Ondertekenen en adresseren in REST

Een profiel voor Edukoppeling

Versie 0.4 (Concept)
Marc Fleischeuers, Kennisnet

Versiehistorie

Versie	Datum	Auteur	Beschrijving
0.0	17-12-2018	M. Fleischeuers	Eerste opzet
0.1	20-12-2018	M. Fleischeuers	Commentaar development team
0.2	7-2-2019	M. Fleischeuers	Verduidelijking ondertekening en plaatjes. Commentaar.
0.3		M. Fleischeuers	Review WG Edukoppeling: administratienummer, niet beperken tot JSON berichten. Algemene tekst aanpassingen.
0.3.1	26-2-2019	M. Fleischeuers	Optionele en verplichte claims toegelicht en aangepast in tabel
0.4	12-3-2019	M. Fleischeuers	Canonicalization toegevoegd

Goedkeuring

Versie	Datum	Naam	Functie

1 Inleiding **2**
1.1 Relatie met andere standaarden 2

2 Beveiliging van berichtverkeer **4**

3 Overzicht van relevante standaarden en referenties **5**

4 Ondertekening **6**
4.1 Alternatieven en keuzes 6
4.1.1 Gebruik van JWS en JWT 6
4.1.2 Ondertekening 7
4.1.3 Structuur van een ondertekening 9
4.1.4 Algoritmen voor c14n, hashing en ondertekening 10
4.2 Verzenden van berichten 11
4.2.1 Verantwoordelijkheden van afzender 11
4.2.2 Verantwoordelijkheden van transparante intermediair 12
4.2.3 Verantwoordelijkheden van ontvanger 14

5 Versleuteling van berichten **16**

6 Adresseren en routeren **17**
6.1 Adresseren 17
6.2 Routeren 18

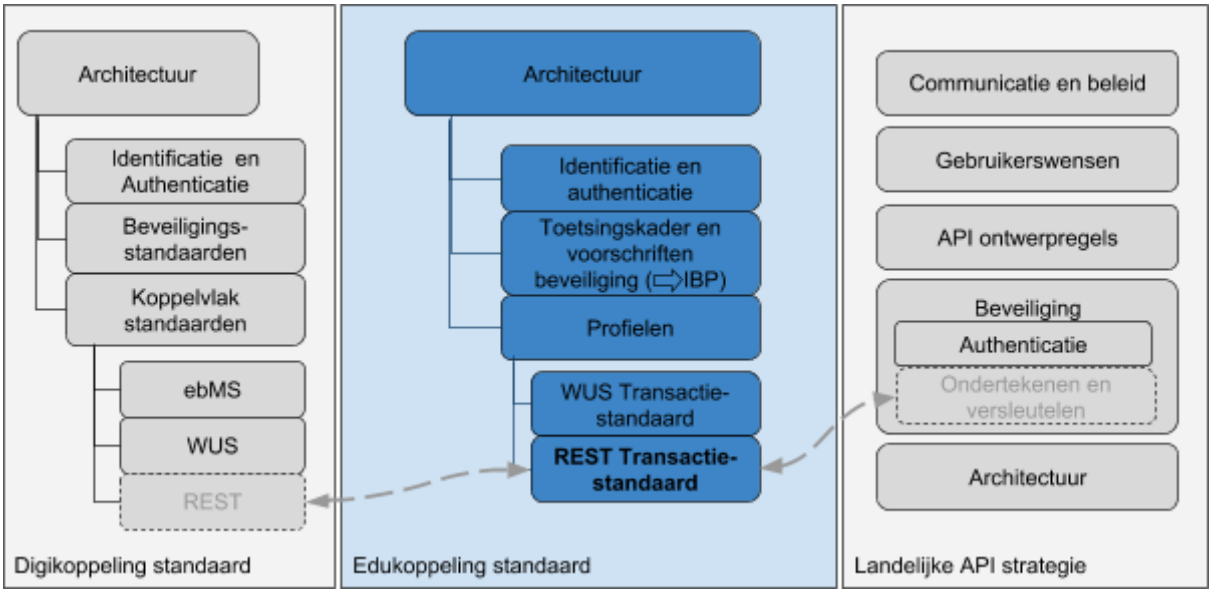
7 Bijlage: Claims **19**
7.1 Claims in de header 19
7.2 Claims in de payload 20

8 Bijlage: Voorbeeld **22**

9 Bijlage: Simple canonicalization **25**

1 Inleiding

Om tegemoet te komen aan een toenemende vraag naar veilige en betrouwbare berichtuitwisseling voor machine - machine koppelingen gebaseerd op REST principes is dit profiel ontwikkeld. Met dit profiel wordt beoogd om soortgelijke waarborgen voor integriteit, veiligheid en interoperabiliteit te bereiken voor REST-gebaseerde koppelingen als nu beschikbaar met het Edukoppeling standaard voor SOAP berichtverkeer.



Figuur 1. Positionering van dit profiel in Edukoppeling, Digikoppeling en de landelijke API strategie.

De ontwikkelingen van dit profiel kunnen niet los worden gezien van soortgelijke (toekomstige) ontwikkelingen bij Digikoppeling en in de Landelijke API strategie. Het streven is om het profiel voor Edukoppeling zodanig op te lijnen met Digikoppeling en de Landelijke API strategie, zodat een eigen REST profiel voor het onderwijs niet nodig zal zijn.

Dit document is bedoeld voor ICT specialisten die betrokken zijn bij het ontwerpen en ontwikkelen van systeem-naar-systeem koppelingen van en / of naar de educatieve sector. Het gaat hier om mensen (ontwikkelaars, architecten, projectmanagers, informatiemanagers enzovoort) die werkzaam zijn bij of voor onderwijsgerelateerde organisaties in publieke en private sectoren.

1.1 Relatie met andere standaarden

Dit profiel beschijft

- De ondertekening van berichten.
- De verificatie van de ondertekening van berichten.
- De adressering van afzender en ontvanger.
- De identificatie van de beoogde service voor het verwerken van berichten.
- (TODO: de versleuteling, ontsleuteling van berichten).

Voor alle andere aspecten van veilig en betrouwbaar berichtverkeer wordt uitgegaan van andere beschikbare standaarden en conventies:

- Routeren van berichtverkeer vindt plaats met een toepasselijk transportprotocol zoals http (hier beschreven). Alternatieven zoals smtp en mqtp zijn denkbaar maar niet verder uitgewerkt.
- Adressering van partijen is gebaseerd op de identificatie van organisaties door middel van het OIN.
- De relatie tussen de identificatie van de partijen (OIN) en een adres voor het transportprotocol (http) is vastgelegd in een (sectoraal) register en dit register is beschikbaar.
- Betrouwbare identificatie van een organisatie aan de hand van het OIN van de organisatie in een PKIo certificaat wordt verondersteld.
- Maatregelen voor een veilig en betrouwbaar tranport (TLS en dergelijke) zijn beschreven in toepasselijke toetsingskaders, bijvoorbeeld het Certificeringsschema.

2 Beveiliging van berichtverkeer

Communicatie op basis van REST-gebaseerde protocollen kan worden beveiligd door berichten te ondertekenen (dat wil zeggen, voorzien van een cryptografisch sterke handtekening) of door te ondertekenen en versleutelen. Er zijn een aantal standaarden die beschrijven hoe een en ander in zijn werk gaat. Binnen deze standaarden zijn voor bepaalde maatregelen meerdere varianten beschikbaar. Hiervoor moeten keuzes worden gemaakt en vastgelegd.

Afspraken voor gebruik van ondertekening of encryptie en ondertekening in berichtverkeer in de onderwijsketen volgen uit de BIV classificatie van het certificeringsschema voor dit berichtverkeer. Daarnaast nog additionele afspraken worden gemaakt, per toepassing of per keten, bijvoorbeeld ondertekenen van berichten als er via brokers of intermediairs gerouteerd wordt.

REST-GEN -001	De toepassing van ondertekening of versleuteling en ondertekening van REST-gebaseerd berichtverkeer in de educatieve keten verloopt volgens de afspraken in dit profiel.	Interoperabiliteit en de gewenste betrouwbaarheid en veiligheid kan alleen maar worden bereikt met nadere afspraken. De generieke standaarden (zie hieronder) zijn breed opgezet en ondersteunen meer soorten berichtuitwisseling dan voor voor dit doel noodzakelijk.
REST-GEN -002	De aanbieder van de dienst bepaalt of zijn dienst gebruikt maakt van *ondertekening* van de invoer en / of uitvoer van de dienst.	De aanbieder van de dienst gebruikt de gevoeligheid van de gegevens van de dienst om te bepalen of ondertekenen van objecten een passende maatregel is.
REST-GEN -003	De aanbieder van de dienst bepaalt of zijn dienst gebruik maakt van *versleuteling* van de invoer en / of uitvoer van de dienst.	De aanbieder van de dienst gebruikt de gevoeligheid van de gegevens van de dienst om te bepalen of versleutelen van objecten een passende maatregel is.
REST-GEN -004	De afnemer van een bericht moet de ondertekening van het bericht valideren als de aanbieder van de dienst bepaald heeft dat de dienst gebruik maakt van ondertekening voor deze uitwisseling.	Als een dienst zijn uitvoer ondertekent, valideert de afnemer deze ondertekening. Als een bericht van deze dienst toch geen ondertekening bevat, zal de afnemer het bericht afkeuren.
REST-GEN -005	Voor ondertekening en / of versleuteling van REST-gebaseerd berichtverkeer worden afspraken gebaseerd op de JOSE-standaarden (zie hieronder) en JWT.	Er zijn meerdere standaarden die gebruikt kunnen worden. Voor deze afspraken beperken we ons tot JOSE, JWS, JWE, JWK, JWA en JWK, en maken hierbij gebruik van JWT.

Tabel 1: Generieke principes voor het gebruik van ondertekenen en versleutelen van berichtverkeer.

3 Overzicht van relevante standaarden en referenties

JOSE	https://tools.ietf.org/html/rfc7520	The JSON Object Signing and Encryption (JOSE) technologies -- JSON Web Signature [JWS], JSON Web Encryption [JWE], JSON Web Key [JWK], and JSON Web Algorithms [JWA] -- can be used collectively to encrypt and/or sign content using a variety of algorithms.
JWS	https://tools.ietf.org/html/rfc7515	JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JSON-based data structures.
JWE	https://tools.ietf.org/html/rfc7516	JSON Web Encryption (JWE) represents encrypted content using JSON-based data structures.
JWK	https://tools.ietf.org/html/rfc7517	A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) data structure that represents a cryptographic key. This specification also defines a JWK Set JSON data structure that represents a set of JWKs.
JWA	https://tools.ietf.org/html/rfc7518	This specification registers cryptographic algorithms and identifiers to be used with the JSON Web Signature (JWS), JSON Web Encryption (JWE), and JSON Web Key (JWK) specifications. It defines several IANA registries for these identifiers.
JWT	https://tools.ietf.org/html/rfc7519	JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties.
IANA JOSE registry	https://www.iana.org/assignments/jose/jose.xhtml	Registratie van parameters en algoritmen voor gebruik bij JSON Object Signing and Encryption
IANA JWT Registry	https://www.iana.org/assignments/jwt/jwt.xhtml	Registratie van claims in JWT tokens
JCS	https://cyberphone.github.io/ietf-json-canon/	JSON Canonicalization Scheme, standaard in ontwikkeling voor het standaardiseren van de notatie van een JSON object

Tabel 2: Overzicht van relevante documentatie en overzichten rondom de berichtstandaarden.

4 Ondertekening

Een digitale ondertekening van een document of tekst is een hash die gemaakt is volgens een cryptografisch sterk algoritme. Een belangrijke eigenschap van een cryptografisch sterk hashing-algoritme is dat het bijzonder onwaarschijnlijk dat twee teksten (bijvoorbeeld een origineel en een iets gewijzigde versie, of twee ongerelateerde teksten) dezelfde hash opleveren. Als de ontvanger van een bericht de hash verifieert en op dezelfde waarde uitkomt als de verzender, kan de ontvanger er van uitgaan dat het ontvangen bericht identiek is aan het bericht dat de afzender heeft verstuurd. Digitale ondertekening is daarmee een belangrijk onderdeel van betrouwbaar berichtverkeer.

Dit hoofdstuk bevat afspraken voor het uitvoeren van digitale ondertekening met behulp van JWT tokens. Deze afspraken worden gehonoreerd door partijen die digitaal berichten uitwisselen met partijen in het onderwijs. Nadere afspraken bovenop de beschreven standaarden zijn noodzakelijk:

- De standaarden ondersteunen methoden van ondertekenen die niet bijdragen aan betrouwbaar berichtverkeer of niet praktisch toepasbaar zijn in het onderwijs.
- De beschreven standaarden zijn flexibel en op meerdere manieren toepasbaar. In het kader van de interoperabiliteit wordt er hier voor een werkwijze gekozen.

4.1 Alternatieven en keuzes

4.1.1 Gebruik van JWS en JWT

JWT is een JSON gegevensstructuur waar ondertekening(en) van een bericht in kunnen worden ondergebracht. JWS is de standaard om cryptografische ondertekening in een JWT aan te brengen en te valideren.

Een JWT (uitgesproken als *jot*) bvat 3 of 5 velden met precies beschreven inhoud, die per toepassing (hier: ondertekening en/of versleuteling) verschilt. Voor een JWT zijn er drie verschillende representaties (formaten voor weergave of uitwisseling) gedefinieerd:

- *Compacte representatie*: base64url weergave van de drie of vijf onderdelen van een JWT, geschikt voor opname in een url of http header.
- *JWS JSON representatie*: gebruik van een JWT als JSON weergaveobject, voor additionele flexibiliteit bijvoorbeeld meerdere ondertekeningen bij encryptie.
- *Platte JWS JSON representatie*: vereenvoudigde variant van de JWS JSON representatie.

Een JWT staat los van het feitelijke bericht (d.w.z. het is een *detached* ondertekening) en moet op een transport-protocol specifieke manier met het oorspronkelijke bericht worden verstuurd naar de ontvanger. Voor het http-protocol zijn er een aantal mogelijkheden:

- Een URL parameter: http://server.io/parm1=value1&parm2=[jwt token]
- Een http “Authorization: bearer” header variabele, als in OAuth2
- Een zelf-gedefinieerde http header variabele
- Een http cookie

Voor de keuze van de JWT-representatie en –vorm is het van belang dat het niet interfereert met bestaande oplossingen. Het gebruik van een Authorization:bearer kan een risico worden, dit is namelijk de header die gebruikt wordt door OAuth. Het is niet uit te sluiten dat ondertekening van

berichten in machine-machine verkeer gebruikt wordt in een context waarbij er ook een OAuth JWT token moet worden doorgegeven. Om die reden kiezen we ervoor om *niet* de Authorization: bearer http header te gebruiken voor doorgifte van het JWT token.

De manier waarop cookies worden gezet en verwijderd (de actie om een cookie te verwijderen is aan de client) lijkt niet te passen met dit gebruik, bovendien zouden cookies eigenlijk betekenisloos moeten zijn.

Opname in een URL kan problemen geven als het token te groot wordt, bijvoorbeeld als het publieke deel van het certificaat onderdeel is van de header van het token.

Daarmee is een eigen header variabele de meest voor de hand liggende manier om JWTs uit te wisselen.

Volgens de RFC's kunnen JWT's andere JWT's bevatten, bijvoorbeeld bij het doorgeven van JWT tokens van een systeem naar een ander systeem. Deze werkwijze is vergelijkbaar met het doorgeven van SOAP berichten als attachment bij een SOAP bericht. De complexiteit en voorspelbaarheid van het berichtverkeer wordt hiermee nadelig beïnvloed.

REST-USE-001	Gebruik de compacte representatie van JWT. Gebruik de JWS JSON representatie als de additionele functionaliteit van die representatie nodig is. Gebruik niet de platte JWS JSON representatie.	Additionele functionaliteiten in JWE JSON zijn: meerdere ondertekeningen, een bericht encrypten voor meerdere geadresseerden, en unprotected (niet-ondertekende) headers.
REST-USE-002	Een JWT in compacte representatie wordt in het http transport protocol van client naar server doorgegeven in de vorm van een zelf-gedefinieerde http header variable '`edustd-jwt`'.	Deze keuze vermijdt mogelijke toekomstige conflicten met OAuth, waarvoor ook JWT's gebruikt kunnen worden, maar dan met andere inhoud.
REST-USE-003	Gebruik geen geneste JWT's.	In elk geval niet totdat er een duidelijke noodzaak voor is, en er duidelijkheid is over de gevolgen voor interoperabiliteit.

Tabel 3: Afspraken die de representatie en doorgifte mbv het http transport beschrijven.

4.1.2 Ondertekening

PKIo certificaten, met als belangrijke eigenschap de hoge zekerheid van identificatie van de partij die het certificaat gebruikt, worden ook gebruikt bij ondertekening van REST berichtverkeer. Het publieke deel van het certificaat wordt opgenomen in een jwk claim in de header van een JWT. Een tweede mogelijkheid is dat partijen het publieke deel van hun certificaat publiceren in een (statische) url, en deze url opnemen in de jwk claim. Met een url in plaats van een publiek deel van een certificaat is een JWT token duidelijk kleiner, ten koste van een iets groter risico dat het certificaat niet beschikbaar is om de ondertekening van het bericht te valideren. Partijen kunnen zelf deze afweging maken, de standaard staat beide vormen toe.

De header en payload van een JWT token bevatten *claims*. Dit zijn naam- en waarde paren (voorbeeld "message": "Hello world"), waarbij de authenticiteit van deze claims wordt door de ondertekening van het token bewezen. Er zijn drie soorten claims gedefinieerd:

- *Registered claims*, die onderdeel zijn van de standaard.
- *Public claims*, hiervan zijn de namen vastgesteld bij IANA.
- *Private claims,* gedefinieerd door de afzender zelf en niet gestandaardiseerd.

Het is dus in principe mogelijk om een JWT token te gebruiken als ‘envelop’ voor de uitwisseling van applicatie-specifieke berichten. Dit is echter niet de intentie van dit profiel. Het voorstel is om de *informatie over de ondertekening* (en later ook de encryptie) van berichten op te nemen in het JWT token, en dit token voegen bij een verder ongewijzigd bericht. Hiermee wordt ook de parallel met de werkwijze van het SOAP profiel van Edukoppeling in stand gehouden.

Het voorstel is om een (in eerste instantie private) claim af te spreken die gebruikt wordt om het feitelijke bericht mee te ondertekenen, gebruik makend van een cryptografisch sterk hashing algoritme, en voor ondertekening van berichtverkeer alleen deze claim op te nemen in de payload van een JWT token. Met deze claim wordt de integriteit van het bericht geborgd. Het JWT token wordt vervolgens op de standaard manier ondertekend met het PKIo certificaat. Hiermee is de integriteit van het JWT token geborgd.

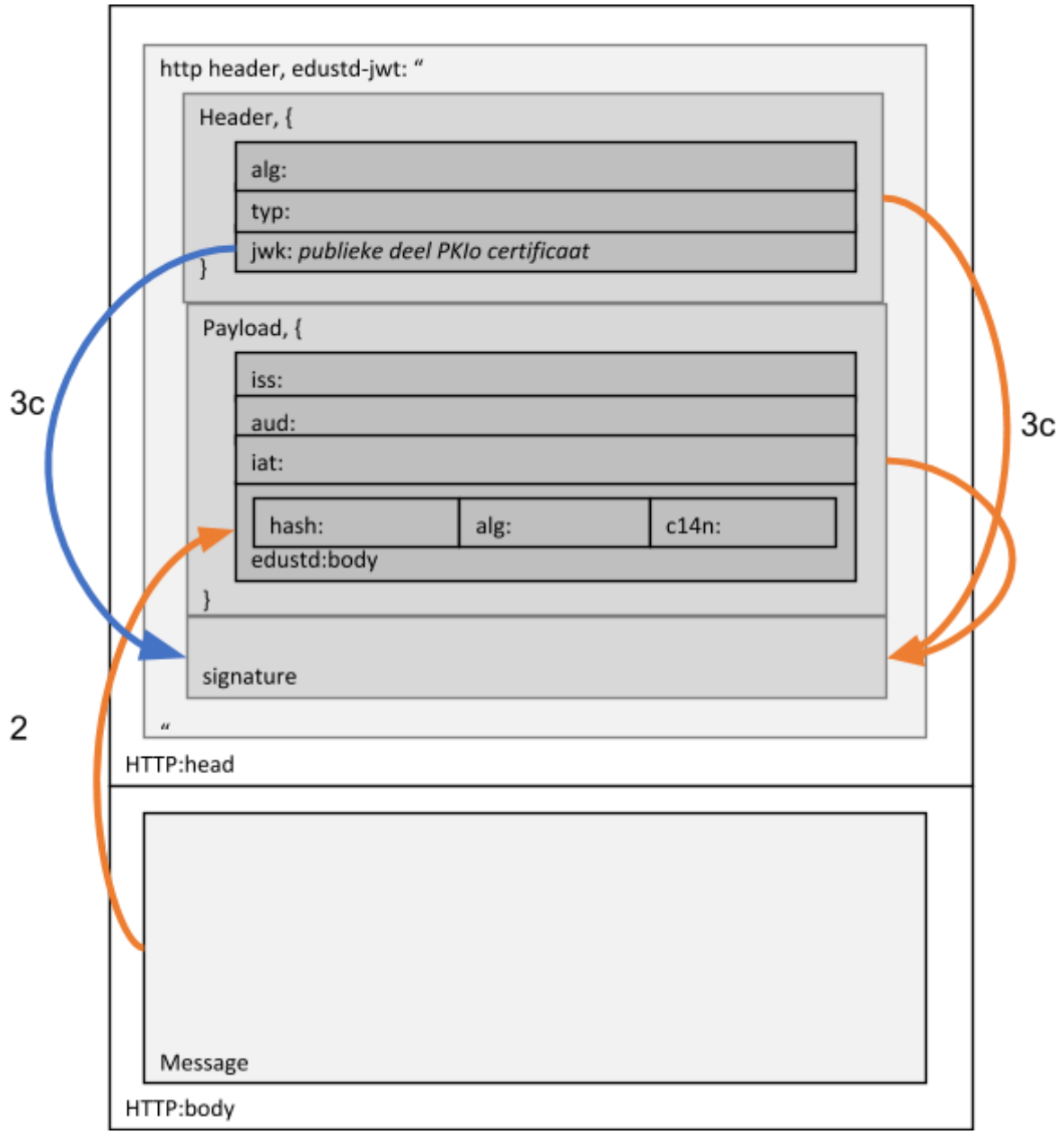
REST-SIG-001	REST berichtuitwisseling maakt gebruik van PKIo certificaten voor ondertekening en encryptie volgens de werkwijze die nu ook voor SOAP berichten wordt gebruikt.	De logica van ondertekening en encryptie bij REST is analoog aan de bekende werkwijze, en dat is belangrijk omdat partijen hier al ervaring mee hebben.
REST-SIG-002	Voor ondertekening van berichten wordt het publieke deel van het PKIo certificaat (of een verwijzing daar naartoe) opgenomen in de header van een JWT token en een claim met de ondertekeing opgenomen in de payload.	
REST-SIG-003	De werkwijze van Edukoppeling met SOAP, waarbij adressering, routering en ondertekening en encryptie van het bericht gescheiden is van de feitelijke inhoud van het bericht, wordt ook gehandhaafd in de werkwijze voor REST.	

Tabel 4: Afspraken rondom de ondertekening van berichten in de vorm van JWT tokens.

Volgens deze afspraken worden deze claims gebruikt:

- **jwk** in de JWT header, waarbij gebruik gemaakt wordt van de opties voor x509 certificaten in de standaard (JWK, https://tools.ietf.org/html/rfc7517)
- **edustd:body** in de payload van het JWT token, dat twee velden bevat:
    - **alg:** bevat het label van het gebruikte hashing algoritme. Zie tabel 6 voor toegestane waarden.
    - **hash:** bevat de hashwaarde van het bericht volgens het opgegeven algoritme.
    - **c14n**: bevat de naam van het algoritme waarmee het bericht op een standaard manier wordt weergegeven. Zie tabel tabel 5 voor toegestane waarden.

Een schematische weergave van een ondertekend bericht dat met het http transport wordt doorgegeven en waarbij het JWT token met de ondertekening is opgenomen in een http header variabele `edustd-jwt` is weergegeven in figuur 2.



Figuur 2. Structuur van een bericht met JWT ondertekening. De nummers bij de pijlen verwijzen naar acties in Verantwoordelijkheden van afzender.

Een voorbeeldbericht is opgenomen in Voorbeeld.

4.1.3 Structuur van een ondertekening

De ondertekening van een bericht is een hashwaarde die volgens een vastgesteld algoritme is berekend. Voor bepaalde soorten berichten zoals JSON of XML berichten moeten mogelijk op een gestandaardiseerde wijze worden geschreven. Dit wordt canonicalization (c14n) genoemd, en elke berichtsyntax heeft eigen algoritmen om dit mee te doen. In tabel 5 staan de ondersteunde algoritmen beschreven.

De hash van *het bericht,* het algoritme waarmee de canonicalization is uitgevoerd en het algoritme waarmee de hash is berekend zijn opgenomen in claim `edustd:body` in de payload van het JWT

token. Het algoritme om deze hash te berekenen gebruikt geen geheimen (*salt*) dus iemand die het bericht aanpast kan ook de hash opnieuw berekenen met het opgegeven algoritme. Het bericht is nu dus nog niet digitaal ondertekend.

De header en payload van het *JWT token* worden ondertekend met het algoritme en bijbehorende paramers dat in de header van het JWT token is beschreven. In deze ondertekening wordt het PKIo certificaat gebruikt: de private sleutel van de afzender voor de ondertekening zelf. De publieke sleutel wordt meegeleverd in de header of beschikbaar gemaakt op een url voor de validatie van de ondertekening.

Met deze operatie wordt het JWT token digitaal ondertekend. Wijzigingen in het bericht leiden tot een ongeldige hash in `edustd:body:hash`, en wijzigingen in die hash leiden tot een ongeldige ondertekening. Zonder private sleutel van het PKIo certificaat van de afzender kan er geen nieuwe geldige ondertekening worden gemaakt.

4.1.4 Algoritmen voor c14n, hashing en ondertekening

Er is nog geen gestandaardiseerd algoritme voor de canonicalization van JSON berichten. Er is wel een standaard in ontwikkeling, JCS (https://cyberphone.github.io/ietf-json-canon/), en hier zijn al enkele voorbeeld implementaties van (https://github.com/cyberphone/json-canonicalization). Zodra deze RFC goedgekeurd is, kan deze standaard worden ingevoerd. Tot die tijd echter willen we een eigen, iets eenvoudigere versie ("c14n": "simple") van dit algoritme voorstellen. Zie Bijlage: Simple canonicalization voor een beschrijving van dit algoritme en de verwachte verschillen met de JCS voorgestelde standaard.

c14n	Canonicalization	Berichtsyntax	Implementatie
simple	Zie Bijlage: Simple canonicalization	JSON	Verplicht
jcs	Volgens https://cyberphone.github.io/ietf-json-canon/ zodra deze RFC definitief is	JSON	Aanbevolen
xmlc14n	XML SOAP canonicalization volgens http://www.w3.org/2001/10/xml-exc-c14n#.	XML	Optioneel

Tabel 5. Lijst met toegestane algoritmen voor canonicalization van berichtsyntax. De termen onder implementatie ('verplicht', 'aanbevolen' en 'optioneel') gelden voor de ontvangers van berichten.

De algoritmen die gebruikt worden voor de ondertekening van het bericht zijn niet gestandaardiseerd in de RFCs of bij IANA. De lijst met toegestane algoritmen hiervoor zijn opgenomen in tabel 6.

alg	Hashing- of MAC algoritme	Implementatie
B64SHA256	BASE64URL geëncodeerde SHA-256 hash	Verplicht

Tabel 6: Lijst met algoritmen voor hashing van het JSON bericht. De termen onder implementatie ('verplicht', 'aanbevolen' en 'optioneel') gelden voor de ontvangers van berichten.

Het B64SHA256 algoritme wordt als volgt geïmplementeerd:

1. Bereken een hash volgens SHA256 van het oorspronkelijke, ongewijzigde bericht.
2. Serialiseer de hash met behulp van de base64url encoding.

De JWA RFC (https://tools.ietf.org/html/rfc7518) beschrijft alle toegestane algoritmen voor ondertekening van het JWT (RFC 7518, par 3.1). Deze lijst bevat een aantal algoritmen die niet bijdragen aan

betrouwbare ondertekening of niet praktisch toepasbaar zijn in de educatieve keten. Van de lijst worden gebruikt:

alg	Digitaal onderteken- of MAC algoritme	Implementatie
RS256[1]	RSASSA-PKCS1-v1_5 met SHA-256	Verplicht
RS384	RSASSA-PKCS1-v1_5 met SHA-384	Optioneel
RS512	RSASSA-PKCS1-v1_5 met SHA-512	Optioneel
ES256	ECDSA met P-256 en SHA-256	Aanbevolen
ES384	ECDSA met P-256 en SHA-384	Optioneel
ES512	ECDSA met P-256 en SHA-512	Optioneel
PS256	RSASSA-PSS met SHA-256 en MGF1 met SHA-256	Optioneel
PS384	RSASSA-PSS met SHA-384 en MGF1 met SHA-384	Optioneel
PS512	RSASSA-PSS met SHA-512 en MGF1 met SHA-512	Optioneel
none	Geen digitale ondertekening of MAC	Niet toegestaan

Tabel 7: Lijst met algoritmen voor ondertekening van het JWT token. De termen onder implementatie (‘verplicht’, ‘aanbevolen’ en ‘optioneel’) gelden voor de ontvangers van berichten.

Ontvangers van ondertekende berichten moeten tenminste de als ‘verplicht’ aangemerkte onderteken-algoritmen ondersteunen (d.w.z. kunnen valideren). Afzenders van berichten met ondertekening kiezen bij voorkeur een van de ‘verplicht’ of ‘aanbevolen’ algoritmen in de tabel en gebruiken niet de ‘niet toegestaan’ algoritmen. Afzenders houden er rekening mee dat ontvangers niet alle optionele of aanbevolen algoritmes ondersteunen.

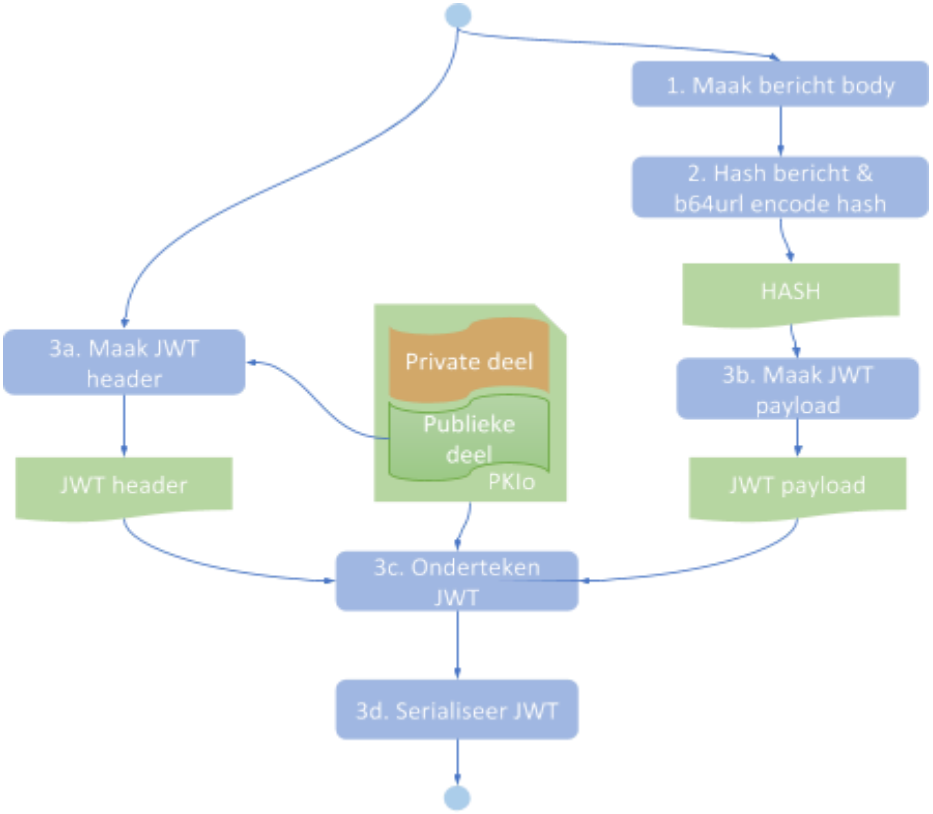
4.2 Verzenden van berichten

4.2.1 Verantwoordelijkheden van afzender

De afzender stelt het bericht (JSON gegevensstructuur) samen dat naar de ontvanger moet worden gestuurd, maakt een ondertekening in een JWT token volgens onderstaande beschrijving en maakt een http bericht met het JWT token in de header en het bericht in de body en stuurt dit bericht naar de ontvanger met een http POST.

De werkwijze om een JWT token te genereren dat het bericht ondertekent is hieronder weergegeven. Deze werkwijze is ontleend aan de JWS RFC (https://tools.ietf.org/html/rfc7515) par 5.1.

[1] Dit is het algoritme dat naar verwachting met de meeste PKIo certificaten zal worden gebruikt.



Figuur 3. Stappen die een afzender zet om een JWT token te genereren dat een bericht ondertekent.

Om een JWT token te maken dat een bericht ondertekekent volgt de afzender de volgende werkwijze:

1. Stel het bericht samen dat naar de ontvanger wordt gestuurd
2. Maak een hash voor het canonicalized bericht aan en encodeer deze hash met base64url encoding, volgens de algoritmen in tabel 6.
3. Maak het JWS token
    a. Maak een JSON header met de claims. Neem hierin tenminste de verplichte claims voor de header (zie Claims in de header) op.
    b. Maak de content voor de JWS payload. Neem hierin tenminste de verplichte claims voor de payload (zie Claims in de payload) op.
    c. Kies een ondertekenalgoritme (*signing algorithm*) en het eigen PKIo certificaat (private en publieke deel). Zie tabel 7 voor een lijst van toegestane algoritmen. Bereken de ondertekening van het JWS token over de JWS header en payload: `JWS Ondertekening = Sign(ASCII(BASE64URL(UTF8(JWT Header)) || '.' || BASE64URL(JWT Payload)), [cert_priv.pem]).`
    d. Maak de geserialiseerde output (JWS compacte serialisatie): `BASE64URL(UTF8(JWT Header)) || '.' || BASE64URL(JWT Payload) || '.' || BASE64URL(JWS Ondertekening).`
4. Bepaal de adressen van endpoint(s) of intermediair voor de geadresseerde(n) in het bericht en routeer het bericht naar deze ontvanger(s).

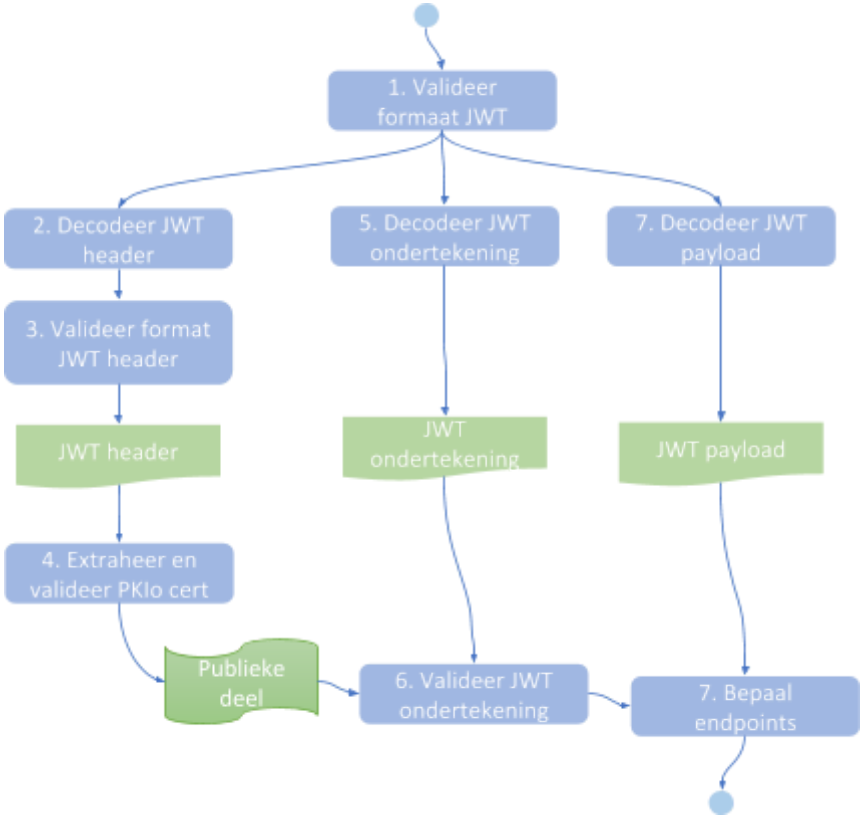
De onderstreepte onderdelen uit bovenstaande beschrijving worden door JWT library functies uitgevoerd.

4.2.2 Verantwoordelijkheden van transparante intermediair

Een transparante intermediair wordt gebruikt als een ondertekend JSON bericht niet rechtstreeks tussen afzender en geadresseerde kan worden gerouteerd, maar wel via een (reeks van)

intermediairs. In dergelijke gevallen stelt de afzender het bericht op voor de beoogd ontvanger, maar levert het af bij de transparanter intermediair.

De intermediair heeft een verantwoordelijkheid voor het valideren van de ondertekening van het JWT token en voor het afleveren van het bericht, maar niet voor de validatie van inhoud of ondertekening van het JSON bericht. In onderstaand overzicht wordt de werkwijze weergegeven en toegelicht die een transparante intermediair volgt om een bericht te routeren (ontleend aan de JWS RFC (https://tools.ietf.org/html/rfc7515) par 5.1).



Figuur 4. Stappen die een transparante intermediair zet om een bericht te valideren alvorens het te routeren.

Elk van onderstaande stappen MOET worden uitgevoerd. Als een van de stappen faalt, kan het token niet gevalideerd worden. Het bericht wordt in dat geval niet doorgestuurd maar vernietigd en de afzender ontvangt een respons met een foutcode 400 (Bad Request) en een toepasselijke foutboodschap.

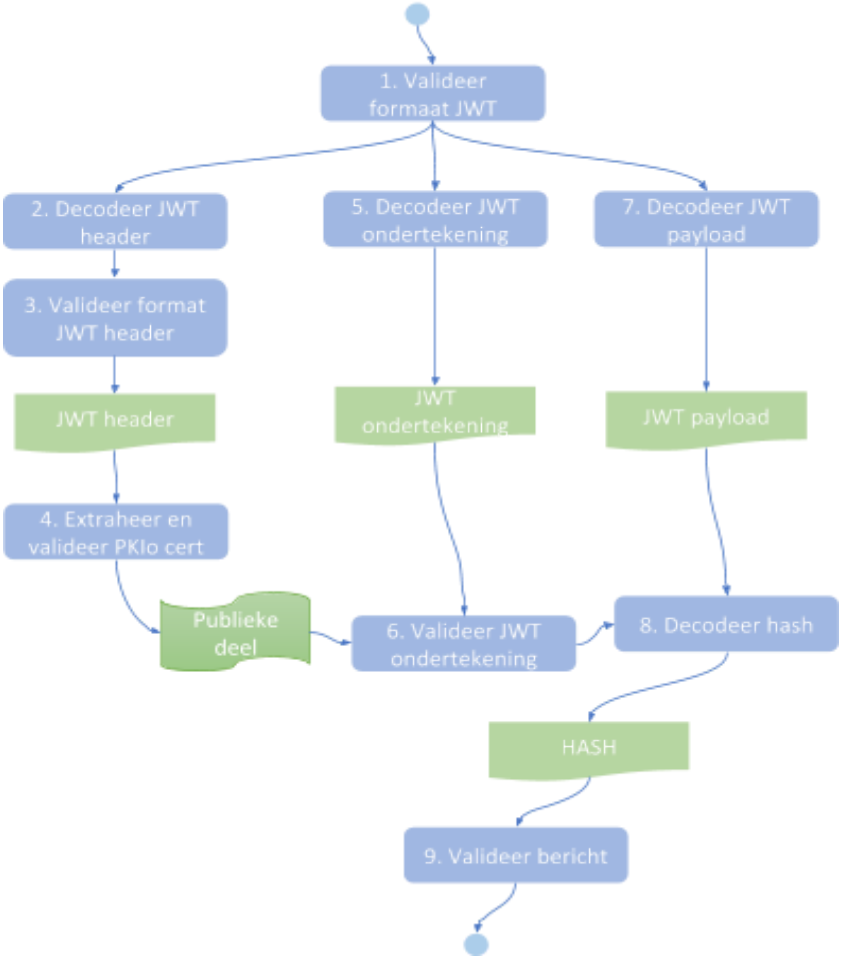
1. Ga na dat het JWT token 3 uit door '.' gescheiden letterreeksen bestaat.
2. Decodeer de header, dat wil zeggen het eerste deel van het JWT token tot aan de eerste punt: `BASE64URLDecode(JWT header)`.
3. Valideer het formaat van de header:
    a. Stel vast dat de gedecodeerde header een UTF-8 representatie is van een geldig JSON object.
    b. Verifieer dat de header geen dubbele parameters bevat.
    c. Verifieer dat er geldige waarden voor het ondertekenalgoritme worden gebruikt.
    d. Ga na of alle verplichte velden in de header verwerkt kunnen worden, net als alle waarden van deze velden.
4. Extraheer en valideer het certificaat dat gebruikt is om het bericht te ondertekenen:
    a. Bepaal het certificaat in de header van het JWT token, ofwel aan de hand van de `x5c` claim ofwel op basis van de `x5u` claim in de header.

b. Ga na dat
    i. Het een geldig PKIo certificaat betreft.
    ii. Het certificaat niet verlopen is.
    iii. Het certificaat niet is ingetrokken door de leverancier van het certificaat.
5. Decodeer de JWT ondertekening, dat wil zeggen het laatste deel van het JWT token, vakaf het eerste karakter na de eerste punt, tot aan het einde: `BASE64URLDecode(JWT signature)`.
6. Valideer de ondertekening van het token tegen de header en payload van het JWT token: `ASCII(BASE64URL(UTF8(JWT Header)) || '.' || BASE64URL(JWT Payload))` op de manier die voorgeschreven is voor het gebruikte ondertekenagoritme JWT header.alg.
7. Bepaal de endpoint(s) of intermediair voor de geadresseerde(n) in het bericht en routeer het bericht naar deze ontvangers.

De onderstreepte onderdelen uit bovenstaande beschrijving worden door JWT library functies uitgevoerd.

4.2.3 Verantwoordelijkheden van ontvanger

De ontvanger valideert het JWS token, en met de ondertekening in de payload van het token, het bericht zelf. Deze stappen gaan uit van een JWT in compacte serialisatie.



Figuur 5. Stappen die een ontvanger zet om de ondertekening van een ontvangen bericht te valideren.

Elk van onderstaande stappen MOET worden uitgevoerd. Als een van de stappen faalt, kan het token niet gevalideerd worden. Het bericht wordt in dat geval niet doorgestuurd maar vernietigd en de

afzender ontvangt een respons met een foutcode 400 (Bad Request) en een toepasselijke foutboodschap.

1. Ga na dat het JWT token 3 uit door '.' gescheiden letterreeksen bestaat.
2. Decodeer de header, dat wil zeggen het eerste deel van het JWT token tot aan de eerste punt: `BASE64URLDecode(JWT header)`.
3. Valideer het formaat van de header:
    a. Stel vast dat de gedecodeerde header een UTF-8 representatie is van een geldig JSON object.
    b. Verifieer dat de header geen dubbele parameters bevat.
    c. Verifieer dat er geldige waarden voor het ondertekenalgoritme worden gebruikt.
    d. Ga na of alle verplichte velden in de header verwerkt kunnen worden, net als alle waarden van deze velden.
4. Extraheer en valideer het certificaat dat gebruikt is om het bericht te ondertekenen:
    a. Bepaal het certificaat in de header van het JWT token, ofwel aan de hand van de `x5c` claim ofwel op basis van de `x5u` claim in de header.
    b. Ga na dat
        i. Het een geldig PKIo certificaat betreft.
        ii. Het certificaat niet verlopen is.
        iii. Het certificaat niet is ingetrokken door de leverancier van het certificaat.
5. Decodeer de JWT ondertekening, dat wil zeggen het laatste deel van het JWT token, vakaf het eerste karakter na de eerste punt, tot aan het einde: `BASE64URLDecode(JWT signature)`.
6. Valideer de ondertekening van het token tegen de header en payload van het JWT token: `ASCII(BASE64URL(UTF8(JWT Header)) || '.' || BASE64URL(JWT Payload))` op de manier die voorgeschreven is voor het gebruikte ondertekenagoritme.
7. Decodeer de JWT payload, dat wil zeggen het tweede deel van het JWT token, vanaf het eerste karakter na de eerste punt, tot aan de tweede punt: `BASE64URLDecode(JWT payload)`.
8. Extraheer en decodeer de hash van het bericht: `BASE64URLDecode(edustd:body:hash:[hashwaarde])`.
9. Canoncalize het bericht volgens het algoritme in `"edustd:body:c14n"`; hash het canonicalized bericht volgens het algoritme gegeven in de JWT payload `"edustd:body:alg"` en vergelijk deze hash met de zojuist gemaakte hash. Als beide waarden identiek zijn is het bericht valide.

De onderstreepte onderdelen uit bovenstaande beschrijving worden door JWT library functies uitgevoerd.

5 Versleuteling van berichten

<TO DO, aandachtspunten:

- publiek certificaat van ontvangende partij moet publiek gepubliceerd zijn, hiervoor hebben we infrastructuur (PKI! Functie voor het serviceregister?)
- Encryptie van JWT en JSON: nog niet helemaal uitgekristalliseerd. Parallel met werkwijze voor ondertekening zou zijn:
    - Encrypt het bericht met een symmetrisch algoritme, plaats de sleutel in een claim in de payload van het JWT token
    - Het JWT Token moet nu ook encrypted worden. Encryptie van het JWT token encrypt payload en signature; header blijft leesbaar
    - Voor routering is het nodig dat claims iss, aud en sub leesbaar zijn - deze moeten dus nu (ook) worden opgenomen in de JWT header

>

6 Adresseren en routeren

6.1 Adresseren

Op basis van een aantal claims in het JWT token kan een bericht gerouteerd worden. Er zijn in de JWS RFC (https://tools.ietf.org/html/rfc7515) claims gedefinieerd voor de afzender van het bericht, de bestemming van het bericht, het onderwerp van de berichtuitwisseling, een unieke identifier van een bericht en een identifier van een eerdere uitwisseling waar een bericht aan gerelateerd is.

De informatie in deze claims is vergelijkbaar met de WS-Addressing velden die ook in Edukoppeling met SOAP gebruikt worden. Het is verleidelijk om private claims te maken met namen die bekend zijn uit de WS-Addressing standaard. Hiermee wordt in elk geval de herkenbaarheid verbeterd. Echter, de meeste bibliotheken en toolkits die gebruikt worden om JWT tokens mee te maken, hebben standaard functies om afzender en ontvangers van berichten in te vullen en uit te lezen, en gebruiken hiervoor de registered JWT claims. In het kader van gemak en interoperabiliteit is het aan te raden om deze conventies over te nemen.

Om afzender en bestemming van het bericht uniek te kunnen identificeren worden OIN's en OIN's met administratienummers voor respectievelijk organisaties en administraties gebruikt. Voor het bewaken van de integriteit van het bericht is belangrijk dat deze properties wel onder de *ondertekening* van een het JWT vallen. Bij versleuteling van een JWT wordt de payload (en daarmee normaal gesproken ook de claims iss, aud, sub) versleuteld terwijl de header van het JWT alleen ondertekend wordt, zodat de claims hierin leesbaar blijven. Volgens de JWT RFC (https://tools.ietf.org/html/rfc7519) par 5.3 is het in geval van een versleuteld JWT mogelijk om de claims iss, aud en sub in de payload van een token te dupliceren naar de header van het token.

REST-ADR-001	De standaard claims "`iss`", "`aud`" en "`sub`" worden gebruikt om respectievelijk de oorspronkelijke afzender, beoogd ontvanger of ontvangers van berichten en de beoogde service voor verwerking van berichten mee aan te duiden.	
REST-ADR-002	De claims voor routering ("`iss`", "`aud`" en "`sub`") worden opgenomen in de *payload* van het JWT token. Als het JWT token versleuteld wordt, worden deze claims ook in de *header* van het JWT token opgenomen.	
REST-ADR-003	Voor de aanduiding van afzender en ontvanger wordt de notatie "`edustd:oin:[OIN]`" gebruikt, waarbij [OIN] wordt ingevuld met het 20-cijferig OIN dat aan de organisatie is toegekend. In situaties dat er routering naar administraties binnen een onderwijsinstelling nodig is, wordt het OIN inclusief het administratienummer in het suffix hier ingevuld.	
REST-ADR-004	Voor automatische routering op basis van het type berichtverkeer wordt de	

	namespace van de berichtstandaard opgenomen in de “sub” claim van het bericht.	

Tabel 8: Afspraken omtrent de adressering en routering van berichten.

6.2 Routeren

Routeren van een bericht is het met behulp van een transport protocol doorgeven van een bericht aan of in de richting van zijn bestemming. Routeren is daarmee een verantwoordelijkheid op het niveau van transport. Om te kunnen routeren is het in het algemeen wel noodzakelijk om te kunnen beschikken over informatie in het bericht of, zoals in dit profiel nader beschreven, in de ondertekening van het bericht.

Om een bericht op basis van deze afspraken te kunnen routeren worden de claims “aud” en “sub” gebruikt. De informatie in deze claims is respectievelijk de beoogde ontvanger of lijst van ontvangers, en een kenmerk dat de beoogde service voor verwerking van het bericht aangeeft. Het Onderwijs Serviceregister van Kennisnet heeft een service om op basis van de informatie in deze claims het endpoint van de service te bepalen. Zie de documentatie van het Onderwijs Serviceregister (https://developers.wiki.kennisnet.nl/index.php?title=OSR:2019/Opvragen_endpoints) voor meer informatie.

7 Bijlage: Claims

In onderstaande tabellen zijn de verplichte en optionele claims van dit profiel opgenomen. Implementaties zijn vrij om extra claims in header en payload op te nemen, en per toepassing of API kunnen er ook extra claims verplicht worden gesteld. Claims die opgenomen zijn in het token die niet kunnen worden verwerkt door de ontvanger van het token moet de ontvanger in beginsel[2] gewoon negeren.
De volgorde van de claims binnen header en payload is vrij.

7.1 Claims in de header

Onderstaande tabel toont de verplichte en optionele claims in de header bij gebruik van JWT voor ondertekening van berichten in het onderwijs.

#	Claim	V/O?	Invulling
h1	alg	v (JWS)	Algoritme voor ondertekening van het JWT token. Zie tabel 7 voor mogelijke waarden.
h2	jwk	v	Container voor velden h2.1 t/m h2.6, bevat het parameters van het PKIo certificaat dat gebruikt is voor de ondertekening van het bericht, inclusief noodzakelijke metadata, formaat volgens RFC7517 (JWK, https://tools.ietf.org/html/rfc7517)
h2.1	kty	v (JWA)	Type van het PKIo certificaat, “RSA”
h2.2	n	v (JWA)	Modulus van de publieke RSA sleutel van het x509 (PKIo) certificaat, geëncodeerd als base64url integer.
h2.3	e	v (JWA)	Exponent van de publieke RSA sleutel, geencodeerd als besa64url integer.
h2.4	x5c *of* x5u	v	Ofwel een lijst met de base64url-geëncodeerde pem representatie van de publieke sleutel van het PKIo certificaat, ofwel een url die naar de pem representatie van de publieke sleutel van het certificaat wijst.
h2.5	kid	o	Key ID, wordt gebruikt om de juiste sleutel te kunnen kiezen als er meerdere publieke sleutels in de lijst zijn opgegeven.
h2.6	use	o	Zegt waar de sleutel voor gebruikt wordt, kan “sign” of “enc” bevatten voor respectievelijk ondertekenen of versleutelen. Hiermee kun je aangeven waar deze sleutel voor gebruikt wordt. Zolang er alleen ondertekening gestandaardiseerd is, is deze parameter optioneel.

Tabel 9: Lijst van voor dit profiel relevante claims in de header van het JWT token. Als een claim verplicht is vanwege een bovenliggende standaard of RFC is dat er tussen haakjes bij vermeld.

Alleen bij versleutelde berichten:

#	Claim	V/O?	Invulling

[2] De optionele claim “crit” bevat een lijst met claims die de ontvanger van een JWT moet kunnen verwerken.

h3	iss	v	Afzender van het bericht (*issuer*), notatie `'edustd:oin' ||` [de OIN van de afzenderorganisatie]. Als deze is opgegeven, is de waarde identiek aan de waarde in de payload.
h4	aud	v	Geadresseerden van het bericht (*audience*). Hier kan een enkele geadresseerde worden ingevuld in een string-notatie, of een lijst van geadresseerden. De notatie voor geadresseerden is `'edustd:oin' || [OIN van de geadresseerde]`, of `'edustd:oin' || [OIN van de geadresseerde inclusief zijn administratie-kenmerk]`. Als deze is opgegeven, is de waarde identiek aan de waarde in de payload.
h5	sub	o	Onderwerp van de berichtuitwisseling (*subject*). Hier is mogelijk om hier het service-version kenmerk (d.w.z. de namespace van de service) in te vullen, zodat deze informatie beschikbaar is om additionele gegevens van de service en zijn endpoint in het OSR op te zoeken. Als deze is opgegeven, is de waarde identiek aan de waarde in de payload.

Tabel 10: Aanvulling van voor dit profiel relevante claims in de header, voor versleutelde JWT tokens.

Voorbeeld van een header met alleen de verplichte velden:

```
{
 "alg": "RS256",
 "jwk": {
    “kty": "RSA",
     "n": "tQpMz... Tn9ZyiQ",
    "e": "AQAB",
    "x5c": [
      "MIIDr ...3TsHy/20="
    ]
  }
}
```

7.2 Claims in de payload

Onderstaande tabel vertoont de verplichte en optionele claims in de payload bij gebruik van JWT voor ondertekening van berichten in het onderwijs.

#	Claim	V/O?	Invulling
p1	iss	v*	Afzender van het bericht (*issuer*), notatie `'edustd:oin' ||` [de OIN van de afzenderorganisatie] of `'edustd:oin' ||` [de OIN van de afzenderorganisatie inclusief administratie-kenmerk]
p2	aud	v*	Geadresseerden van het bericht (*audience*). Hier kan een enkele geadresseerde worden ingevuld in een string-notatie, of een lijst van geadresseerden. De notatie voor geadresseerden is `'edustd:oin' || [OIN van de geadresseerde]`, of `'edustd:oin' || [OIN van de geadresseerde inclusief zijn administratie-kenmerk]`.
p3	sub	o*	Onderwerp van de berichtuitwisseling (*subject*). Hier is mogelijk om hier het service-version kenmerk (d.w.z. de namespace van de service) in te vullen, zodat deze informatie beschikbaar is om additionele gegevens van de service en zijn endpoint in het OSR op te zoeken.

p4	iat	v	Timestamp (seconden sinds epoch) waarop het token is aangemaakt (*issued at*).
p5	nbf	o	Tijdstip (seconden sinds epoch) vanaf welke het JWT token geldig is (*not before*). Als niet opgegeven, gelijk aan 'iat'.
p6	exp	o	Tijdstip (seconden sinds epoch) tot welke het JWT token geldig is (*expires*). Als niet opgegeven, een uur (3.600.000 milliseconden) na iat.
p7	edustd: body	v	Container voor velden p7.1, p7.2 en p7.3, bevat de ondertekening (hash) van het bericht.
p7.1	hash	v	De base64url-geëncodeerde hashwaarde van het bericht zelf, `BASE64URLEncode(HASH(message body))`
p7.2	alg	v	Het algoritme waarmee de hashwaarde van het bericht is bepaald. Zie tabel 6 voor een lijst van toegestane waarden voor deze claim.
p7.3	c14n	o**	Het algoritme voor canonicalization van het bericht. Zie #tab_c14n voor een lijst met toegestane waarden voor deze claim.

Tabel 11: Lijst van voor dit profiel relevante claims in de payload van het JWT token.

*) Als het bericht versleuteld is en dit veld in de header staat, moet het toch ook verplicht in de body opgenomen, met dezelfde waarde als in de header.
**) Dit veld is optioneel omdat niet alle berichten canonicalized moeten worden, maar voor berichten met een gestandaardiseerde syntax (XML, JSON) wel sterk aan te raden. Als deze claim niet is ingevuld, wordt een "simple" canonicalization verondersteld (zie Bijlage: Simple canonicalization).

Voorbeeld van een payload met alleen de verplichte velden:

```
{
 "iat": 1544540142,
  "aud": "edustd:oin:0000000700099AA00123",
 "iss": "edustd:oin:00000003272448340116",
 "sub": "http://osr-api.kennisnet.nl/api/v1",
  "edustd:body": {
      "hash": "cYmtvidgihuvGr94yvpZhE3IWToZtXFzSqPaffxP9nQ=",
      "alg": "b64sha256",
       "c14n": "simple"
   }
}
```

8 Bijlage: Voorbeeld

<TODO bijwerken>
Een voorbeeld van een encoded JWT

eyJhbGciOiJSUzI1NiIsInR5cGUiOiJKV1QiLCJqd2siOnsia3R5IjoiUlNBIiwibiI6InRRcE16aXIzZTB4N0RqNklmczBKenN
JZkhxRWhTSUJTOV9wRjlVUXNHT2dfSlZRZjZuVkVkbFN5VDd2dmZOTjNXZV9ZN1VBdzkyUWxDRHlLaDM1YjQxVk
NGdmNEbHBGSW9jZFBpQk5mMG52Q0pPTmtXWVhUb3RlTWhsNmk4S3U3XzN2NVVWZi1QOGFLZjdZTmt2ZnV
PR093bTNJa3FHZzkweTliTktEczBSc1FPUFZyekwxWm1xXzhzek1iUlZ6MVNtWDRnUkJiMExTQzFPbHNubmRxUWt
2QnZGbDg1SzdMZ3Btd2NMTkJLejhDRkZac2MwcVhuV3VPWm4wM2t2NnNFSnlXNWVncDRSVFE5ZTIwaHI4cDJr
cW40b29ZNXllUmJkSkFqSVItSDB4RjVuSjJFLUxWR2Zfa3NyRUc3WngwWkJyVFp3anpjbkZqM3RhVG45WnlpUSIsI
mUiOiJBUUFCIiwieDVjIjpbIk1JSURyRENDQXBRQ0NRRElMRmtnWEZ1cWpUQU5CZ2txaGtpRzl3MEJBUXNGQURD
Qmx6RUxNQWtHQTFVRUJoTUNUa3d4RlRBVEJnTlZCQWdNREZwMWFXUXRTRzlzYkdGdVpERVRNQkVHQTFVRU
J3d0tXbTlsZEdWeWJXVmxjakVTTUJBR0ExVUVDZ3dKUzJWdWJtbHpibVYwTVJJd0VBWURWUVFMREFsRlpIVmpZ
WFJwYjI0eEVqQVFCZ05WQkFNTUNXeHZZMkZzYUc5emRERWdNQjRHQ1NxR1NJYjNEUUVKQVJZUmRXNXJibTkz
YmtCc2IyTmhiR2h2YzNRd0hoY05NVGd3T1RJMU1UTTFOekF6V2hjTk5EWXdNakE1TVRNMU56QXpXakNCbHpFT
E1Ba0dBMVVFQmhNQ1Rrd3hGVEFUQmdOVkJBZ01ERnAxYVdRdFNHOXNiR0Z1WkRFVE1CRUdBMVVFQnd3S1d
tOWxkR1Z5YldWbGNqRVNNQkFHQTFVRUNnd0pTMlZ1Ym1semJtVjBNUkl3RUFZRFZRUUxEQWxGWkhWallYUn
BiMjR4RWpBUUJnTlZCQU1NQ1d4dlkyRnNhRzl6ZERFZ01CNEdDU3FHU0liM0RRRUpBUllSZFc1cmJtOTNia0JzYjJO
aGJHaHZjM1F3Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLQW9JQkFRQzFDa3pPS3ZkN1RIc
09Qb2grelFuT3doOGVvU0ZJZ0ZMMytrWDFSQ3dZNkQ4bFZCL3FkVVIyVkxKUHUrOTgwM2RaNzlqdFFERDNaQ1V
JUElxSGZsdmpWVUlXOXdPV2tVaWh4MCtJRTEvU2U4SWs0MlJaaGRPaTE0eUdYcUx3cTd2L2UvbFJWLzQveG9wL
3RnMlM5KzQ0WTdDYmNpU29hRDNUTDFzMG9PelJHeEE0OVd2TXZWbWFyL3l6TXh0RlhQVktaZmlCRUZ2UXRJT
FU2V3llZDJwQ1M4RzhXWHprcnN1Q21iQndzMEVyUHdJVVZteHpTcGVkYTQ1bWZUZVMvcXdRbkpibDZDbmhGT
kQxN2JTR3Z5bmFTcWZpaWhqbko1RnQwa0NNaEg0ZlRFWG1jbllUNHRVWi8rU3lzUWJ0bkhSa0d0Tm5DUE55Y1
dQZTFwT2YxbktKQWdNQkFBRXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnRUJBRFNRMnp2SDRud1Q0Ny82WFkze
itSTlkvY0hFK1hjbHZzRm5wTXJSVzQrSlQrQU1LN3pQUWZiNDQ3WktOR3NVcWlQZllFOGo4TEI3TFVyejNsTkxQcF
NwemhHMkF5WVVGSXI5NERISnVWSWtWOHNxajRxZzVUR1QrTEN4c3dhM3E1dU1GSmR4MkpVR0VXSE1JUEF
WYUNZSm50UlJrbDhHM3ZTQnZXK2Z4UEJLWXJsZ0VjbloxT0J1Zjc2MTY3THJnZmNralUxaXFPbE1wQmdvelAzMX
dvMTBDZ0MrWGY2QXZIRllvQzdUM3F0RU1MTzJwUWdkU1BTNHhqYWdFa2NFRWJMekVYc0J6T3drQ0tITWQra
E1vSUdGWkFTa01PRW1RWDhxWXErQThwR1o2SHdTSGtvTzI2SUMzeU5qdDRLL0ZPUEVEUTczNGRtamI5azNUc
0h5LzIwPSJdLCJ4NXQiOiJHODBjY1NPYUw0d3hiRFhYSkhOdm83d3JBSUEiLCJ4NXQjMjU2IjoiNkhpWU1yaVZINW5
1djViOS12dzZGM2xWVGszUWFLem9IV0NGWGFYdVk3ZyIsImtpZCI6Iktlbm5pc25ldCBzaWduaW5nIGNlcnRpZmlj
YXRlIiwiYWxnIjoiUlMyNTYiLCJ1c2UiOiJzaWcifX0.
eyJpYXQiOjE1NDQ1NDAxNDIsIm5iZiI6MTU0NDU0MDE0MiwiZXhwIjoxNTY0NTQzNzQyLCJhdWQiOiIwMDAwM
DAwMzI3MjQ0ODM0MDIwNCIsImlzcyI6IjAwMDAwMDAzMjcyNDQ4MzQwMTE2IiwiaGFzaCI6ImNZbXR2aWRna
Wh1dkdyOTR5dnBaaEUzSVdUb1p0WEZ6U3FQYWZmeFA5blE9In0.
Xol3X3miKfKLwu7MZf9QcGQ-PF464LJcU8R-oFK-RBc24uy2NZo7w05iuGZOMjFx9Mi5VZak_hdYX2cVHBLz7QUxq
ye4vDa-K4zGrkjKvAc84Rr1TpSl7VWX8TqJFfCiOHLCr2tl3-Gi-mxZ3S-1lllhnvffdVr1LlXvC6ZeRJHPjkib9hTNgFWEH2
BO2tW3T3txDha9vJJyDvGBSFPW2oLhXf3PqWZ2SZC601IVcMDmPKM0oESF81OUGiA7sr7AZbKXWhDvHOVUhW
YMwSHxpZBuSBkCyiUn6ZPZvyFTc01YUH5a51p3rx0t1YfJb1JUQpQHH7M6xq5ODDDtPIQlaw
(regeleinden toegevoegd voor de leesbaarheid)

Het gedecodeerde bericht staat hieronder. Naast de verplichte claims bevat dit bericht nog enkele andere claims die door de gebruikte library gegenereerd zijn.

Header:

```
{
```

```
 "alg": "RS256",
 "type": "JWT",
 "jwk": {
        "kty": "RSA",
        "n":
"tQpMzir3e0x7Dj6Ifs0JzsIfHqEhSIBS9_pF9UQsGOg_JVQf6nVEdlSyT7vvfNN3We_Y7UAw92QlCDyKh35b4
1VCFvcDlpFIocdPiBNf0nvCJONkWYXToteMhl6i8Ku7_3v5UVf-P8aKf7YNkvfuOGOwm3IkqGg90y9bNKDs0
RsQOPVrzL1Zmq_8szMbRVz1SmX4gRBb0LSC1OlsnndqQkvBvFl85K7LgpmwcLNBKz8CFFZsc0qXnWuOZn0
3kv6sEJyW5egp4RTQ9e20hr8p2kqn4ooY5yeRbdJAjIR-H0xF5nJ2E-LVGf_ksrEG7Zx0ZBrTZwjzcnFj3taTn9Zy
iQ",
        "e": "AQAB",
        "x5c": [

"MIIDrDCCApQCCQDILFkgXFuqjTANBgkqhkiG9w0BAQsFADCBlzELMAkGA1UEBhMCTkwxFTATBgNVBAg
MDFp1aWQtSG9sbGFuZDETMBEGA1UEBwwKWm9ldGVybWVlcjESMBAGA1UECgwJS2VubmlzbmV0MRI
wEAYDVQQLDAlFZHVjYXRpb24xEjAQBgNVBAMMCWxvY2FsaG9zdDEgMB4GCSqGSIb3DQEJARYRdW5rb
m93bkBsb2NhbGhvc3QwHhcNMTgwOTI1MTM1NzAzWhcNNDYwMjA5MTM1NzAzWjCBlzELMAkGA1UE
BhMCTkwxFTATBgNVBAgMDFp1aWQtSG9sbGFuZDETMBEGA1UEBwwKWm9ldGVybWVlcjESMBAGA1UE
CgwJS2VubmlzbmV0MRIwEAYDVQQLDAlFZHVjYXRpb24xEjAQBgNVBAMMCWxvY2FsaG9zdDEgMB4GCS
qGSIb3DQEJARYRdW5rbm93bkBsb2NhbGhvc3QwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBA
QC1CkzOKvd7THsOPoh+zQnOwh8eoSFIgFL3+kX1RCwY6D8lVB/qdUR2VLJPu+9803dZ79jtQDD3ZCUIPIqHf
lvjVUIW9wOWkUihx0+IE1/Se8Ik42RZhdOi14yGXqLwq7v/e/lRV/4/xop/tg2S9+44Y7CbciSoaD3TL1s0oOzR
GxA49WvMvVmar/yzMxtFXPVKZfiBEFvQtILU6Wyed2pCS8G8WXzkrsuCmbBws0ErPwIUVmxzSpeda45mf
TeS/qwQnJbl6CnhFND17bSGvynaSqfiihjnJ5Ft0kCMhH4fTEXmcnYT4tUZ/+SysQbtnHRkGtNnCPNycWPe1p
Of1nKJAgMBAAEwDQYJKoZIhvcNAQELBQADggEBADSQ2zvH4nwT47/6XY3z+RNY/cHE+XclvsFnpMrRW4+
JT+AMK7zPQfb447ZKNGsUqiPfYE8j8LB7LUrz3lNLPpSpzhG2AyYUFIr94DHJuVIkV8sqj4qg5TGT+LCxswa3q
5uMFJdx2JUGEWHMIPAVaCYJntRRkl8G3vSBvW+fxPBKYrlgEcnZ1OBuf76167LrgfckjU1iqOlMpBgozP31wo
10CgC+Xf6AvHFR/C7T3qtEMLO2pQgdSPS4xjagEkcEEbLzEXsBzOwkCKHMd+hMoIGFZASkMOEmQX8qYq+
A8pGZ6HwSHkoO26IC3yNjt4K/FOPEDQ734dmjb9k3TsHy/20="
        ],
        "x5t": "G80ccSOaL4wxbDXXJHNvo7wrAIA",
        "x5t#256": "6HiYMriVH5nuv5b9-vw6F3lVTk3QaKzoHWCFXaXuY7g",
        "kid": "Kennisnet signing certificate",
        "alg": "RS256",
        "use": "sig"
 }
}
```


Hierin staan de volgende velden:

1. **alg**: het gebruikte algoritme voor ondertekening van het JWT, hier RSASSA-PKCS1-v1_5 met SHA-256
2. **type**: Type van het token, altijd JWT
3. **jwk**: sectie voor het publieke deel van het certificaat dat de afzender heeft gebruikt voor de ondertekening van de payload. De rest van de velden hebben betrekking op dit certificaat.
    4. **kty**: type van het certificaat, hier RSA
    5. **n**: de digitale sleutel van het certificaat, deze wordt gebruikt voor de validatie van de ondertekening. NB deze sleutel is ook onderdeel van het certificaat, maar moet hier expliciet worden opgenomen.
    6. **e**: vaste string?

7. **x5c**: lijst met publieke delen van de certificaat keten. Alternatief hiervoor is x5u met als waarde de url waar het certificaat kan worden opgehaald.
8. **x5t**: de fingerprint van het certificaat, op basis van het sha-1 algoritme
9. **x5t#256**: de fingerprint van het certificaat, op basis van het sha-256 algoritme
10. **kid**: uniek kenmerk van dit certificaat, waarmee het bijvoorbeeld in de keystore kan worden gelabeld.
11. **alg**: het algoritme voor de ondertekening die met het certificaat is gedaan, hier RSASSA-PKCS1-v1_5 met SHA-256
12. **use**: geeft aan of het certificaat bruikbaar is voor ondertekening (sig) of encryptie (enc)

Payload:

```
{
  "iat": 1544540142,
  "nbf": 1544540142,
  "exp": 1564543742,
  "aud": "edustd:oin:0000000700099AA00123",
  "iss": "edustd:oin:00000003272448340116",
  “sub”: “http://osr-api.kennisnet.nl/api/v1”,
   "edustd:body": {
       “hash”: "cYmtvidgihuvGr94yvpZhE3IWToZtXFzSqPaffxP9nQ=",
       “alg”: “B64SHA256”,
       “c14n”: “simple”
     }
}
```

Hierin staan de volgende velden:

1. **iat**: tijdstip waarop het token is aangemaakt (*issued at*)
2. **nbf**: vroegste tijdstip waarop dit token geldig is (*not before*)
3. **exp**: uiterste geldigheid van het certificaat (*expiration*)
4. **aud**: beoogde ontvanger van het bericht, kan zowel een individuele string zijn als een lijst van strings voor een bericht dat bestemd is voor meerdere ontvangers
5. **iss**: oorspronkelijke afzender van het bericht
6. **sub**: namespace of soortgelijk uniek kenmerk van de dienst
7. **edustd:body**: claim met hash en hash-algoritme van het bericht
8. **hash**: hashwaarde van het eigenlijke bericht waar dit JWT token bij hoort (*message hash*)
9. **alg**: algoritme waarmee de hashwaarde is bepaald.
10. **c14n**: algoritme waarmee het bericht is canonicalized.

9 Bijlage: Simple canonicalization

Het simple canonicalization algoritme wordt voorgesteld om op interoperabele manier cryptografische hashwaarden van JSON berichten te kunnen maken, in afwachting van een echt, in een RFC vastgelegd standaard algoritme om dit mee te kunnen doen. De JCS specificatie (https://cyberphone.github.io/ietf-json-canon/) is op weg (maart 2019) om een dergelijk algoritme te standaardiseren. Simple canonicalization zal voor een groot deel van de situaties gelijk zijn aan JCS (zie verderop voor verwachte verschillen).

Simple canoncalization doet het volgende:

1. Spaties, tabs, carriage-return en linefeed (*whitespace*) tussen JSON elementen moet worden verwijderd.
2. Basis datatypes (string, getal, booleans) moeten worden weergegeven met een formattering conform JSON.stringify() van ES6 (Javascript).
3. De sleutel-elementen (*keys*) in het JSON object moeten worden gesorteerd
    a. Op een recursieve manier: als er geneste JSON objecten zijn, moeten de sleutel-elementen in geneste objecten ook worden gesorteerd.
    b. In arrays: als arrays JSON objecten bevatten, moeten de sleutel-elementen van deze objecten worden gesorteerd, maar de waarde-elementen in de array zelf moeten NIET worden gesorteerd.
    c. Sortering moet gebeuren volgens lexicografische ordening van laag naar hoog.

De JCS specificatie gaat verder dan deze specificatie in de manier waarop de basis datatypes worden weergegeven, met name bij de weergave van getallen (decimalen, machten) en strings met bijzondere unicode karakters. De verwachting is echter dat dergelijke waarden niet vaak voorkomen in JSON objecten. Als dat wel het geval gaat worden, moet deze specificatie verder worden aangevuld of moet er gemigreerd worden naar de (hopelijk tegen de tijd vastgestelde) RFC van JCS.

Voorbeeld

Voor canonicalization:


```
{
  "numbers": [333333333.33333,

1e+30, 4.50,          0.002, 0.00000000000000000000000001],
  "string": "\u20ac$\u000F\u000aA'\u0042\u0022\u005c\"\/",
  "literals": [null, true, false]
}
```


Na canonicalization:


```
{"literals":[null,true,false],"numbers":[333333333.33333,1.0e+30,4.5,0.002,1.0e-27],"string":"€$\u000f\nA'B\"\\\"\/"}
```