of a response. An error response message is identical to the error response as defined in OIDC (https://openid.net/specs/openid-connect-core-1_0.html#AuthError) and OAuth2 (https://tools.ietf.org/html/rfc6749#section-5.2).

Effectively, an error response has the following contents:

param	0..1	description	comment	example value	documentation references
error	1	A code defining the error.	Valid values for the error code are defined in OIDC and OAuth2 specifications.	`access_denied`	https://openid.net/specs/openid-connect-core-1_0.html#AuthError https://tools.ietf.org/html/rfc6749#section-4.1.2.1
error_description	0..1	Optional. A human-readable description of the error.	The `error_description` MAY be localized using the `ui_locales` authentication request parameter.		https://openid.net/specs/openid-connect-core-1_0.html#AuthError https://tools.ietf.org/html/rfc6749#section-4.1.2.1
error_uri	0..1	Optional. A URL with further information for the error.			https://openid.net/specs/openid-connect-core-1_0.html#AuthError https://tools.ietf.org/html/rfc6749#section-4.1.2.1
state	0..1	In case a `state` parameter was included in the request: MUST			https://openid.net/specs/openid-

		be included. Exact copy of the request `state` parameter.			connect-core-1_0.html#AuthError https://tools.ietf.org/html/rfc6749#section-4.1.2.1

To prevent abuse of services and open redirection issues, RD / OP MUST NOT send error responses to invalid redirection URIs (`redirect_uri`) for the DV-system / Client. When the requested `redirect_uri` is invalid for the client or if the requested `client_id` is invalid, the RD / OP MUST NOT send an error response to the requested `redirect_uri`. In such cases, RD / OP SHOULD display an error page to the end user to improve the user experience.

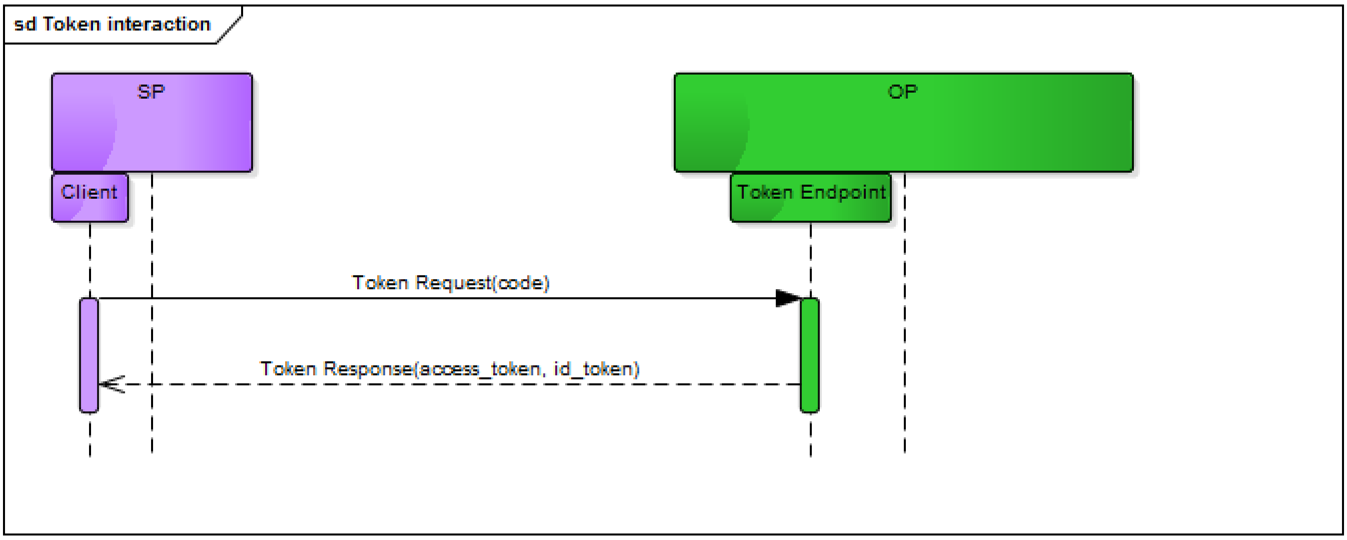
Error processing rules

- In case a DV-system / Client applies logic for handling specific errors, the value of the `error` parameter SHOULD be used as input for this logic.

OIDC Token interaction

After obtaining an Authorization Grant (Authorization Code), the DV-system / Client requests the Access Token and ID Token at the RD / OP's Token Endpoint. The Token Endpoint can be found in the OIDC Provider metadata.

The DV-system / Client initiates the interaction, triggered by the User's user-agent returning with the OIDC Authentication / Authorization Response, the RD / OP responds on its Token Endpoint.



The Token Request is initiated by the DV-system / Client, as a request to the RD / OP's Token Endpoint with the code (a.k.a. authorization grant) obtained from the OIDC Authentication / Authorization Response.

The OP will respond with an OIDC Access Token and an OIDC ID Token.

Interaction transport

Request to be sent by DV-system / Client using HTTP POST over HTTPS, to RD / OP's Token Endpoint (https://openid.net/specs/openid-connect-core-1_0.html#TokenRequest). This effectively is a form of back-channel interaction. The Token Endpoint can be found using the OIDC Provider metadata.

Interaction authentication

A requesting client MUST be authenticated, using the registered `token_endpoint_auth_method` for the client (see OIDC Client registration Request).

As the request is sent over HTTPS, the server is always authenticated using it's PKIoverheid certificate.

OIDC Token Request

Request sent by the DV-system / Client to the RD / OP's Token Endpoint as an HTTP POST request over HTTPS.

Request encoding

This message is sent using Form Serialization, with the request parameters encoded as form parameters (see https://openid.net/specs/openid-connect-core-1_0.html#FormSerialization / https://tools.ietf.org/html/rfc6749#section-4.1.3). Therefore the Content-Type of the request is `application/x-www-form-urlencoded`.

Request example

The following is a non-normative example of a Token Request using `private_key_jwt` client authentication method (Line breaks and formatting applied for display purposes):

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

grant_type=authorization_code&
code=auotI-YrhcF3A_JKOXk-i9sCfdcdlE_tdnA8Y7te&
redirect_uri=https%3A%2F%2Fserviceprovider.example.com%2Fcallback-url&
client_assertion=eyJraWQiOiJwa2p3dF8xX2sxIiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiJwa2p3dF8xIi
wiYXVkIjoiaHR0cHM6XC9cL29wZW5pZHByb3ZpZGVyLmV4YW1wbGUuY29tOjkwMzFcL2FzXC90b2tlbi5vYXV0
aDIiLCJpc3MiOiJwa2p3dF8xIiwiZXhwIjoxNTQ0NjE3NDkxLCJpYXQiOjE1NDQ2MTc0MzEsImp0aSI6IjdjcT
g0T1VLemNaOXJLS2w3VVhPIn0.KtgNUeO_fk-0BiMCAmYkaBzowtf3660038SnQqpSn4wV4LcNVRASkboijtFS
cpSasVmMuZvNey3-iClfmYjlFVBca5VVj6Y1rJ2YvipRIBlg4vCWQ8DteW0JZ3jIhYdkXLtPkyR-HhrJDZpfPx
2CmQjOuUg9L55BIhQfbG2rk4adPa1NxaNqntTiJew-YEKd4tUD374uCy_AwXDkSJ35lBB5aLBGi7dYz_7OnXxz
lt4vl0uKuzcy2mvv-bnE97ewrzrIEF1m2Ec4YAU9MO9h6_owqlDI0WAUT0RYaS73mOo8szxTV_OS2oYCU2zd1o
IH_SgiQZ-5qosaCqi4kOMz6w&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer
```

See OIDC Client Authentication for more information on `private_key_jwt` client authentication method and `client_assertion`.

Request contents

Token requests support the following parameters:

param	0..1	description	comment	example value	documentation references
grant_type	1	Type of grant, fixed value for authorization code flow		`authorization_code`	https://openid.net/spe /openid-connect-core- html#TokenRequest https://tools.ietf.org/h /rfc6749#section-4.1.
code	1	Copy of value received in Authorization Response	replay detection, expire after 10 minutes	`JNL6+mvE3k2uAqrgE/ScOmK92W14fP+cn8+Zu7N0p/k=`	https://openid.net/spe /openid-connect-core- html#TokenRequest https://tools.ietf.org/h /rfc6749#section-4.1.
redirect_uri	0..1	URI where the application expects the user/browser to return	Value MUST be identical to the redirect_uri in the	`https://dv.example.nl/eid/auth`	https://tools.ietf.org/h /rfc6749#section-4.1.

			authentication request.		
client_id	0..1	depends on client authentication: • MUST be included when client authentication using `tls_client_auth` is utilized • MAY be included in other cases	Note: When using a combination of client authentication methods including `tls_client_auth` – most noteworthy the 'clusteraansluiting' – the client_id MUST be included.		https://tools.ietf.org/h /rfc7521#section-4.2
client_assertion	0..1	MUST be included when client authentication using `private_key_jwt` is utilized	See OIDC Client Authentication for details.		https://openid.net/spe /openid-connect-core- html#ClientAuthentic https://tools.ietf.org/h /rfc7521#section-4.2
client_assertion_type	0..1	MUST be included when client authentication using `private_key_jwt` is utilized		`urn:ietf:params:oauth:client-assertion-type:jwt-bearer`	https://openid.net/spe /openid-connect-core- html#ClientAuthentic https://tools.ietf.org/h /rfc7521#section-4.2
client_secret	0	MUST NOT be used, as client_secret_post is not supported in this specification.			https://tools.ietf.org/h /rfc6749#section-2.3.
code_verifier	0..1	In case PKCE was applied in the Authentication Request (i.e. a code_challenge was present), the code verifier MUST be present.	The value MUST be the original random code, used to create the hashed challenge.		https://tools.ietf.org/h /rfc7636#section-4.5
resource	0..n	URI(s) referring to the resource(s) that is/are to be accessed with the resulting tokens. Each reference MUST be an absolute URI.	Typically these are HTTPS URLs (`https://...`), matching the structure of the protected resource that will be accessed (e.g. API).	`http://example.com/service/read`	https://tools.ietf.org/h /draft-ietf-oauth-resou indicators-01#section-

Request processing

Sending DV-system / Client processing rules:

- SHOULD request the ID Token a.s.a.p. after receiving the code; MUST request the ID Token no later than <*grant.expiry-time*> after initiating the Authentication / Authorization Request.
- MUST use the `client_id` as received during registration (see OIDC Client registration Response) and as used for the Authentication / Authorization Request.
- MUST apply the registered client authentication method.
- MUST use the code from the preceding received Authentication Response
- MUST use the same `redirect_uri` as used in the preceding Authentication Response
- MUST validate the TLS certificate of the server.

Receiving RD / OP processing rules:

- MUST authenticate the requesting client and only process the request if the authentication matches the registered authentication details for the client.
    - In case private_key_jwt client authentication method is used, MUST apply detection of duplicates based on the `jti` parameter, in order to prevent replay attacks.
- MUST authenticate a request if the request object method is used and MUST verify the keys and request object signing & encryption algorithms (`request_object_signing_alg`, `request_object_encryption_alg` and `request_object_encryption_enc`) registered for the DV-system / Client are being used.
- MUST validate the request.
- MUST validate that the `code` was issued in response to an Authentication Request for the same client.
- MUST only process a request for a `code` once.
- MUST only process requests for recent (no older than <*grant.expiry-time*>) issued codes.
- MUST validate that the `redirect_uri` matches a registered `redirect_uri` for the client and is the same as the `redirect_uri` used in the preceding Authentication request.
- MUST validate the `code_verifier` if present or if the `code_challenge` and/or `code_challenge_method` were present in the Authentication Request.
    - MUST validate the `code_challenge` value with the `code_verifier` from the Token Request using the PKCE S256 method, as per RFC 7636 §4.6 (https://tools.ietf.org/html/rfc7636#section-4.6), in case the `code_challenge_method` is `S256`.

OIDC Token Response

Response to an OIDC Token Request, sent synchronously in response to a DV-system / Client's Token Request, sent from RD / OP to DV-system / Client over HTTPS.

Response encoding

This message is encoded as JSON object, containing an (OAuth2) Access Token as well as an (OpenID Connect) ID token. The Content-Type of the HTTP response is `application/json.`

The following is a non-normative token endpoint response example (Lines breaks and formatting added to improve readability) :

Token Response

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

{
"access_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6Imp3dGF0c2lnbmluZyJ9.eyJzY29wZSI6Im9wZW5pZCBuY
W1lIiwiY2xpZW50X2lkIjoicGtqd3RfMSIsImp0aSI6IjJ3eFFUYjY2aWZkTEc3NGJzaUdtVzlhZzRSZ3B1YmVlIiwi
aXNzIjoiaHR0cHM6Ly9vcGVuaWRwcm92aWRlci5leGFtcGxlLmNvbTo5MDMxIiwiYXVkIjoiaHR0cHM6Ly9yZXNvdXJ
jZXNlcnZlci5leGFtcGxlLmNvbSIsInN1YiI6InRlc3QyIiwiYXpwIjoicGtqd3RfMSIsImV4cCI6MTU0NDYyNzE1OX
0.ROxqA2dIUuYBzJ1P7By4RqRx-GxbBwdZOvF4V1xXfnQ1-cr-zK0ktAJBZGQthzQ1PmyBrx7F_in0Cyf01_WD_1SwU
sHEt5jVGKfeSlvvyRxNHVBnwPeWOl97vlqynYiNi3QUgg32T0qwtVUl_Xf8QxFl7led1-eWPMN_yoRJPSgpBHpmGpMf
Hat7lHl2TlR_t7hXsPHBIjCulKJk7swK_Fh1OR3ChG7ppvRPygqA1sz_IDQjcqi1PClaxLObnhgRdThpCBe3YYF84rP
1-iCOjnRfygGkPu8Qv0JXA5UqQ7sxtOP4YQrxQdGnSsTVI94p52M_qODQwWF1opz82RIPyw",
"id_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6ImVRWVdFa05NOEw0eFpsRWkxWTVaY3RweEttRSJ9.eyJzdWIiOi
J0ZXN0MiIsImF1ZCI6InBrand0XzEiLCJqdGkiOiJXSXBXem5qeWh2YVExUTloQ3N1UXp4IiwiaXNzIjoiaHR0cHM6L
y9vcGVuaWRwcm92aWRlci5leGFtcGxlLmNvbTo5MDMxIiwiaWF0IjoxNTQ0NjI2ODU5LCJleHAiOjE1NDQ2MjcxNTks
Im5vbmNlIjoicElEWjJFanROTSIsImFjciI6InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDphYzpjbGFzc2VzOnV
uc3BlY2lmaWVkIiwiYXV0aF90aW1lIjoxNTQ0NjI2ODU4LCJuYmYiOjE1NDQ2MjY4MjYsIm5hbWUiOiJKb2huIERvZS
IsInNfaGFzaCI6IkUxNG10VTBWU2NjY3lvYTZucEQwSVEifQ.tFrVYPegOG92YqTKYinE30-ViYL8dRlI7YdSbHpjeV
gO2CnQVfd9D77s1-89AysOS82sBudyER0CvGzEGPsUQzUMLSQ3NkJXRhwbqyGT-nwJHdQFLzZJ66FPaAC9dG-OQ3K2G
YQVrXGtga7BGeGvBtpLRuUsOfGTQ-cyhcqQtKZuxX7sU2rekN1jM7Otx7MgPA4_gTCjQh_NAu5iZ720qTFZSqLx3KhR
MEz3IOjIeXBbaQONIVjBFDWMIvPbp_jSvPd2_7GVXheFJi3Gs-TYZP2JrfTX9Q825xkbXLTiDskuXzYVl2rJ9Ffput5
ydnO3rHs4g8NCBR_2JKeCcqaJzA",
"token_type":"Bearer",
"expires_in":299
}
```

Decoding the above access token gives :

```
{
  "alg": "RS256",
  "kid": "jwtatsigning"
}
{
  "scope": "openid name",
  "client_id": "pkjwt_1",
  "jti": "2wxQTb66ifdLG74bsiGmW9ag4Rgpubee",
  "iss": "https://openidprovider.example.com:9031",
  "aud": "https://resourceserver.example.com",
  "sub": "test2",
  "azp": "pkjwt_1",
  "exp": 1544627159
}
```

For details on the access token, see OIDC Access Token.

Decoding the above id_token gives :

```
{
  "alg": "RS256",
  "kid": "eQYWEkNM8L4xZlEi1Y5ZctpxKmE"
}
{
  "sub": "test2",
  "aud": "pkjwt_1",
  "jti": "WIpWznjyhvaQ1Q9hCsuQzx",
  "iss": "https://openidprovider.example.com:9031",
  "iat": 1544626859,
  "exp": 1544627159,
  "nonce": "pIDZ2EjtNM",
  "acr": "urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified",
  "auth_time": 1544626858,
  "nbf": 1544626826,
  "name": "John Doe",
  "s_hash": "E14mtU0VScccyoa6npD0IQ"
}
```

For details on the ID token, see OIDC ID Token.

Response contents

This message has the following contents:

key	0..1	description	comment	example value	documentation references
access_token	1	Structured (JWT) access token a.k.a. a Bearer token.	See OIDC Access Token.		https://openid.net/specs/openid-connect-core-1_0.html#TokenResponse https://tools.ietf.org/html/rfc6749#section-4.1.4 https://tools.ietf.org/html/rfc6749#section-5.1 https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.2.1
token_type	1	Type of token. Only bearer tokens supported currently.		`"Bearer"`	https://openid.net/specs/openid-connect-core-1_0.html#TokenResponse https://tools.ietf.org/html/rfc6750
refresh_token	0..1	optional refresh token, for easy token renewal			
expires_in	0..1	The lifetime in seconds of the access token.		`300`	https://tools.ietf.org/html/rfc6749#section-

					5.1
scope	0..n	Scope(s) of the access (token) granted ("doelbinding"); multiple scopes are separated by whitespace. The scope `openid` will be included by default per OpenID Connect. Other scopes included depend upon requested and authorized service(s).	The scope is omitted if it is identical to the scope requested, as allowed by the OAuth2 specification.	`"openid urn:nl-gdi-services:ea0eb2b6-9cf5-4a6e-99de-e41da799eb8b"`	https://tools.ietf.org/html/rfc6749#section-5.1
id_token	1	ID token for the authenticated /authorized user.	See OIDC ID Token.	"*base64.base64.base64*"	https://openid.net/specs/openid-connect-core-1_0.html#TokenResponse

Response processing

Sending RD / OP processing rules:

- MUST provide a Structured (JWT Bearer) access token, as specified under OIDC Access Token.
- MUST include the applicable value(s) for the `scope`, based on the scope(s) in the request, the Service Instance(s) and Service Definition(s) referenced thereby, for which consent was granted by the user and for which the user is authorized.
  - In case Claims (attributes) were requested using the `scope` parameter in the relevant OIDC Authentication / Authorization Request, these SHOULD be included in the scope provided in this OIDC Token Response.
- MAY omit the `scope` parameter, in case it is identical to the `scope` parameter in the related OIDC Authentication / Authorization Request.
- MUST provide the ID Token as specified under OIDC ID Token.
- MUST set `expires_in` to <*access-token.max-validity*>.
- TODO
- ...

Receiving DV-system / Client processing rules:

- MUST verify the values for the applicable scope(s) are present, based on the scope(s) in the request, the service(s) referenced thereby, for which consent was granted by the user and for which the user is authorized.
- MUST not use access tokens or ID tokens for other purposes then those covered in the provided scope(s), as only consent and authorization for these are applicable.
- In case one or more "divisible" Service Instance Set(s) or referenced Service Definition Set(s) were requested: MUST inspect the provided scope(s) and handle access decision based on the value(s) for scope provided, these may provide only a subset of the requested scope(s). See Usage of datamodel Services for further details.
- TODO
- ...

Error handling

In case a RD / OP cannot process a (request) message or has to reject a request, an error response is sent instead of a response. An error response message is identical to the error response as defined in OIDC (https://openid.net/specs/openid-connect-core-1_0.html#TokenErrorResponse) and OAuth2 (https://tools.ietf.org/html/rfc6749#section-5.2).

An HTTP 401 (Unauthorized) status code MAY also be returned as described under invalid_client at https://tools.ietf.org/html/rfc6749#section-5.2,.

Effectively, an error response has the following contents:

param	0..1	description	comment	example value	documentation references
error	1	A code defining the error.	Valid values for the error code are defined in OIDC and OAuth2.	`invalid_grant`	https://openid.net/specs/openid-connect-core-1_0.html#TokenErrorResponse https://tools.ietf.org/html/rfc6749#section-5.2
error_description	0..1	Optional. A human-readable description of the error.	The `error_description` MAY be localized using the requests `ui_locales` parameter.	Grant expired.	https://openid.net/specs/openid-connect-core-1_0.html#TokenErrorResponse https://tools.ietf.org/html/rfc6749#section-5.2
error_uri	0..1	Optional. A URL with further information for the error.			https://openid.net/specs/openid-connect-core-1_0.html#TokenErrorResponse https://tools.ietf.org/html/rfc6749#section-5.2

Error processing rules

- In case a DV-system / Client applies logic for handling specific errors, the value of the `error` parameter SHOULD be used as input for this logic.

OIDC Access Token

The Access Token included in the OIDC Token Response, contains the actual identity/authentication information. This can be used by the DV-system / Client for its access control decision and can be used as bearer of an authorization by the DV-system / Client when making requests on behalf of the Subject to a Resource Server in subsequent requests.

Access Token format

This token uses the JSON Web Token (JWT, https://tools.ietf.org/html/rfc7519) format. The Access Token MUST be signed and MAY be encrypted, using sign-then-encrypt (as per OAuth2 https://tools.ietf.org/html/rfc7519#section-11.2 recommended default). If the Access Token is encrypted, it MUST be encrypted to the DV-system / Client, using a key and encryption algorithm registered for the `client_id` (see OIDC Registration).

> ⚠ In case a "clusteraansluiting" is applicable, the Access Token is encrypted to the DV-system / Client's certificate. NOTE: the "cluster" system's certificate MUST NOT be used and therefore the "cluster" private key cannot be used for decryption, usage of the DV's private key is explicitly required.

Access Token contents

The JWT Access Token has the following contents:

key	0...1	description	comment	example value	documentation references
iss	1	Issuer of this token. MUST be included.	The URI identifying the issuing RD / OP.	`"https://rd.example.com"`	https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.2.1
azp	1	The client ID of the client to whom this token was issued. MUST be included.	Client ID as issued during registration.		https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.2.1
exp	1	The expiration time of the token. MUST be included.			https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.2.1
jti	1	Identifier for this Access Token. MUST be included.	This identifier will differ from the identifier (`jti`) in the ID Token, as both are distinct different messages.		https://tools.ietf.org/html/rfc7519#section-4.1.7 https://openid.net/specs/openid-igov-oauth2-1_0-03.

					html#rfc.section.3.2.1
sub	0..1	An identifier of the User involved in this interaction by the DV-system / Client.	See OIDC Identifier types and formats.		https://tools.ietf.org/html/rfc7519#section-4.1.2 https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.2.1
urn:nl-gdi-eid:1.0:oidc:sub_type	0..1	The type of subject identifier, as provided in `sub`. In case a service is registered for a variable subject identifier type use case, this member MUST be used to establish the effective subject type provided. MAY be present in other cases.	See OIDC Identifier types and formats.	`"urn:nl-eid-gdi:1.0:id:BSN"`	Custom to this specification.
aud	~~0..n~~ 1..n	An array of identifiers of the audience(s), for which this access token is intended. The Client who requested the authentication/authorization MUST be included, identified by its client_id. Additional audiences MAY be included. In case the array has a single value, it MAY be encoded as string rather than an array.	An additional audience is typically a Resource Server.		https://tools.ietf.org/html/rfc7519#section-4.1.3 https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.2.1
scope	1..n	An array of scope identifiers, specifying the scope(s) applicable to this access token. MUST be included.	Service identifiers are defined as UUID, prefixed using `urn:nl-gdi-services:` to make it a URI.	`urn:nl-gdi-services:ea0eb2b6-9cf5-4a6e-99de-e41da799eb8b`	(Partially / related) https://tools.ietf.org/html/rfc7523#section-2.1 https://tools.ietf.org/html/rfc7521#section-4.1 https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.4.1
cnf	0..1	In case a Client acts as a DienstBemiddelaar (DB), MUST be included. The confirmation claim (`cnf`) specifies the public key (certificate) to be used by the Client to prove its identity to any audience it is using this access token.	See under Confirmation claim.	`{ "jwk": { "x5c": "MIIFdDCC..." } }`	https://tools.ietf.org/html/rfc7800#section-3.1

Confirmation claim

A confirmation claim is a claim that contains information that enables confirmation that the presenter of a (bearer) token is the legitimate presenter, as intended by the issuer of that token. Confirmation claims support the concept of "proof of possession" of a secret or private key, to assert the presenter is legitimate by substantiating the claim through proof created using a cryptographic key. This concept is also known as "holder of key". Confirmation claims are specified in RFC7800 (https://tools.ietf.org/html/rfc7800).

The Confirmation claim has the following contents:

key	0...1	description	comment	example value	documentation references

jwk	0...1	In case a Client acts as a DienstBemiddelaar (DB), MUST be included, binding the containing token to the public key / PKIo certificate of the client. The `jwk` MUST contain a `x5c` with the DB-systems/Clients X.509 PKIoverheid certificate and MUST NOT contain other elements.		`{ "x5c":`
				`"MIIFdDCC..."`
				`}`
					https://tools.ietf.org/html/rfc7800#section-3.2
					https://tools.ietf.org/html/rfc7517

ⓘ Note: the exact method of confirmation of the proof of possession is out of scope of these specifications.

Access token processing

A sending RD / OP:

- MUST sign the JWT access token, using a key as listed for the RD / OP in the OIDC Provider metadata.
- MAY encrypt (sign-then-encrypt) the access token; in case the Access Token is encrypted, it MUST be encrypted to the Client, using a key registered for encryption for the Client.
- MUST include a `jti`, with a value that is unique during at least 12 months and an entropy equivalent to at least 128-bit of cryptographic random (as per https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.3.2.1).
- In case of Dienstbemiddeling, MUST include a `cnf` containing a `jwk` with a `x5c` containing the DB's PKIo certificate.

A receiving DV-system / Client:

- MUST NOT process or use a received access token after `expires_in` seconds have elapsed since receiving that access token .
- MUST NOT process or use a received access token after the time defined by the value of `exp` in an access token.
- MUST verify it is the authorized party to use the access token through the `azp` parameter and it is an intended audience through the `aud` parameter and MUST NOT process or use a received access token if itself is not listed as the authorized party and as an audience.
- MUST verify the values for the applicable scope(s) are present, based on the scope(s) in the request, the service(s) referenced thereby, for which consent was granted by the user and for which the user is authorized.

- MUST not use access tokens for other purposes then those covered in the provided scope(s), as only consent and authorization for these are applicable.
- In case one or more "divisible" Service Instance Set(s) or referenced Service Definition Set(s) were requested: MUST inspect the supplied scope(s) and handle access decision based on the value(s) for scope provided, these may provide only a subset of the requested scope(s). See Usage of datamodel Services for further detail.

A relying Resource Server:

- MUST NOT process or use a received access token after the time defined by the value of `exp` in an access token.
- MUST verify that it is an intended audience through the `aud` parameter and MUST NOT process or use a received access token if itself is not listed as an audience.
- MUST verify the values for the applicable scope(s) are present in the access token relied upon and MUST not grant access to services for other purposes then those covered in the provided scope(s), as only consent and authorization for these are applicable.
- In case one or more "divisible" Service Instance Set(s) or referenced Service Definition Set(s) are applicable: MUST inspect the supplied scope(s) and handle access decision based on the actual value(s) for the `scope` provided, these may provide only a subset of a "divisible" Service Set referenced by the scope(s). See Usage of datamodel Services for further detail.
- MUST process and validate a `cnf` claim, if present in a received access token.
- In case the `cnf` contains a `jwk`/`x5c`, MUST validate the presenter of the token is authenticated using the PKIoverheid certificate stated in the `x5c`.
    - RECOMMENDED is authentication using mutual authenticated TLS, OPTIONALLY extended using message authentication. Alternative client authentication methods MAY be used, but MUST offer security at least equivalent to that of mutual authenticated TLS.

OIDC ID Token

The ID Token included in the OIDC Token Response, contains the actual identity/authentication information. This can be used by the DV-system / Client for its access control decision and as information for processing the contents of subsequent requests.

Token format

The ID Token uses the JSON Web Token (JWT, https://tools.ietf.org/html/rfc7519) format. The ID Token MUST be signed and MAY be encrypted, using sign-then-encrypt (as per OIDC, https://openid.net/specs/openid-connect-core-1_0.html#IDToken). The ID Token MUST be signed by the issuing RD / OP using a valid key as listed in (the JWS key set in) the RD / OP metadata (see OIDC Provider metadata). In case the ID Token is encrypted, it MUST be encrypted to the Client, using a key and encryption algorithm registered for that `client_id` (see OIDC Client registration Request).

> ⚠ In case a "clusteraansluiting" is applicable, the ID Token is encrypted to the DV-system / Client's certificate. NOTE: the "cluster" system's certificate MUST NOT be used and therefore the "cluster" private key cannot be used for decryption, using the DV's private key is necessary.

Token contents

The ID Token has the following contents:

> ⓘ Delegation - Representation related contents are subject to change
>
> Please note that delegation and representation related topics are not finalized and parts of this specification that cover delegation and representation are more likely to change than other parts.

key	0..1	description	comment	example value	documentation references
nonce	1	MUST be present. Nonce, to be used to prevent replay/CSRF/XSRF/etc... attacks.	Verbatim copy of nonce from Authentication / Authorization Request		https://openid.net/specs/openid-connect-core-1_0.html#IDToken
at_hash	0..1	Optional. Access Token hash value.	Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the `access_token` value of the related access token in the OIDC Token Response. The hash (digest) algorithm MUST use the digest used in the ID Token's signature, as per OIDC.		https://openid.net/specs/openid-connect-core-1_0.html#CodeIDToken
iss	1	MUST be present. Identifier of the Issuer of this Token.	MUST be the RD / OP Issuer Identifier URL.	`"https://rd.example.nl/oidc"`	https://openid.net/specs/openid-connect-core-1_0.html#IDToken
sub	1	MUST be present. Identifier of the subject authenticated / authorized.	See OIDC Identifier types and formats.		https://openid.net/specs/openid-connect-core-1_0.

		In case of "anonymous" access, that is *only using attributes* for the access decision, a random `sub` that only has meaning for the scope and duration in the context of the initial Authentication / Authorization Request. That is, the `sub` is non-persistent and it is extremely unlikely a user with the same `sub` ever to return.			html#IDToken
urn:nl-gdi-eid:1.0:oidc:sub_type	0..1	The type of subject identifier, as provided in `sub`. In case a service is registered for a variable subject identifier type use case, this member MUST be used to establish the effective subject type provided. MAY be present in other cases.	See OIDC Identifier types and formats.	`"urn:nl-eid-gdi:1.0:id:BSN"`	Custom to this specification.
aud	1..n	MUST be present. Audience(s) for this ID token, list of identifiers of parties (by default client_ids) intended /allowed to consume this ID Token. In case the array has a single value, it MAY be encoded as string rather than an array.	= client_id of requesting DV-system / Client. Others to be included included in specific use cases (TODO: format and use cases to be clarified. ?).		https://openid.net/specs/openid-connect-core-1_0.html#IDToken
azp	0..n	Authorized Party, list of identifiers (client_id / URI) of parties allowed to present this ID Token.	optional by default left empty, as per OIDC; only present when there is more than one Audience and/or the Authorized party(s) are not the same as the Audience.		https://openid.net/specs/openid-connect-core-1_0.html#IDToken
amr	**0**..1	MUST NOT be present.	This specification uses `acr` over `amr`, by choice. The authentication context class reference (acr) includes relevant business rules, the authentication method reference (`amr`) is more specific to a method. The method may vary over time and does not include other necessities, such as the enrollment process. See https://tools.ietf.org/html/rfc8176#section-3.		https://openid.net/specs/openid-connect-core-1_0.html#IDToken https://tools.ietf.org/html/rfc8176#section-3
auth_time	0..1	Time of authentication /authorization, optional.	Time in seconds since the epoch (1/1/1970 UTC).	`1311280969`	https://openid.net/specs/openid-

		MUST be present when client registered using "require_auth_time = 1".	In case of representation, this time will hold the time of the using time of the last/most recent obtained declaration of authorization information.		connect-core-1_0.html#IDToken
exp	1	MUST be present. Expiration time of this ID token.	Time in seconds since the epoch (1/1/1970 UTC).	`1311280980`	https://openid.net/specs/openid-connect-core-1_0.html#IDToken
iat	1	MUST be present. Time this ID token is issued.	Time in seconds since the epoch (1/1/1970 UTC).	`1311280970`	https://openid.net/specs/openid-connect-core-1_0.html#IDToken
nbf	0..**1**	MUST be present. Not before, ID token is not to be used before this time.	Although not before (`nbf`) is not used or specified in OpenID Connect itself, it is specified in JWT and required for the ID Token in the iGov profile.		https://openid.net/specs/openid-igov-openid-connect-1_0-03.html#rfc.section.3.1 https://tools.ietf.org/html/rfc7519#section-4.1.5
acr	1	MUST be present. Authentication context class reference, containing the applicable Level of Assurance, using the eIDAS specified values.	The Level of Assurance of the authentication. In case of representation, this values will contain the minimum of the chain of authentication and authorization(s).	`"http://eidas.europa.eu/LoA/high"`	https://openid.net/specs/openid-connect-core-1_0.html#IDToken
vot	**0**..1	MUST NOT be present. Vector of trust.	As eIDAS-based Level of Assurance (LoA) values are used as `acr`, vectors of trust are not applicable.		https://openid.net/specs/openid-igov-openid-connect-1_0-03.html#rfc.section.3.1
vtm	**0**..1	MUST NOT be present. Vector of trust trustmark.	As eIDAS-based Level of Assurance (LoA) values are used as `acr`, vectors of trust are not applicable.		https://openid.net/specs/openid-igov-openid-connect-1_0-03.html#rfc.section.3.1
jti	1	MUST be present. Identifier for this ID Token.	Value MUST be unique during at least 12 months.	*uuid*	https://openid.net/specs/openid-igov-openid-connect-1_0-03.html#rfc.section.3.1

					https://tools.ietf.org/html/rfc7519#section-4.1.7

act	0..1	Optional. Mutually exclusive with `may_act`. In case of representation, this field will hold details of the acting subject.	In case of Representation, the `act` claim MUST contain information about the acting subject. An act claim MAY contain claims normally included in the ID Token (e.g. `sub`, `name`, etc...). Claims containing non-subject related information have no meaning and SHOULD NOT be included (e.g. `exp`, `azp` and `nonce`). A chain of delegation can be expressed by nesting one `act` claim within another (i.e. recursively).
			Specific meaning for the following fields when used within an `act` field applies:
			• iss – the issuer of the authorization information.
			• acr – the level of assurance applicable to the registration of the authorization information.
				`{"sub": "abc....",`
				`"iss": "http://mr.example.nl/",`
				`"name": "John"}`
					https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-14#section-4.1

may_act	0..1	Optional. Mutually exclusive with `act`. In case an authorization (*machtiging*) is used, this field will hold details of the represented person.	Inverse of `act`. When may_act is present, the user may act on behalf of the subject described under `may_act`. Specific meaning for the following fields when used within an `act` field applies: • iss – the issuer of the authorization information. • acr – the level of assurance applicable to the registration of the authorization information.		https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-14#section-4.4
~~*urn:nl-gdi-eid:1.0:oidc:db*~~	~~0..1~~	~~In case of Service Intermediation, MUST be included to DB, MUST NOT be included to DA.~~ ~~In other cases, MUST NOT be included.~~	~~See Below.~~		
cnf	0..1				https://tools.ietf.

		Optional. Reserved for future updates to these specifications.			org/html/draft-ietf-oauth-token-binding-08#section-3.1
attr_name	0..*n*	Attribute *`attr_name`* if available. An attibute is available if it has been (validated and) registered and the user provided consent. TODO: Encrypted using indirect (aggregated) notation ❓?	See OIDC Attribute catalog for available attribute Claim names and their semantics.		

ID Token processing

A sending RD / OP:

- MUST sign the ID Token, using the ID token signing algorithm (`id_token_signed_response_alg`) registered for the DV-system / Client, using a key as listed for the RD / OP in the OIDC Provider metadata.
- MAY encrypt (sign-then-encrypt) the ID Token; in case the ID Token is encrypted, it MUST be encrypted to the Client using the key and ID token encryption algorithm(s) (`id_token_encrypted_response_alg` and `id_token_encrypted_response_enc)` registered for the DV-system / Client`.`
- MUST set 'auth_time' in ID Token in case client has been registered with `require_auth_time` set to true.
- MUST ensure the identifier for an ID Token (`jti`) is unique across all ID Tokens and clients – across all DVs and RDs – during at least 12 months.
- MUST set the Level of Assurance obtained/established for the user Authentication / Authorization as Authentication Context Class Reference (`acr`). Combined LoA's result in the minimum LoA of each individual part of the Authentication / Authorization.
- in case attribute claims, to be provided in the ID Token, were requested:
  - MUST request consent from the user, and
  - MUST provide the available attributes, with available being a (validated,) registered and consented attribute.
  - in case attributes are not available, MUST provide an ID token omitting any non-available attribute (and not returning an null or empty (string) valued attribute, in line with OIDC https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse).
- in case of "anonymous" access request:
  - MUST provide an arbitrary random opaque value as `sub` with highly improbable chance of reappearing.
  - MUST provide requested and consented attributes as attribute claims after authentication at the requested Level of Assurance.
- In case a service is registered for a variable subject identifier type use case, the nl_eid_sub_type MUST be used to establish the effective subject type provided. The `sub` can be interpreted accordingly.
- MUST set the value of the `iat` parameter to the time of issuance of the ID Token.
- MUST set the value of the `nbf` parameter to a time no earlier than the time of obtaining authentication/consent of the user and no later than the time of issuance of the ID Token.
- MUST set the value of the `exp` parameter to a maximum of *<id-token.max-validity>* after the time in the value of the `nbf` parameter.
- TODO ...
- ...

A receiving DV-system / Client:

- MUST verify the signature to authenticate the message as originating in unmodified form from a trusted RD / OP.
- If the ID token contains an at_hash claim, MUST verify the at_hash with the access token, as per OIDC core §3. 1.3.8.
- MUST validate the ID Token before processing and further use.
- MUST verify that itself is listed as an Audience ('aud') in the ID Token before processing and further use.
- MUST validate the nonce matches the nonce sent in the Authentication / Authorization Request and belongs to the session with the user-agent of the subject for whom that Authentication / Authorization Request was initiated.
- MUST process and validate a `cnf` claim, if present in a received token.
- SHOULD check the ID token for Claims requested to be provided under the ID Token. In case Essential Claims were requested but are not provided (or have an unacceptable value), the RD / OP MUST inform the user the service cannot be provided with the reason why and SHOULD inform the user on how to continue.
- TODO azp
- SHOULD verify the Level of Assurance received in the ID Token ('act') is at least the Level of Assurance requested.
- SHOULD process an `act`/`may_act` claim in an ID Token in order to recognize representation is applicable and handle requests accordingly.
- MAY use information in `act`/`may_act` claim in an ID Token to accommodate the acting, representing user with improved user experience.
- MUST not accept, use, process or trust an ID token:
    - if the time set in `iat` is in the future or too long in the past (stale token),
    - if the time set in `nbf` is in the future or too long in the past (stale token),
    - if time set in `exp` is in the past. Existing sessions with the user-agent of the user MUST be terminated or re-authenticated after `exp`.
- TODO
- ~~TODO rules for DB;~~ included below under service intermediation
- ~~TODO fules for DA;~~ included below under service intermediation
- ...

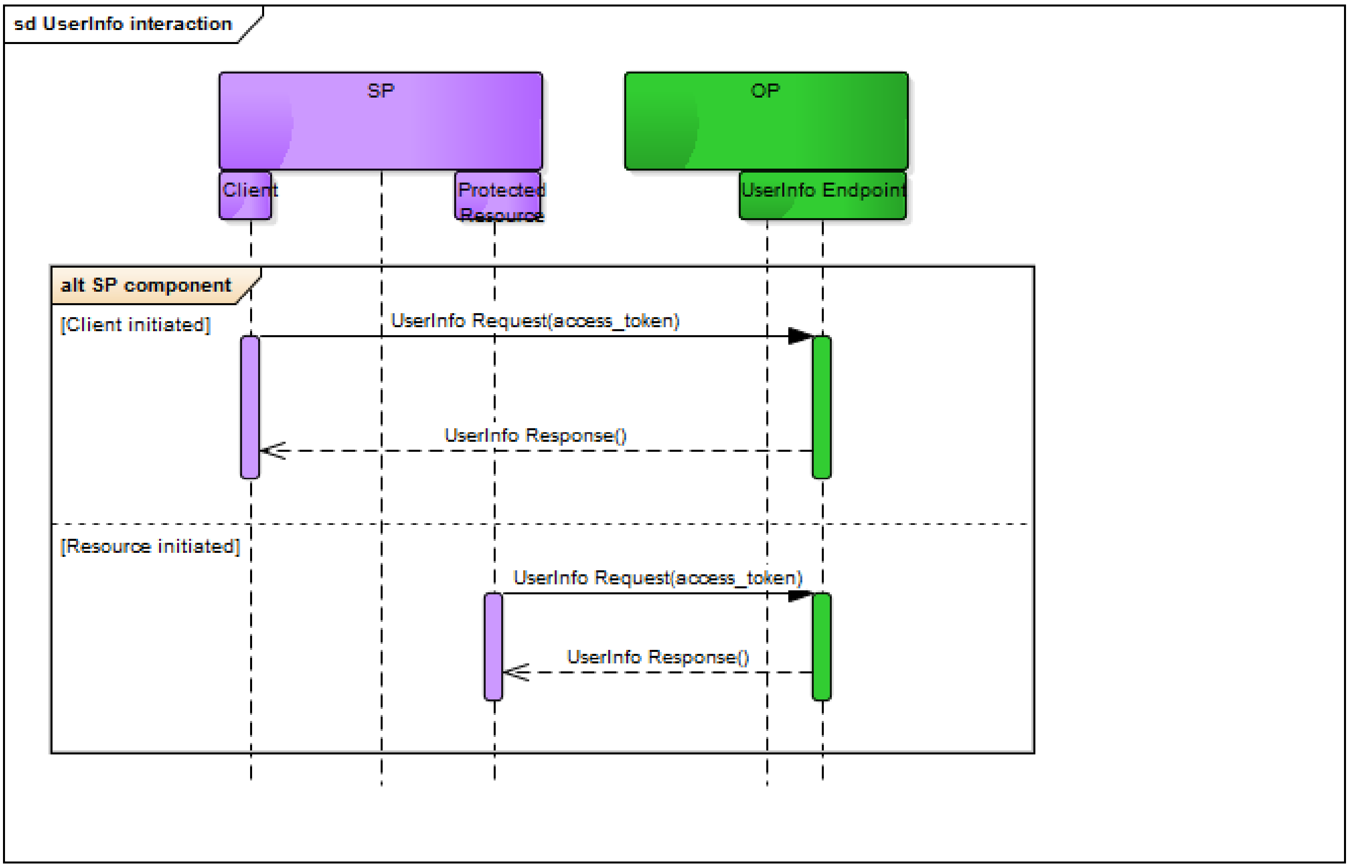
Open issues:

- how sub with EI/EP fit within 255 limit? ?

OIDC Userinfo interaction

If Claims (attributes) were requested to be provided via UserInfo, this interaction is required to actually obtain these Claims (attributes).

In case of a native application Client, the information regarding the User MUST be acquired using a UserInfo request. In other cases, this MAY be used and this interaction is optional.



The DV-system / Client initiates the interaction by requesting the UserInfo at the RD OP, with a OIDC Token Request using the `access_token` received with the OIDC Token Response. The RD / OP responds on its UserInfo Endpoint with a OIDC UserInfo Response.

> ⓘ Note that the Resource Server role of the SP MAY initiate the interaction as well, rather than the Client role. Initiating the OIDC Userinfo interaction from the Client role is RECOMMENDED.

Interaction transport

This is a synchronous request over HTTPS by the DV-system / Client (including the Resource Server of the DV) to the RD / OP's UserInfo Endpoint.

Interaction authentication

Authentication of the requesting DV-system / Client/Resource Server is not applicable, as per OIDC specifications. The request MUST be made using the HTTP Authorization Request Header field containing the OIDC Access Token for authorization, see RFC 6750 §2.1 (https://tools.ietf.org/html/rfc6750#section-2.1).

ⓘ Although RFC6750 specifies form-encoded and URI query as methods for passing the access token, both methods are not supported in this specification.

As the request is sent over HTTPS, the responding server (OP UserInfo Endpoint) is always authenticated using it's PKIoverheid certificate.

OIDC UserInfo Request

The request is made by the DV-system / Client over HTTPS to the RD / OP's UserInfo Endpoint. The UserInfo Endpoint can be found in the OIDC Provider metadata.

Request encoding

The request is either HTTP GET or HTTP POST encoded (see OIDC Transport interoperability and security requirements). As per OIDC specification, using HTTP GET is RECOMMENDED and the `Authorization` header MUST be used.

Request contents

The UserInfo request has no parameters other than the Access Token in the `Authorization` header.

A non-normative UserInfo request example:

```
GET /userinfo-endpoint HTTP/1.1
Host: openidprovider.example.com
Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6Imp3dGF0c2lnbmluZyJ9.eyJzY29wZSI6Im9wZW5p
 ZCBuYW1lIiwiY2xpZW50X2lkIjoicGtqd3RfMSIsImp0aSI6IjJ3eFFUYjY2aWZkTEc3NGJzaUdtVzlhZzRSZ3B1YmV
 lIiwiaXNzIjoiaHR0cHM6Ly9vcGVuaWRwcm92aWRlci5leGFtcGxlLmNvbTo5MDMxIiwiYXVkIjoiaHR0cHM6Ly9yZX
 NvdXJjZXNlcnZlci5leGFtcGxlLmNvbSIsInN1YiI6InRlc3QyIiwiYXpwIjoicGtqd3RfMSIsImV4cCI6MTU0NDYyN
 zE1OX0.ROxqA2dIUuYBzJ1P7By4RqRx-GxbBwdZOvF4V1xXfnQ1-cr-zK0ktAJBZGQthzQ1PmyBrx7F_in0Cyf01_WD
 _1SwUsHEt5jVGKfeSlvvyRxNHVBnwPeWOl97vlqynYiNi3QUgg32T0qwtVUl_Xf8QxFl7led1-eWPMN_yoRJPSgpBHp
 mGpMfHat7lHl2TlR_t7hXsPHBIjCulKJk7swK_Fh1OR3ChG7ppvRPygqA1sz_IDQjcqi1PClaxLObnhgRdThpCBe3YY
 F84rP1-iCOjnRfygGkPu8Qv0JXA5UqQ7sxtOP4YQrxQdGnSsTVI94p52M_qODQwWF1opz82RIPyw
```

Where decoding the above access token gives:

```
{
  "alg": "RS256",
  "kid": "jwtatsigning"
}
{
  "scope": "openid name",
  "client_id": "pkjwt_1",
  "jti": "2wxQTb66ifdLG74bsiGmW9ag4Rgpubee",
  "iss": "https://openidprovider.example.com:9031",
  "aud": "https://resourceserver.example.com",
  "sub": "test2",
  "azp": "pkjwt_1",
  "exp": 1544627159
}
```

Request processing

Sending DV-system / Client processing rules:

- SHOULD only request UserInfo if the corresponding OIDC Authentication / Authorization Request did contain any Claim (attribute) request under the `userinfo`.
- SHOULD request the UserInfo using an access token obtained at the Token Endpoint only once.
- TODO

Receiving RD / OP processing rules:

- MUST validate the access token provided in the `Authorization` header was issued by the receiving RD / OP.
- MUST reject requests if the access token was issued more than *<access-token.max-validity>* ago.
- MAY reject requests if the corresponding OIDC Authentication / Authorization Request did not contain any request to provide a Claim under the `userinfo`.
- ~~SHOULD only respond to each unique access token once and MAY reject any second request for UserInfo using an access token.~~
- TODO

OIDC UserInfo Response

The UserInfo is a synchronuous response by the RD / OP to the UserInfo Request message of the DV-system / Client, delivered over HTTPS.

Response encoding

The response contains the UserInfo parameters encoded as a JSON object which is signed and optionally encrypted, resulting in a JSON Web Token (JWT, https://tools.ietf.org/html/rfc7519). The UserInfo MUST be signed and MAY be encrypted, using sign-then-encrypt (as per OIDC, https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse). The provided UserInfo MUST be signed by the issuing RD / OP using a valid key as listed in (the JWS key set in) the RD / OP metadata (see OIDC Provider metadata). In case the UserInfo is encrypted, it MUST be encrypted to the DV-system / Client, using a key and encryption algorithm registered for that `client_id` (see OIDC Client registration Request).

The resulting JWT is provided as the HTTP Body of the response, as per https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse. The `Content-Type` is defined as `application/jwt`.

Response contents

This message can contain Claims, see "Provided Claims" under "ID Token".

This mesage has the following contents:

key	0..1	description	comment	example value	documentation references
sub	1	MUST be provided, containing the identifier of the subject authenticated / authorized and MUST have the exact same value as the corresponding ID Token.	See OIDC Identifier types and formats.		https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse
urn:nl-gdi-eid:1.0:oidc:sub_type	0..1	The type of subject identifier, as provided in `sub`. In case a service is registered for a variable subject identifier type use case, this member MUST be used to establish the effective subject type provided. MAY be present in other cases.	See OIDC Identifier types and formats.	"urn:nl-eid-gdi:1.0:id:BSN"	Custom to this specification.
iss	1	Identifier of the Issuer of this UserInfo Token.	MUST be the RD / OP Issuer Identifier URL.	`https://rd.example.nl/oidc`	https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse
aud	0..1	SHOULD be included, TODO			https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse
attr_name	0..*n*	MUST contain the value of the attribute, only in case the attribute is provided/available. If an attribute can not be provided, either because it is unavailable or because a user did	See OIDC Attribute catalog for available attribute claim	`"family_name": "Doe"`	https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse

		not provide consent, the attribute MUST NOT be provided.	names and their semantics.		

Response processing

Sending RD / OP processing rules:

- MUST sign the provided UserInfo, using the UserInfo signing algorithm (`userinfo_signed_response_alg`) registered for the applicable DV-system / Client, using a key as listed for the RD / OP in the OIDC Provider metadata.
- MAY encrypt (sign-then-encrypt) the provided UserInfo; in case the UserInfo is encrypted, MUST encrypting using the key and UserInfo encryption algorithm(s) (`userinfo_encrypted_response_alg` and `userinfo_encrypted_response_enc`) registered for the DV-system / Client to whom the access token used with the UserInfo Request was issued.
- MUST NOT provide attributes for which the user did not provide consent to do so.
- MUST provide only the available attributes, with available being a (validated) registered and consented attribute.
- In case attributes are not available, MUST provide a UserInfo omitting any non-available attribute (and not returning a null or empty (string) valued attribute, as per OIDC https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse).
- MUST only provide attributes as Claims under the UserInfo for Claims which were requested in the corresponding OIDC Authentication / Authorization Request under the `userinfo`.
- MUST provide the exact same value for `sub` as was provided in the corresponding ID Token. In case of "anonymous" access, that means the same random value for `sub` as used in the corresponding ID Token MUST be used.
- MUST provide the `aud` using ... TODO.

Receiving DV-system / Client processing rules:

- MUST validate the message as per OIDC, see https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponseValidation and MUST validate the signature on the UserInfo (rather than SHOULD as specified in OIDC).
- MUST validate the UserInfo `sub` matches the ID Token `sub`, as per https://openid.net/specs/openid-connect-core-1_0.html#UserInfoResponse.
- SHOULD check the UserInfo for Claims requested to be provided under the UserInfo. In case Essential Claims were requested but are not provided (or have an unacceptable value), the RD / OP MUST inform the user the service cannot be provided with the reason why and SHOULD inform the user on how to continue.
- TODO

Error handling

In case a RD / OP cannot process a (request) message or has to reject a request, an error response is sent instead of a response. An error response message is identical to the error response as defined in OIDC (https://openid.net/specs/openid-connect-core-1_0.html#UserInfoError) and OAuth2 (https://tools.ietf.org/html/rfc6750#section-3).

Effectively, in case a request or Access Token used in the request is invalid, the UserInfo Endpoint MUST respond with a HTTP 401 (Unauthorized) error. This HTTP error MUST include a `WWW-Authenticate` header, with an `error` parameter, and optionally an `error_description` and `error_uri` parameter. The error codes that can be used as value for the `error` parameter are defined in https://tools.ietf.org/html/rfc6750#section-3.1.

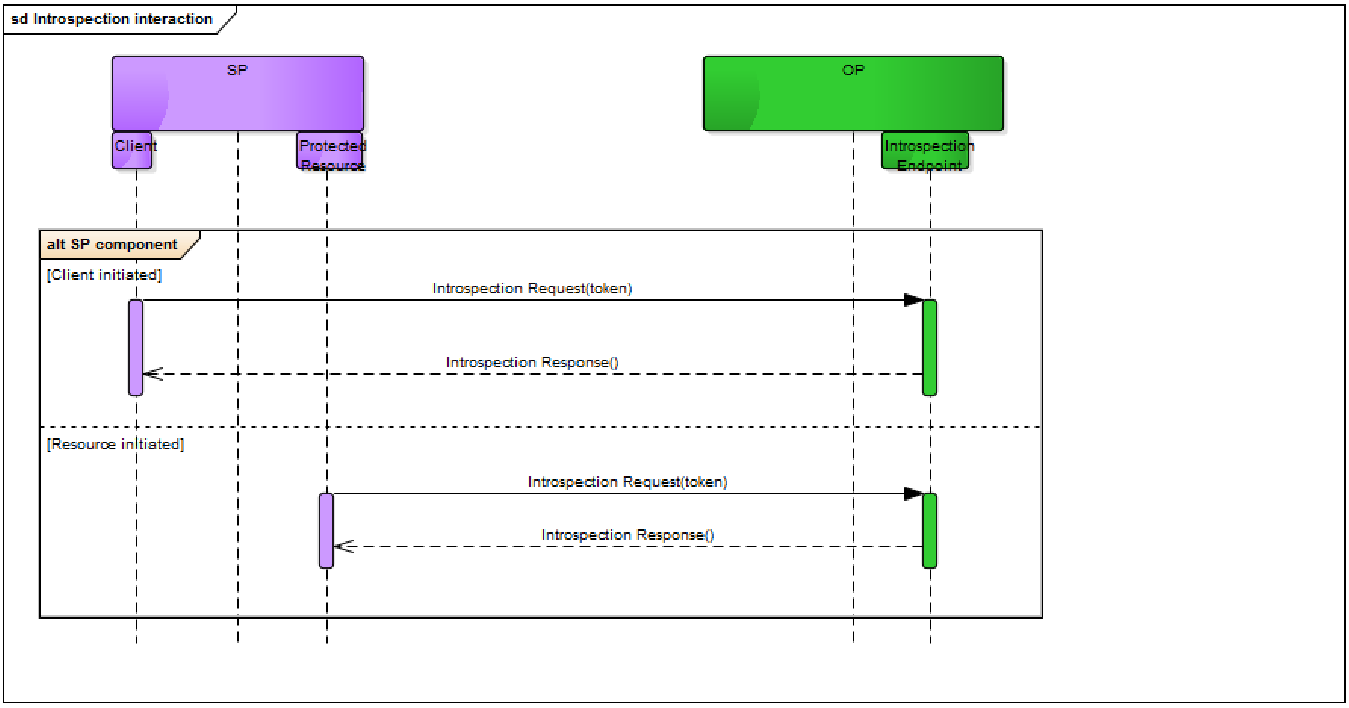
Error processing rules

- In case a DV-system / Client applies logic for handling specific errors, the value of the `error` parameter SHOULD be used as input for this logic.

Introspection interaction

A DV-system / Client or a Resource Server MAY inquire more information, such as current validity status of the token or associated data, from the RD / OP about an Access Token, using Token Introspection. Token Introspection is defined in https://tools.ietf.org/html/rfc7662, with signed response being specified in https://tools.ietf.org/html/draft-ietf-oauth-jwt-introspection-response-01.

A DV-system / Client or Resource Server initiates the interaction, with an Introspection Request using an `access_token` received in an OIDC Token Response or in a resource request from a DV-system / Client respectively, and the RD / OP responds on its Interaction Endpoint with an Introspection Response.



ⓘ Note: Initiating the Introspection interaction from the Resource Server role is the default and RECOMMENDED usage of the Introspection interaction.

Interaction transport

This is a synchronous request over HTTPS by the Resource Server or DV-system / Client to the RD / OP's Introspection Endpoint.

Interaction authentication

A requesting Resource Server / DV-system / Client MUST be authenticated, using one of the methods defined in OIDC Client Authentication (also see OIDC Client registration Request).

ⓘ

Note: Identifier and authentication details for the Resource Server and DV-system / Client role can be different. Ensure that the applicable credentials are being used.

As the request is sent over HTTPS, the server is always authenticated using its PKIoverheid certificate.

Introspection Request

An introspection request is sent by a Resource Server or a DV-system / Client over HTTPS to the RD / OP's Introspection Endpoint. The Introspection Endpoint can be found in the OIDC Provider metadata.

Request encoding

The request MUST be sent using form encoding (`application/x-www-form-urlencoded` data) in a HTTP POST request, as specified in https://tools.ietf.org/html/rfc7662#section-2.1.

Request contents

This message has the following contents

param	0..1	description	comment	example value	reference
token	1	The access token to be introspected. MUST be included.			https://tools.ietf.org/html/rfc7662#section-2.1
token_type_hint	0..1	SHOULD be present. Valid values are either `access_token` or `refresh_token`, depending on the type of token. Defaults to `access_token`.		`access_token`	https://tools.ietf.org/html/rfc7662#section-2.1
client_id	0..1	depends on client authentication • MUST be included when client authentication using `tls_client_auth` is utilized • MAY be included in other cases	Note: When using a combination of client authentication methods including `tls_client_auth` – most noteworthy the 'clusteraansluiting' – the client_id MUST be included.		https://tools.ietf.org/html/draft-ietf-oauth-mtls-13#section-2
client_assertion_type	0..1	MUST be included when client authentication using `private_key_jwt` is utilized		`urn:ietf:params:oauth:client-assertion-type:jwt-bearer`	https://tools.ietf.org/html/rfc7522#section-2.2
client_assertion	0..1	MUST be included when client authentication using `private_key_jwt` is utilized	Assertion MUST be a signed JWT using HMAC using client_secret or signed using private key, depending on registered client auth method and algorithm		https://tools.ietf.org/html/rfc7522#section-2.2

Request processing rules

Sending DV-system / Client or Resource Server processing rules:

- MUST authenticate as DV-system / Client or Resource Server, by using the registered OIDC Client Authentication method.
- TODO

Receiving RD / OP processing rules:

- MUST authenticate the requester as DV-system / Client or Resource Server, by verifying the requester through the registered OIDC Client Authentication method.
- MUST validate the `token` was issued by the RD / OP itself, by verifying the issuer and signature in the presented `token`.
- MUST verify the requester is an audience of the presented `token` before responding to the request.
- SHOULD use the `token_type_hint` to validate and interpretation of the `token` provided.

Introspection Response

The Token Introspection Response is a synchronous answer to the Introspection Request.

Response encoding

The response contains the Introspection response parameters encoded as a JSON object.

In case the response message is sent without a signature and/or encryption, the JSON object is the HTTP Body of the response. The `Content-Type` is defined as `application/json`.

In case the response message is signed (and/or encrypted), the JSON object is contained in a JWT, a signed (then encrypted) JSON Web Token (JWT, https://tools.ietf.org/html/rfc7519) as the HTTP Body of the response, as per https://tools.ietf.org/html/draft-ietf-oauth-jwt-introspection-response-01#section-3. The `Content-Type` is defined as `application/jwt`.

Response contents

This message has the following contents

param	0..1	description	comment	example value	reference
iss	1	MUST be present. Issuer of the inspected token.	The identifier of the OP.		https://tools.ietf.org/html/draft-ietf-oauth-jwt-introspection-response-01#section-3 https://tools.ietf.org/html/rfc7662#section-2.2
aud	0..1	In case the Introspection Response is signed: MUST be present, audience of the inspection response. In case the Introspection Response is not signed: OPTIONAL, audience of the introspected token.	TODO: To be clarified. There is an inconsistency between the definition of aud in the two linked RFCs.		https://tools.ietf.org/html/draft-ietf-oauth-jwt-introspection-response-01#section-3 https://tools.ietf.org/html/rfc7662#section-2.2
active	1	MUST be present. Boolean indicator of whether or not the presented token is currently active.		`true`	https://tools.ietf.org/html/rfc7662#section-2.2
scope	0..n				

		A space-separated list of scopes associated with the inspected token.			
client_id	0..1	The client_id of the Client that requested the inspected token.			
username	0	MUST NOT be used.	Although specified in OAuth, not supported for privacy reasons.		https://tools.ietf.org/html/rfc7662#section-2.2
token_type	0..1	Type of the inspected token.			
exp	0..1	Expiration time of the inspected token.			https://tools.ietf.org/html/rfc7519#section-4.1.4
iat	0..1	Issuance time the inspected token.			https://tools.ietf.org/html/rfc7519#section-4.1.6
nbf	0..1	Token not valid before of the inspected token.			https://tools.ietf.org/html/rfc7519#section-4.1.5
sub	0..1	Subject (user identity) of the inspected token.	See OIDC Identifier types and formats.		https://tools.ietf.org/html/rfc7519#section-4.1.2
urn:nl-gdi-eid:1.0:oidc:sub_type	0..1	The type of subject identifier, as provided in `sub`. In case a service is registered for a variable subject identifier type use case, this member MUST be used to establish the effective subject type provided. MAY be present in other cases.	See OIDC Identifier types and formats.	`"urn:nl-eid-gdi:1.0:id:BSN"`	Custom to this specification.
jti	**0**..1	SHOULD NOT be used. Identifier of the token.	As RFC 7662 in combination with https://tools.ietf.org/html/draft-ietf-oauth-jwt-introspection-response-01#section-3 can be ambiguous, no identifier for the token is to be included.		https://tools.ietf.org/html/rfc7519#section-4.1.7

Response processing

Sending RD / OP processing rules:

- In case the `introspection_signed_response_alg` for the requesting DV-system / Client / Resource Server is registered as other than `none`, the response MUST be signed using that algorithm.
- MAY encrypt the Introspection Response; in case the `introspection_encrypted_response_alg` / `introspection_encrypted_response_enc` for the requesting DV-system / Client / Resource Server is registered as other than `none`, the response MUST be encrypted using the registered algorithm(s) and key.
- TODO

Receiving DV-system / Client or Resource Server processing rules:

- TODO

Resource Server interaction

Depending on the DV's use case(s), the DV-system / Client MAY interact with Resource Server(s), using the Access Token received from the Token Endpoint.

In case the Resource Server is operated under responsibility of the DV itself, this is out of scope of this specification and the specifications below MAY be applied.

In case the Resource Server is operated by a third party (even if both parties have their systems hosted by the same entity), this is a use case of "Dienstbemiddeling" and the specification below MUST be applied.

A requesting DV-system / Client:

- MUST only use an Access Token during its validity, that is MUST NOT use an Access Token after its expiration date and time.
- SHOULD send the Access token as `Authorization` HTTP header, as per https://tools.ietf.org/html/rfc6750#section-2.1; MAY use form encoded requests (https://tools.ietf.org/html/rfc6750#section-2.2) instead; MUST NOT use uri-encoded requests (https://tools.ietf.org/html/rfc6750#section-2.3)
- In case the access token has a `cnf` parameter, MUST use authentication using the referenced key/method.

A receiving Resource Server:

- MUST apply mutual-authenticated TLS as per this specification or an equivalent for transport/message security (e.g. WS-Security with encryption *and* signatures, IPSec with appropriate choice of endpoints and secure configuration, ...).
- SHOULD receive the Access token as `Authorization` HTTP header, as per https://tools.ietf.org/html/rfc6750#section-2.1; MAY use form encoded requests (https://tools.ietf.org/html/rfc6750#section-2.2) instead; MUST NOT use uri-encoded requests (https://tools.ietf.org/html/rfc6750#section-2.3)
- MUST validate an Access Token to be authentic and originating from a RD / OP by verifying the signature.
- MUST validate and process an Access Token as specified under the processing rules for a relying Resource Server under OIDC Access Token.
- MUST validate an Access Token, ... TODO ...
- In case the access token has a `cnf` parameter, MUST authenticate the requester using the referenced key/method.
- MAY include additional criteria for authorization in its access decision; these are beyond the scope of this specification.
- TODO MUST apply https://openid.net/specs/openid-igov-oauth2-1_0-03.html#rfc.section.4.3

TODO add requirements and recommended default implementation/reference architecture.

Error handling

In case a Resource Server cannot process a (request) message or has to reject a request, an error response is sent instead of a response. An error response message is identical to the error response as defined in OAuth2 Bearer Token Usage (https://tools.ietf.org/html/rfc6750#section-3).

Effectively, in case an Access Token used in the request is invalid or access is denied, the Resource Server MUST respond with a HTTP 401 (Unauthorized) error. This HTTP error MUST include a `WWW-Authenticate` header, with an `error` parameter, and optionally an `error_description` and `error_uri` parameter. The error codes that can be used as value for the `error` parameter are defined in https://tools.ietf.org/html/rfc6750#section-3.1.

Error processing rules

- In case a DV-system / Client applies logic for handling specific errors, the value of the `error` parameter SHOULD be used as input for this logic.

OIDC Provider metadata

The RD / OP server MUST publish metadata about its service(s), in order to facilitate DV-system / Client to automatically process this data and its changes so that registration and configuration of the DV-system / Client can be automated as much as possible. The DV-system / Client SHOULD use the RD / OP metadata for auto-configuration and registration.

ⓘ Changes to OP configuration

RD / OP SHOULD NOT make uncoordinated breaking (e.g backwards incompatible) changes that may affect existing integrations.

RD / OP SHOULD NOT assume that all clients are capable of automatically detecting changes that may affect them.

RD / OP SHOULD establish a coordinated update method, which is out of the scope of this specification.

Metadata transport

The form of this metadata is as per OIDC Discovery (https://openid.net/specs/openid-connect-discovery-1_0.html) and OAuth2 Server Metadata (RFC8414, https://tools.ietf.org/html/rfc8414) specifications. The metadata of the RD / OP server can be requested using a HTTP GET request at a well-known endpoint ("https://rd.example.nl/.well-known/openid-configuration" and/or "https://rd.example.nl/.well-known/oauth-authorization-server").

The metadata is published in JSON format, and the response MUST have the `application/json Content-Type`, as per OIDC Discovery / RFC8414.

Auto-discovery based on the web finger protocol MUST NOT be supported or used, for privacy concerns among other reasons. Hosting or discovery of multiple issuers on a single domain MUST NOT be supported.

TODO add time for caching/refreshing OP metadata...

Metadata Contents

The RD / OP metadata has the following content:

key	0..n	description	comment	example
issuer	1	issuer ID of the RD / OP, MUST be the issuer URL		"https://
authorization_endpoint	1	authorization endpoint of the RD / OP, to be used to send the Authentication / Authorization Requests to		"https://
token_endpoint	1		Required.	"https://

		Endpoint to send the Token Request to, MUST be a HTTPS URL		
userinfo_endpoint	0..1	Endpoint to send the UserInfo Request to, MUST be a HTTPS URL		"https://rd.ex
jwks_uri	1	URL of RD / OP key(s), used for signing / encryption of OIDC messages	See under OIDC JWKS.	`"https://`
signed_jwks_uri	0..1	TODO		
registration_endpoint	0..1	URL where DV-system / Client can register with this RD / OP		`"https://`
scopes_supported	1..n	Scopes supported, MUST include "openid".	Does not have to include all supported scopes; ❓ support stating client registered service scopes, using TLS mutual auth on well-known OIDC discovery endpoint?	`["openid"`
response_types_supported	n	Response types supported.	Only authorization code flow supported in this profile.	`["code"]`

response_modes_supported	0..n	❓ `query` or `form_post`	The mode `fragment` MUST NOT be used.	
grant_types_supported	0..n	Authorization grant types support.	Only authorization code flow supported in this profile. Although this is considered a Dynamic OpenID provider, the "implicit" flow is not supported.	`["authori`

acr_values_supported	**0..n**	Authentication Context Class References (LoA's) supported. MUST include eIDAS substantial and eIDAS high, MAY include eIDAS low.		`["http://`
				`/substant`
				`eu/LoA/hi`

subject_types_supported	0..n	❓ map pairwise and public to "pseudo" versus "VI"? In addition to OIDC, the value `transient` MUST be supported for use cases of "anonymous" attribute provisioning.		

		In addition to OIDC, the value `variable` MUST be supported for use cases of variable subject identifier type.		
id_token_signing_alg_values_supported	0..n	Algorithm(s) supported for signing ID Tokens.	See signing messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["PS256",`
id_token_encryption_alg_values_supported	0..n	Algorithm(s) supported to encrypt the key used to encrypt (content encryption key, CEK) the ID Tokens.	See encrypting messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["RSA-OAE`
id_token_encryption_enc_values_supported	0..n	Algorithm(s) supported to encrypt the content in ID Tokens.	See encrypting messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["A256GCM`
userinfo_signing_alg_values_supported	0..n	Algorithm(s) supported for signing Claims returned from the UserInfo endpoint.	See signing messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["PS256",`
userinfo_encryption_alg_values_supported	0..n	Algorithm(s) supported to encrypt the key used to encrypt (content encryption key, CEK) the Claims returned from the UserInfo endpoint.	See encrypting messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["RSA-OAE`
userinfo_encryption_enc_values_supported	0..n	Algorithm(s) supported to encrypt the content in Claims returned from the UserInfo endpoint..	See encrypting messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["A256GCM`
request_object_signing_alg_values_supported	0..n	Algorithm(s) supported for request signing.	See signing messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["PS256",`
request_object_encryption_alg_values_supported	0..n	Algorithm(s) supported for	See encrypting messages in OIDC	`["RSA-OAE`

		encryption of the key used to encrypt (content encryption key, CEK) request objects.	Message notation and OIDC Algorithms for supported algorithms.	
request_object_encryption_enc_values_supported	0..n	Algorithm(s) supported for encryption of request objects.	See encrypting messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	["A256GCM
token_endpoint_auth_methods_supported	0..n	Methods of authentication supported for sending Token Requests to the Token Endpoint.	See client authentication methods in OIDC Client Authentication.	["tls_cli
token_endpoint_auth_signing_alg_values_supported	0..n	Algorithm(s) supported for signing in the client authentication method used when sending Token Requests to the Token Endpoint. MUST be included if the `private_key_jwt` authentication method is supported.	JSON array containing a list of the JWS signing algorithms. See signing messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	["PS256",
display_values_supported	0..n	List of the display parameter values that the RD / OP supports. The value `wap` MUST NOT be used.		["page",
claim_types_supported	0..n	List of the Claim Types that the RD / OP supports.	? distributed schrappen?	["normal" "distribu
claims_supported	0..n	List of the Claim Names of the Claims that the RD / OP MAY be able to supply values for.	See OIDC Attribute catalog for available claim names and their semantics.	["18+", "
service_documentation	0..n	URL to documentation of the RD / OP.		

claims_locales_supported	0..n	Languages supported by the RD / OP for claims.	At least Dutch (`nl`) MUST be supported.	`["nl"]`
ui_locales_supported	0..n	Languages supported by the RD / OP in the UI.	At least Dutch (`nl`) MUST be supported.	`["nl", "e`
claims_parameter_supported	0..1	Boolean value specifying whether the RD / OP supports use of the claims parameter.	Optional.	`true`
request_parameter_supported	0..1	Boolean value specifying whether the OP supports use of the request parameter.	RD / OP MUST support usage of the request parameter.	`true`
request_uri_parameter_supported	0..1	Boolean value specifying whether the OP supports use of the request_uri parameter.	Optional.	`true`
require_request_uri_registration	0..1	Boolean value specifying whether the OP requires any request_uri values used to be pre-registered using the request_uris registration parameter.	Optional.	`true`
op_policy_uri	0..n	URL to the policy of the RD / OP		`"https://`
op_tos_uri	0..n	URL to terms and conditions of the RD / OP		`"https://`

code_challenge_methods_supported	0..n	Code challenge (PKCE) methods supported. Only `S256` to be supported.	The method `plain` MUST NOT be supported.	`["S256"]`
signed_metadata	0..1	Signed version of the same metadata.	The `signed_metadata` is explicitly not included in the signed metadata itself.	*`base64.ba`*
introspection_endpoint	0..1	Endpoint for introspection of access tokens issued by this RD / OP.		
introspection_endpoint_auth_methods_supported	0..1	Methods of authentication supported for sending Introspection Requests by the Introspection Endpoint.		
introspection_endpoint_auth_signing_alg_values_supported	0..1	Algorithm(s) supported for signing in the client authentication method used when sending Introspection Requests to the Introspection Endpoint. MUST be included if the `private_key_jwt` authentication method is supported.	JSON array containing a list of the JWS signing algorithms. See signing messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	`["PS256",`
introspection_signing_alg_values_supported	0..1	Algorithm(s) supported for signing Introspection Responses returned from the Introspection Endpoint.	See signing messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	
introspection_encryption_alg_values_supported	0..1	Algorithm(s)		

		supported to encrypt the key used to encrypt (content encryption key, CEK) the Introspection Responses returned from the Introspection Endpoint.	See encrypting messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	
introspection_encryption_enc_values_supported	0..1	Algorithm(s) supported to encrypt the content in Introspection Responses returned from the Introspection Endpoint.	See encrypting messages in OIDC Message notation and OIDC Algorithms for supported algorithms.	
revocation_endpoint	0..1	MUST NOT be used in this profile.		
revocation_endpoint_auth_methods_supported	0..1	MUST NOT be used in this profile.		
revocation_endpoint_auth_signing_alg_values_supported	0..1	MUST NOT be used in this profile.		
~~as_access_token_token_binding_supported~~	0..1	candidate for future use		
~~as_refresh_token_token_binding_supported~~	0..1	candidate for future use		

ⓘ See OIDC Algorithms for supported algorithms. The RD / OP MUST support all listed algorithms.

Metadata processing rules

RD / OP server processing rules:

- MUST publish its metadata
- MUST publish only via HTTPS, using a PKIoverheid certificate for the server
- SHOULD include the `signed_metadata`, with all values equal to the values in the "plain" metadata, with the exception of the signed_metadata parameter.
- MUST not advertise insecure nor unsupported options (even if they are required by generic OIDC conformance)
- MAY require DV-system / Client authentication for providing its metadata, and MAY adapt the content to contain DV-system / Client specific settings.

Relying Party / DV-system / Client processing rules:

- SHOULD retrieve the RD / OP metadata, to obtain valid metadata from the RD / OP
    - Alternatively, the information MAY be obtained from the RV Regieorgansiatie / RD Beheerorganisatie. The remaining rules below still apply.
- MUST ensure the metadata is retrieved from the verified trusted RD(s) / OP(s), only use metadata acquired from a validated and secure HTTPS connection using a trustworthy location (URL).
- SHOULD validate the signature over `signed_metadata` before processing the metadata (? if metadata published in signed form).
    - In case metadata values appear in both the `signed_metdata` and the containing unsigned document, metadata values conveyed in the signed metadata MUST take precedence over the corresponding values conveyed using plain JSON elements.
- MUST only use the endpoints advertised in RD / OP metadata for interactions with the RD / OP, e.g for the OIDC Token interaction.
- MUST only use options as advertised in the metadata for communicating with RD / OP.

OIDC Registration

⚠ The specifications regarding (dynamic client) Registration are currently not yet complete.

OIDC Registered service data model

TODO redundant, see Datamodel Services

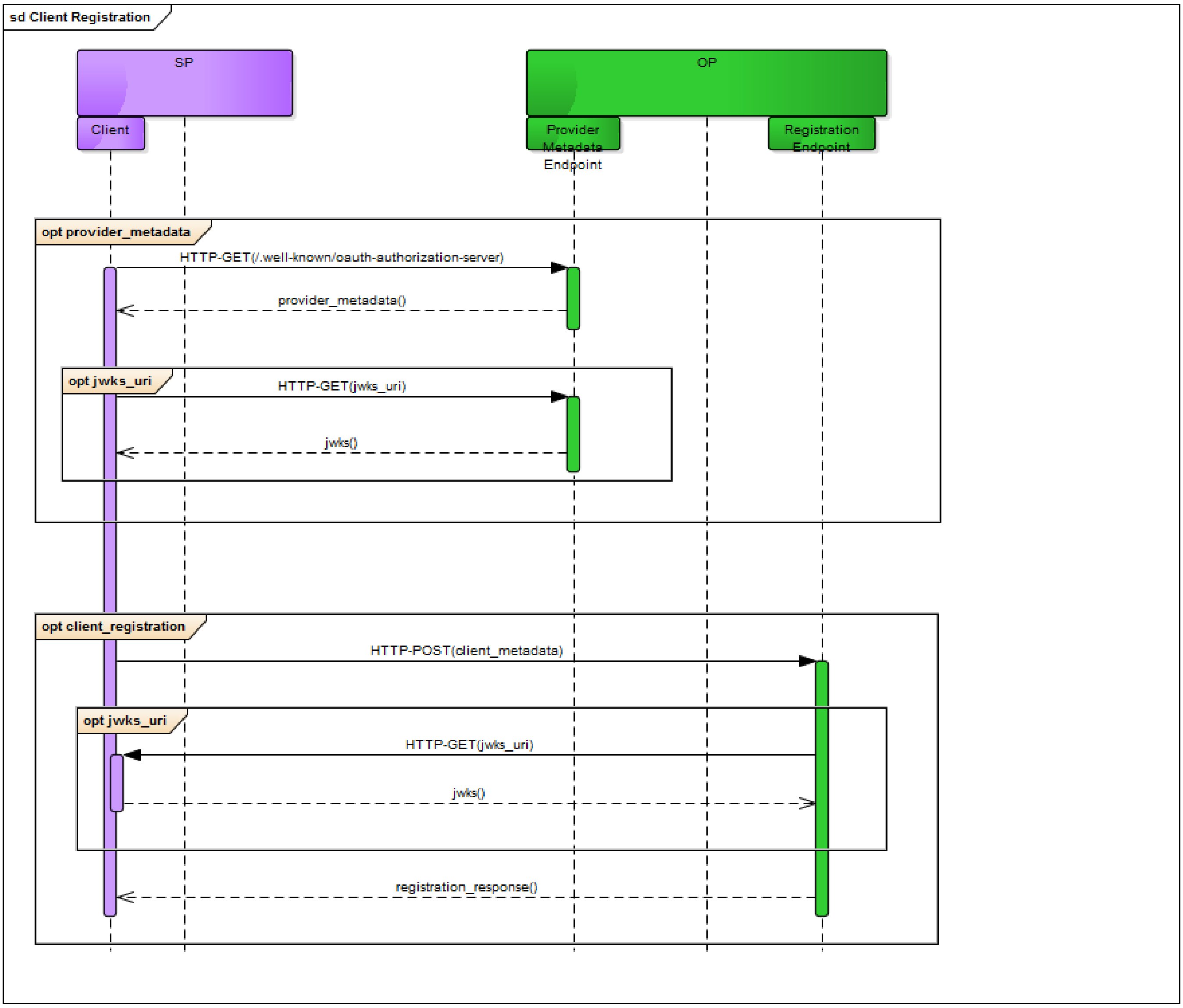
Service MUST be registered prior to being referenced for authentication/authorization.

Important concepts used elsewhere:

ServiceUUID – a UUID assigned that is persistent and both globally and temporal unique. To be used as reference that defines a service for functional and legal contexts (thereby for instance for consent, in tokens, etc...).

ServiceMnemonicURI – an absolute URI for a service, . To be used to technically reference a (instance of a) service, MUST be resolved/resolvable to a ServiceUUID for functional processing.

OIDC Registration interaction



TODO

OIDC Client registration interaction

Each DV-system / client needs to be known by the RD / OP, before requests can be answered. Therefore each DV-system / client MUST register certain data – called client metadata – in order to ensure proper working of the technical interfaces as well as for security reasons.

TODO Client should request OP metadata OIDC Provider metadata before registering, use metadata to select applicable options for registration.

Request transport

For registration, each DV-system / client SHOULD register with the RD / OP using OIDC Dynamic Registration as described below. Alternatively, the client MAY use a self-registration portal ? to register the applicable information.

Prior to dynamic registration, each organisation MUST register itself in order to establish eligibility, align administrative details, etc... This can be done using the TODO Regieorganisatie "proces aansluiten".

In case a "clusteraansluiting" is applicable, each DV-system / Client MUST register as a separate, individual client. That is, each relying party (DV) that is legally a distinct entity, MUST register as a unique client. A relying party MAY register as multiple clients, for legal, organisational or technical reasons at their own discretion.

In case a "clusteraansluiting" is applicable, a `software_statement` MUST be included containing the details of the "cluster" system. See below for more information.

Registration using OIDC dynamic registration is done using an OIDC Client Registration Request (https://openid.net/specs/openid-connect-registration-1_0.html#RegistrationRequest). A Client Registration Request MUST contain the client's metadata (signed ? not mentioned in spec). The Client Registration Request is to be send to the RD / OP using synchronous HTTP POST request to the registration_endpoint of the RD / OP as listed in the RD / OP Metadata.

TODO once per RD / OP, details unique per RD / OP ?.

TODO registering scope per service ?.

TODO Client MUST authenticate during registration. Options ?: Mutual TLS or OAuth2/OIDC token.

OIDC Client registration Request

Request transport

TODO ref interaction.

Request contents

The Client Registration Request and contained metadata has the following contents:

key	0..n	description	comment
redirect_uris	1..n	Array of redirect endpoints, endpoints that will be used in regular authentication flow by this DV-system / client.	only registered endpo MUST NOT be accepte
response_types	0..n	optional; array of response types expected by client	defaults per OIDC to '
grant_types	0..n	optional; array of grant types expected by the client.	default is authorizati (additionally) allowe
application_type	0..1	Kind of the application. The default, if omitted, is web. The defined values are `native` or `web`.	
contacts	0..n	optional; array of string, containing email-addresses of the system administrators responsible for this DV-system / client	
client_name	0..**n**	Name of the DV or DV-system / client ⚠ . Multi-lingual allowed, by using the client_name#locale notation.	❓ allow usage to regi allow PKIo/KvK regist TBD: which detail lev
logo_uri	**0**..n	Logo of the DV or DV-service, to be shown for users on pages for IDP selection or authentication .	Multi-lingual allowed
client_uri	0..**n**	URI where the DV-system / client can be reached. MAY be presented to users, for instance when registering an authorization (*machtiging*) or upon authentication for informing the user as anti-phishing measure.	Multi-lingual allowed
policy_uri	0..**n**	URL to privacy policy of DV. MUST be provided when	Multi-lingual allowed

		using attributes. Will be presented to the user upon authentication when consent is requested.	
tos_uri	0..n	URL to the terms of service of the DV / DV-system / client. Will be presented to the user upon authentication when consent is requested.	Multi-lingual allowec
jwks_uri	0..1	URL to JWKS key(Set), for usage with authentication for, signing or encrypted of OIDC messages. Either jwks_uri or jwks (mutually exclusive) MUST be supplied, usage of jwks_uri is RECOMMENDED.	The URI MUST link ba key set can be retrieve See below under JWK Certificates included the identity of the rel contain the OIN of th Note: In case a "clust the "cluster" MUST be
jwks	0..1	Same as above, containing the key(s) value(s) in JWKS format rather than the URI to he JWKS. Either jwks_uri or jwks (mutually exclusive) MUST be supplied.	Certificates included the identity of the rel contain the OIN of th Note: In case a "clust the "cluster" MUST be
sector_identifier_uri	0..1	MAY be used.	
		The (non OIDC compliant) identifier `urn:nl-eid-gdi:1.0:id:BSN` MAY be used to request a user's BSN. This MUST only be used in combination with a `subject_type` value of `public`.	Alternatively, a `sect` resolving to a JSON fi `eid-gdi:1.0:id` `redirect_uris` i
		The (non OIDC compliant) identifier `urn:nl-eid-gdi:1.0:id:Pseudonym` MAY be used to request a relying party specific pseudonym of the subject. This MUST only be used in combination with a `subject_type` value of `pairwise`.	Alternatively, a `sect` resolving to a JSON fi `eid-gdi:1.0:id` `redirect_uris` i
		The (non OIDC compliant) identifier `urn:nl-eid-gdi:1.0:id:Transient` MAY be used to indicate "anonymous" requests will be made for use cases where only attribute claims are requested. This MUST only be used in combination with a `subject_type` value of `transient`.	Alternatively, a `sect` resolving to a JSON fi `eid-gdi:1.0:id` `redirect_uris` i

		In case a `subject_type` with a value `variable` is requested, MUST be provided. The `sector_identifier_uri` MUST contain a HTTPS URL resolving to a JSON file containing an array with the reference for the applicable subject types and the exact values registered for `redirect_uris` in this message. Valid references for subject types are `urn:nl-eid-gdi:1.0:id:BSN`, `urn:nl-eid-gdi:1.0:id:Pseudonym` and `urn:nl-eid-gdi:1.0:id:Transient`, at least two MUST be included.	Other types may be a
subject_type	0..1	MAY be provided to set the requested type of identifier • In case `public` is requested, this is considered as a request for the BSN by default. Other forms of request can be made using the `sector_identifier_uri`. • In case `pairwise` is requested, this is considered a request for a relying party specific pseudoniem by default. Other forms of request can be made using the `sector_identifier_uri`. • In addition to OIDC, the value `transient` MAY be used for use cases of "anonymous" attribute provisioning. • In addition to OIDC, the value `variable` MAY be used for use cases with variable subject identifier type. In this case a `sector_identifier_uri` MUST be present.	The `subject_typ` and only specific com
id_token_signed_response_alg	0..1	Signing algorithm required for receiving signed ID Tokens. If omitted, the default for the RD / OP will be used instead (currently: `PS256`).	Only an algorithm lis `id_token_signi`
id_token_encrypted_response_alg	0..1	Encryption algorithm required to be used to encrypt the key used to encrypt (content encryption key, CEK) the ID Token. If omitted, the default for the RD / OP will be used instead (currently: `RSA-OAEP-256`).	Only an algorithm lis `id_token_encry` used. The strongest a
id_token_encrypted_response_enc	0..1	Encryption algorithm required to be used to encrypt the ID Token. If omitted, the default for the RD / OP will be used instead (currently: `A256GCM`). When `id_token_encrypted_response_enc` is included, `id_token_encrypted_response_alg` MUST also be provided.	Only an algorithm lis `id_token_encry` used. The strongest a
userinfo_signed_response_alg	0..1		

		Signing algorithm required for receiving signed Claims returned from the UserInfo endpoint. If omitted, the default for the RD / OP will be used instead (currently: `PS256`).	Only an algorithm lis `userinfo_signi`
userinfo_encrypted_response_alg	0..1	Encryption algorithm required to be used to encrypt the key used to encrypt (content encryption key, CEK) the Claims returned from the UserInfo endpointn. If omitted, the default for the RD / OP will be used instead (currently: `RSA-OAEP-256`).	Only an algorithm lis `_encryption_al`
userinfo_encrypted_response_enc	0..1	Encryption algorithm required to be used to encrypt the Claims returned from the UserInfo endpoint. If omitted, the default for the RD / OP will be used instead (currently: `A256GCM`). When `id_token_encrypted_response_enc` is included, `id_token_encrypted_response_alg` MUST also be provided.	Only an algorithm lis `_encryption_en`
request_object_signing_alg	0..1	Signing algorithm the DV-system / client will use for signing request objects. If omitted, the default will be used instead (currently: `PS256`).	Only an algorithm lis `request_object` be used.
request_object_encryption_alg	0..1	Encryption algorithm the DV-system / client will use to encrypt the key used to encrypt (content encryption key, CEK) the request object. The default, if omitted, is that any algorithm supported by the RD / OP and the DV-system / client MAY be used.	Only an algorithm lis `request_object` MUST be used.
request_object_encryption_enc	0..1	Encryption algorithm the DV-system / client will use to encrypt the request object. When `request_object_encryption_enc` is included, `request_object_encryption_alg` MUST also be provided.	Only an algorithm lis `request_object` MUST be used. ? default? OIDC = ur
token_endpoint_auth_method	0..1	Authentication this DV-system / client will use for 'Token Request' messages send to the RD / OP token endpoint. Possible values: • `tls_client_auth` MAY be used • `private_key_jwt` MAY be used • `client_secret_jwt` MUST NOT be used • `client_secret_basic` MUST NOT be used • `client_secret_post` MUST NOT be used • `none` MUST NOT be used In case of a "clusteraansluiting", the method `private_key_jwt` MUST be used.	Only a method listed `token_endpoint` In case of a "clusteraa authenticated using r `private_key_jw` TLS authentication of statement.
token_endpoint_auth_signing_alg	0..1	Signature algorithm this DV-system / client will use for client authentication to the 'Token Request'	Only a method listed `token_endpoint` MUST be used.

		endpoint. MUST be present when `token_endpoint_auth_method` is `private_key_jwt`.	
introspection_signed_response_alg	0..1	Signing algorithm required for receiving Token Introspection Responses. If omitted, the default will be used instead (currently: `PS256`).	
introspection_encrypted_response_alg	0..1	Encryption algorithm required to be used to encrypt the key used to encrypt (content encryption key, CEK) the Introspection Response. If omitted, the default for the RD / OP will be used instead (currently: `RSA-OAEP-256`).	
introspection_encrypted_response_enc	0..1	Encryption algorithm required to be used to encrypt the Introspection Response. If omitted, the default for the RD / OP will be used instead (currently: `A256GCM`). When `introspection_encrypted_response_alg` is included, `introspection_encrypted_response_enc` MUST also be provided.	
default_max_age	**0**..1	Optional, will be ignored.	Optional, will be ignc assurance rules.
require_auth_time	0..1	Boolean value specifying whether the auth_time Claim in the ID Token is REQUIRED by the DV-system / client. Default value is `false`.	
default_acr_values	0..**1** ..n	Array of ACR (=LoA) value(s) requested by this DV-system / client. Only a single value MUST be used, indicating the minimum requested LoA for this service.	Possible values are th • `http://eida` • `http://eida` • `http://eida`
initiate_login_uri	**0**..1	MUST NOT be used. Unsolicited authentication is not to be used for security reasons.	
request_uris	0..n	Array of URIs used for passing a request by reference.	URI either hosted by OP.
tls_client_auth_subject_dn	0..1	Subject DN of the client PKIo certificate used by this DV-system / client when registering `tls_client_auth` as authentication method.	The full subject DN o representation, accor
software_statement	0..1	In case a "clusteraansluiting" is applicable: MUST be present, containing the identity, details and key(s) /certificate(s) of the "cluster". In other cases: MAY be used.	See below.

~~client_access_token_token_binding_supported~~	0..1	candidate for future use	
~~client_refresh_token_token_binding_supported~~	0..1	candidate for future use	
urn:nl-gdi-eid:1.0:oidc:requested_claims	0..1	MAY be used by the client to pre-register claim names that will be requested. The requested_claims is a JSON object containing the requested attribute names, with as value either null (default) or a JSON object with per requested attribute the optional members: • `essential`, (as under Requested Claims in OIDC Authentication / Authorization Request / individual claims OIDC 5.5.1), default false • `value`, (as under Requested Claims in OIDC Authentication / Authorization Request / individual claims OIDC 5.5.1), default unspecified • `values`, optional, (as under Requested Claims in OIDC Authentication / Authorization Request / individual claims OIDC 5.5.1), default unspecified • `purpose`, specifying a purpose ("*doelbinding*") for the requesting the attribute • `purpose#lang`, specifying a purpose in the specified language	Requested attributes registered for each re not pre-register via th pre-registered using t

Request processing

Registering DV-system / client processing rules:

- MUST register a `tls_client_auth_subject_dn` when registering `tls_client_auth` as authentication method.
- MUST register for a "transient" subject identifier with at least one required requested attribute, for use cases where only attribute claims are requested. Such registration MUST be made using either the value `transient` for the `subject_type` or (*not* both) using a `sector_identifier_uri` referring to `urn:nl-eid-gdi:oidc:transient`.

Receiving RD / OP processing rules:

- MUST validate the request before processing
- MUST authenticate the DV-system / client before registering the client.
- MUST register the values for the client and validate other OIDC messages according to these registered values.
- MUST check the *Autorisatielijst BSN* (see Autorisatielijst BSN) before registering a DV-system / client that requests a BSN as subject.
- if present, MUST check the values and combinations of `subject_type` and `sector_identifier_uri` to establish the subject type to register for this service.
- MUST ignore `default_max_age` if present in client registration request / client metadata.

Open issues:

- ⚠ Details in distinction between registering a system and registering for a specific individual service (set).
- ❓ What restrictions on name registration by DV-system / client?

OIDC Software statement

A statement containing details of the software used, MAY be included in the registration request as per https://tools.ietf.org/html/rfc7591#section-3.1.1. Note that the `software_statement` MUST be included in case a "clusteraansluiting" is applicable.

The `software_statement` MUST be in the form of signed JWT, with signature using at least RS256. The signature MUST be created by the issuer of the `software_statement` (i.e. typically not the relying party / DV client being registered). The issuer and signer SHOULD typically be:

- developer statement - the organization that developed the software; the developer's certificate (x5c) is used for the signature.
- hoster statement - the organisation that hosts (shared) software; the hoster's certificate is used for the signature. This situation applies in case a "cluseraansluiting" is applicable.

The contents of the `software_statement` is as follows:

key	0..n	description	comment	example	spec
iss	1	URI identifying the issuer of this software statement.		`"https://a.cloud"`	https://tools.ietf.org/html/rfc7591#section-2.3
software_id	0..1	RECOMMENDED. Unique Identifier of the specific software used. The identifier applies for either the software used (developer statement) or the hosting system (hoster statement).	The value of this field is not intended to be human readable and is usually opaque. RFC5791 The "software_id" SHOULD remain the same across multiple updates or versions of the same piece of software.	`"489d8aec-a982-44cb-af37-30efca52db4b"`	https://tools.ietf.org/html/rfc7591#section-2.3
software_version	0..1	A version identifier string for the client software identified by "software_id". The value of the "software_version" SHOULD change on any update to the client software identified by the same "software_id".	The value of this field is intended to be compared using string equality matching and no other comparison semantics are defined by this specification.	`"1.0"`	https://tools.ietf.org/html/rfc7591#section-2.3
client_name	0..1	RECOMMENDED. Human readable name for this software or system.	TODO: optionally multi-lingual (e.g. `client_name#nl`)	`"Example Software"`	https://tools.ietf.org/html/rfc7591#section-2
tls_client_auth_subject_dn	0..1	In case the software describe a hosting "cluster", the SubjectDN of	Note: The TLS client authentication is used to authenticate the cluster; the		

		TLS certificate(s) used for client authentication by TLS for the "cluster" in the context of this profile MUST be included. The SubjectDN MUST be the string representation as per RFC4514 §2. In other cases: MUST NOT be used.	relying party / DV-client is authenticated using the signature on request objects. Therefor, the SubjectDN of the "cluster's" TLS client certificate MUST be registered in the case a "clusteraansluiting" is applicable.		https://tools.ietf.org/html/draft-ietf-oauth-mtls-09#section-2.1.2 https://tools.ietf.org/html/rfc4514#section-2
software_statement	0..1	In case the software described in this statement is hosted, the developer's software_statement MAY be included in another embedded software_statement with the same structure. The embedded software_statement MUST be signed and included using a JWS representation.		*base64.base64.base46*	https://tools.ietf.org/html/rfc7591#section-2.3
...	0..n	Other claims MAY be included. It is RECOMMENDED to use only claims described in https://tools.ietf.org/html/rfc7591#section-2.	Other claims will be ignored, but MAY be used or prohibited in future versions of this profile.		

TODO: rules

- A "cluster" hosting multiple relying party / DV client systems, MUST include the same software_statement (same `iss` and `software_id`) for all DV-system / Clients hosted on the same system. The "cluster" MUST be registered as such with the RV regieorganisatie, prior to client registration.

OIDC Client registration Response

Synchronous response from RD / OP to the registering DV-system / client.

Response transport

HTTP 201 as per OIDC dynamic client registration.

Response contents

The client receives a response with the following contents:

key	0..n	description	comment	example	spec
client_id	1	Client_id assigned to the registering DV-system / client by the RD / OP.	Must be unique, to be used in subsequent Authentication / Authorization Requests and related messages.		https://openid.net/specs/openid-connect-registration-1_0.html#RegistrationResponse https://tools.ietf.org/html/rfc7591#section-3.2.1
client_secret	0..1	Client secret to use, for client authentication, if applicable.	? only use when not mutual-TLS or private_key_jwt (non-symmetric)? or combined as well?		https://openid.net/specs/openid-connect-registration-1_0.html#RegistrationResponse https://tools.ietf.org/html/rfc7591#section-3.2.1
registration_access_token	0..1	Registration Access Token that can be used at the Client Configuration Endpoint to perform subsequent operations upon the Client registration.			https://openid.net/specs/openid-connect-registration-1_0.html#RegistrationResponse https://tools.ietf.org/html/rfc7592#section-2
registration_client_uri	0..1	Location of the Client Configuration Endpoint where the Registration Access Token can be used to perform subsequent operations upon the resulting Client registration. Implementations MUST either return both a Client Configuration Endpoint and a Registration Access Token or neither of them.			https://openid.net/specs/openid-connect-registration-1_0.html#RegistrationResponse https://tools.ietf.org/html/rfc7592#section-2
client_id_issued_at	0..1	Time at which the Client			

		Identifier was issued.			https://openid.net/specs/openid-connect-registration-1_0.html#RegistrationResponse https://tools.ietf.org/html/rfc7591#section-3.2.1
client_secret_expires_at	0..1	REQUIRED if client_secret is issued. Time at which the client_secret will expire ~~or 0 if it will not expire~~.	Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.		https://openid.net/specs/openid-connect-registration-1_0.html#RegistrationResponse https://tools.ietf.org/html/rfc7591#section-3.2.1

Response processing

Responding RD/ OP processing rules:

- MUST issue a unique `client_id`, that is both globally and temporal unique.
- MUST provide a client secret when registering a client with (token endpoint) client authentication method of `client_secret_jwt`~~, `client_secret_basic` or `client_secret_post`~~ and MUST not be provided otherwise.
  - MUST generate a new, cryptographically secure random secret of sufficient length for the applicable algorithms per registering client when providing a `client_secret`.
  - MUST provide a client_secret_expiration_at date, with a maximum expiration of 1 year in the future.

Receiving DV-system / client processing rules:

- MUST use the value of the `client_id` and (if applicable) the value of the `client_secret` in subsequent OIDC request messages.
- MUST ensure a received `client_secret` is stored and handled securely.
- MUST maintain an up-to-date registration with the RD / OP at `registration_client_uri`.
- MUST use the `registration_access_token` for subsequent operations upon the Client registration.
- MUST renew the client secret before `client_secret_expires_at.`

Open issues:

- ❓ Use registration_access_token for client management, TLS mutual auth, others or allow both registration_access_token and mutual-TLS?
- ❓ Always expire client secret (max X age)?

OIDC Appendix - settings

Some parts of these interface specification contain variable or configurable items, that do not structurally change the interface or its specifications. The underlying sections define the applicable values for these variable items, and may be subject to change separate from changes to these interface specifications.

OIDC Attribute catalog

Attributes defined in this profile:

param	description	comment	example value	documentation references
name	The name (full name) attribute of the subject as per OIDC. Typically the name is used for displaying purposes.	⚠ OIDC allows titles to be included in the name as well, which will not in this profile as these are not part of the eIDAS minimal data set.	`"Jan van Modaal"`	https://openid.net/specs/openid-connect-core-1_0.html#StandardClaims
given_name	The given name(s) attribute of the subject as per OIDC.	⚠ As not all identity provider will have all given names of a user, the name may be incomplete or contain only initials for other than the first given name(s).	`"Jan"`	https://openid.net/specs/openid-connect-core-1_0.html#StandardClaims
middle_name	The middle name attribute of the subject as per OIDC. Only middle names from table 36 (' Voorvoegseltabel GBA') of the Dutch GBA MUST be used under this profile. See Logisch Ontwerp GBA-V (element 02.30, page 236).	⚠ Middle names not part of Table 36 will be included in the family name.	`"van"`	https://openid.net/specs/openid-connect-core-1_0.html#StandardClaims
family_name	The family name attribute of the subject as per OIDC.		`"Modaal"`	https://openid.net/specs/openid-connect-core-1_0.html#StandardClaims
birthdate	The birthdate of the subject. The birthdate will be returned in `YYYY-MM-DD` format, as per OIDC. Additionally, the birthdate MAY contain only `YYYY` or `YYYY-MM` in case the exact date is unknown.	In case only the age is sufficient information for access to the service, relying parties SHOULD request one of the less-specific age-claims below. This fullfills the principle of data minimization for privacy reasons. ⚠ As per OIDC, a birthdate with an unknown date will contain only a year (YYYY). In addition and non-conform to OIDC, the birthdate MAY contain a year-month (`YYYY-MM`) in case the exact date is unknown.	`"1983-05-17"`	https://openid.net/specs/openid-connect-core-1_0.html#StandardClaims
urn:nl-eid-gdi:1.0:attribute:age	The age of the subject as number of years. MAY be requested with specified values.	Age is derived from the birthdate.	`34`	
urn:nl-eid-gdi:1.0:	Whether the subject is at least 12 years old. The value '`true`' will be	Age is derived from the birthdate.	`true`	

attribute:12+	returned in case the subject is 12 years or older, '`false`' otherwise. MAY be requested with a specified value.			
urn:nl-eid-gdi:1.0:attribute:16+	Whether the subject is at least 16 years old. The value '`true`' will be returned in case the subject is 16 years or older, '`false`' otherwise. MAY be requested with a specified value.	Age is derived from the birthdate.	`false`	
urn:nl-eid-gdi:1.0:attribute:18+	Whether the subject is at least 18 years old. The value '`true`' will be returned in case the subject is 18 years or older, '`false`' otherwise. MAY be requested with a specified value.	Age is derived from the birthdate.	`false`	
urn:nl-eid-gdi:1.0:attribute:65+	Whether the subject is at least 65 years old. The value '`true`' will be returned in case the subject is 65 years or older, '`false`' otherwise. MAY be requested with a specified value.	Age is derived from the birthdate.	`true`	

OIDC Identifier types and formats

In the OIDC interface specifications, an identity of a subject is often included in a Token, typically as parameter `sub`. As multiple types of identifiers are in use, a value of a `sub` can have a different type.

The actual type and format can be defined by the context or the requested / referenced Service, but can be variable as well. A custom parameter `urn:nl-gdi-eid:1.0:oidc:sub_type` will accompany a `sub` parameter and contains the type of the applicable identifier.

This section describes the valid values for identifier types, their meaning and a description of the identifier format, along with an example.

Identifier types

Type	Description	Comment	Format(s)	Example	References
`urn:nl-eid-gdi:1.0:id:BSN`	BSN, *Burger Service Number* (Dutch citizen service number).	BSN is assigned to natural persons upon registry in the BRP.	The BSN is provided in an encrypted form, according to Polymorphic Encryption and Pseudonimization. Effectively, this is an Encrypted Identity, as defined in the BSNk technical specifications. The BSN after decryption is 9-digits; prefixed using a `0` if numerical value would be only 8 digits.		
`urn:nl-eid-gdi:1.0:id:Pseudonym`	Pseudonym, specific for a User at the Relying Party.	A Pseudonym is a non-linkable derivative of the BSN. In cases where a natural person uses an eIDAS method for authenication, it is derived of the eIDAS-uniquenessIdentifier.	The BSN is provided in an encrypted form, according to Polymorphic Encryption and Pseudonimization. Effectively, this is an Encrypted Pseudonym, as defined in the BSNk technical specifications. The resulting Pseudonym is defined in the BSNk technical specifications.		
`urn:nl-eid-gdi:1.0:id:Transient`	Non-identifier; a transient identifier is effectively a random placeholder. That is, a "transient" `sub` is non-persistent and it is extremely unlikely any user with the same `sub` ever to return.	Equal to SAML Transient identifiers, even though OpenID Connect does not specify these. A transient identifier SHOULD have an entropy of at least 128-bit.	Random string		
`urn:nl-eid-gdi:1.0:id:`	RSIN, Rechtspersonen en Samenwerkingsverbanden Informatienummer	RSIN is assigned upon registration in the trade register (*Handelsregister*).	A RSIN is 9-digits; prefix using `0` if numerical value would be only 8 digits.		

`RSIN`	(organisation identifier)				
`urn:nl-eid-gdi:1.0:id:KvKnr`	Chamber of Commerce number (*Kamer van Koophandel* nummer or *KvK-nummer*)	A *KvK-nummer* is assigned upon registration in the trade register (*Handelsregister*), operated by the Dutch *Kamer van Koophandel* (Chamber of Commerce).	8-digit number		

OIDC Configuration parameters

For these specifications, the following configuration parameters are applicable.

Anyone implementing these specifications MUST ensure these configuration parameters can be modified and these SHOULD be modifiable at runtime by administrators.

Configuration parameters

The following configuration parameters are defined:

Parameter name	usage	unit	(default) value	comment
grant.max-validity	Maximum validity of a OIDC authorization grant	seconds	300	5 minutes
access-token.max-validity	Maximum validity of an access token	seconds	3600	1 hour
id-token.max-validity	Maximum validity of an ID Token	seconds	300	5 minutes, as per https://openid.net/specs/openid-igov-openid-connect-1_0-03.html#rfc.section.3.1
request-uri.min-lifetime	Minimum lifetime of a request object at a request_uri	seconds	900	15 minutes; allows for pre-emptive request object uploading and the user actually selecting to "log in" within a reasonable time.
client-authentication-jwt.max-lifetime	Maximum lifetime of client authentication JWT's	seconds	900	15 minutes; allows for pre-emptive creation of a client authentication JWTs.

OIDC Algorithms

All cryptographic protocols allow for various cryptographic algorithms to be used. These specifications only support a subset of the available algorithms. This is done for two reasons:

- Set a minimum as baseline for cryptographic strength in order to safeguard security and privacy.
- Limit the number of options to ensure interoperability. Algorithms that needs to be supported at minimum are marked as "MUST be supported" in the following tables.

The following algorithms were selected to balance security and interoperability. Algorithms marked as "MUST be supported" are typically required in relevant specifications and are therefore required in these specifications for interoperability, as they are currently considered sufficiently secure. Algorithms marked as "RECOMMENDED" are preferred if supported, as they are more secure but may not always be available as they are often optional in relevant specifications.

Research and advances in cryptography and cryptanalysis can lead to changes in the status of ciphers considered secure. Therefore, cryptography for information security is not static and anyone:

- SHOULD support at least two ciphers per type;
- SHOULD have the effective algorithms be configurable by administrators;
- SHOULD be able to have specific algorithms be disabled by administrators.

This will allow changing the applicable ciphers on short notice in case an algorithm can no longer be trusted and attacks occur in the field.

Signature algorithms

The following algorithms are in scope for signing of messages.

Name	Status	Description	Reference to specification
RS256	MUST be supported	RSASSA-PKCS1-v1_5 using SHA-256	https://tools.ietf.org/html/rfc7518#section-3.3
PS256	SHOULD be supported, RECOMMENDED	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	https://tools.ietf.org/html/rfc7518#section-3.5
RS384	MAY be supported	RSASSA-PKCS1-v1_5 using SHA-384	https://tools.ietf.org/html/rfc7518#section-3.3
RS512	MAY be supported	RSASSA-PKCS1-v1_5 using SHA-512	https://tools.ietf.org/html/rfc7518#section-3.3
PS384	MAY be supported	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	https://tools.ietf.org/html/rfc7518#section-3.5
PS512	MAY be supported	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	https://tools.ietf.org/html/rfc7518#section-3.5

Encryption algorithms

The following algorithms are in scope for encryption of messages.

Symmetric algorithms

The following algorithms are in scope for symmetric encryption of the contents of messages (JWE content encryption).

Name	Status	Description	Reference to specification
A128GCM	MUST be supported	AES in Galois/Counter Mode with a 128-bit key	https://tools.ietf.org/html/rfc7518#section-5.3
A256GCM	SHOULD be supported, RECOMMENDED	AES in Galois/Counter Mode with a 256-bit key	https://tools.ietf.org/html/rfc7518#section-5.3
A128CBC-HS256	MAY be supported	AES_128_CBC_HMAC_SHA_256 authenticated encryption algorithm	https://tools.ietf.org/html/rfc7518#section-5.2.3
A256CBC-HS512	MAY be supported	AES_256_CBC_HMAC_SHA_512 authenticated encryption algorithm	https://tools.ietf.org/html/rfc7518#section-5.2.5

Asymmetric algorithms

The following algorithms are in scope for asymmetric encryption of keys for encryption of contents (content encryption key or CEK) in messages.

Name	Status	Description	Reference to specification
RSA-OAEP	MUST be supported	RSAES OAEP using default parameters, i.e. SHA-1	https://tools.ietf.org/html/rfc7518#section-4.3
RSA-OAEP-256	SHOULD be supported, RECOMMENDED	RSAES OAEP using SHA-256 and MGF1 with SHA-256	https://tools.ietf.org/html/rfc7518#section-4.3

Bijlage: bronnen

Bron	Verwijzing
Kaderdocument Routeringsvoorziening, bevat de kaders vanuit het programma eID en opdrachtgever voor de Routeringsvoorziening. Geconsulteerd in een openbare marktconsultatie.	https://www.tenderned.nl/tenderned-tap/aankondigingen/125934;section=2
OpenID Connect Core specificaties	https://openid.net/specs/openid-connect-core-1_0.html
Overige OpenID Connect specificaties	https://openid.net/developers/specs/
OAuth2 Authorization Framework	https://tools.ietf.org/html/rfc6749
Overige OAuth2 specificaties	https://datatracker.ietf.org/wg/oauth/documents/
eTD interface specificaties tussen Dienstverlener en Makelaar	https://afsprakenstelsel.etoegang.nl/display/as/Interface+specifications+DV-HM
Documentatie Logius over DigiD (aansluiting)	https://www.logius.nl/ondersteuning/digid/
Documentatie Logius over DigiD Machtigen (aansluiting)	https://www.logius.nl/ondersteuning/digid-machtigen/#c10394