RFC 0470: Plaats stuurgegevens in SOAP Header ipv payload

<https://discussie.kinggemeenten.nl/discussie/gemma/stuf-301/plaats-stuurgegevens-soap-header-ipv-payload>

**Inleiding**

In deze notitie wordt een voorstel gedaan de implementaties van koppelvlakken vrij te laten in de keuze van het opnemen van van de stuurgegevens. Momenteel is het al zo dat in ‘vrije’ berichten de stuurgegevens optioneel zijn (RFC 0134, <https://discussie.kinggemeenten.nl/discussie/gemma/stuf-301-standaard/rfc-%C2%91vrije%C2%92-berichten-toestaan-om-geen-stuurgegevens-te-gebruiken> ) . Het voorstel in deze notitie gaat dan ook over het gebruik van stuurgegevens in de standaard StUF berichten voor synchrone en asynchrone webservices.

**Scope**

Deze RFC betreft alleen aanpassingen aan het ‘ontwikkelspoor’ zoals bg0320. De scherpe ‘StUF4’ koppelvlakken die ontwikkeld worden in het ‘ontdekspoor’ en als standaard worden vastgesteld vallen buiten de scope.

Binnen de scope vallen niet alleen de vrije berichten maar ook voorgedefinieerde berichten. Elk bericht van het koppelvlak in het ontwikkelspoor, of het nu gebruik maakt van vrije berichten, voorgedefinieerde berichten of een mix van beide, zit in scope.

**Achtergrond**

Achtergrond voor dit voorstel is dat gemeentes zoals Den Haag en Rotterdam op dit moment gebruik maken van een gegevens magazijn waarbij voor de koppelvlakken geconformeerd is aan de StUF BG standaard. Gemeente DenHaag bijvoorbeeld had al meer dan 10 jaar een bewezen stabiele maatwerkoplossing die geen StUF berichten gebruiken maar scherp gedefinieerde berichten. Besloten werd gebruik te maken van een gegevensmagazijn en te conformeren aan de StUF standaard.

Er zijn geen scherpe StUF koppelvlakdefinities beschikbaar voor de RSGB bevragingen, slechts algemene StUF schema’s. Ten tijde van de bouw van het gegevens magazijn was de XSD resolver niet bekend. Het gevolg was dat voor het gegevens magazijn de algemene schema’s gebruikt zijn die niet door een XSD Resolver ingeperkt waren. Deze zijn zo groot dat tijdens testen de definities niet in te lezen waren in testtooling en afnemers met problemen opzadelden. Daarnaast zijn de zoekingangen en bevragingen zo ruim – “alles mag” – dat het gegevensmagazijn instabiel werd of zelfs crashte bij zoekvragen die een enorme verwerking opleveren. De schema’s beperken hierin niet of nauwelijks. De kosten die gemaakt werden om leveranciers voor StUF-BG bevragingen aan te sluiten op het gegevensmagazijn waren veel en veel te hoog. En als laatste verplichtten de schema’s stuurgegevens op te nemen in de payload (body) van het bericht die routerings- en authenticatie/autorisatie gegevens bevatten die, als de schema’s gebruikt worden voor synchrone en asynchrone webservices, volgens wereldwijde standaarden en uit security oogpunt daar niet thuishoren.

Den Haag heeft na een eerdere poging besloten opnieuw deel te nemen aan de StUF Expertgroep om de opgedane ervaringen mee te nemen in de StUF standaard. Onderwerpen hiervoor waren onder andere het verminderen van nillables en het vervangen van de stuurgegevens in de payload door gebruik van de internationale standaard WS Addressing. <https://www.w3.org/Submission/ws-addressing/>.

**Nadelen van de stuurgegevens in payload van bericht**

Een nadeel van het opnemen van stuurgegevens in de payload van het bericht is dat deze niet manipuleerbaar mogen zijn, zeker niet in een keten. Deze gegevens moeten dus beschermd worden. Door het opnemen van de stuurgegevens in de body regelt de vragende applicatie eigenlijk zelf de authenticatie en autorisatie. Dit zou een verantwoordelijkheid van de autorisatie en authenticatie componenten moeten zijn.

Om de stuurgegevens uit de body te kunnen lezen en/of aanpassen moet het gehele bericht gedecrypt en later weer versleuteld en gesigned worden. Dit kost meer tijd dan deze bewerkingen alleen op de header uit te voeren. Daarnaast moet door de hele keten aangetoond kunnen worden dat authenticatie gegevens alleen van de authenticatie service afkomstig kan zijn en de autorisatie gegevens van de autorisatie service. Bovendien is een bericht in zijn geheel versleuteld en moet in zijn geheel gedecrypt worden om de stuurgegevens te kunnen lezen. Wanneer het bericht daarna weer versleuteld en gesigned wordt is niet gegarandeerd dat door deze actie de inhoud van het bericht niet aangepast is.

Als stuurgegevens onderdeel uitmaken van de body van het bericht zijn de authenticatie service en autorisatie service onderdeel van de end-to-end points maar deze voorzieningen zijn niet in staat om het StUF bericht samen te stellen. Dit kan alleen de consument applicatie die de autorisatie en authenticatie gegevens moet verwerken in het bericht. Andersom kan een bericht niet door de authenticatie service en autorisatie service gevalideerd worden zonder de berichtbody uit te lezen, waardoor deze ook daarvoor StUF kennis moet hebben. Doordat de StUF stuurgegevens afwijken van internationale standaarden als WS-Addressing kunnen out-of-the-box authenticatie en autorisatie services niet omgaan met StUF stuurgegevens.

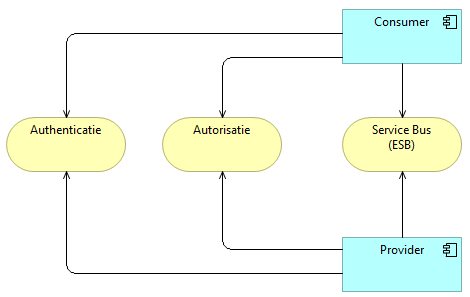
**Mogelijke nadelen van het verplaatsen van stuurgegevens naar de header**

De header is volgens het SOAP Envelope schema vrij om in te vullen. Er kan dus gesteld worden dat de stuurgegevens niet gevalideerd kunnen worden tegen een (StUF) schema. De vraag is echter of dat een probleem is. Wanneer de gegevens niet goed gevuld zijn volgens de afspraken die gelden voor het koppelvlak, werkt de koppeling niet. Dit is al duidelijk voordat de berichtbody verwerkt moet worden, dus nog voor de payload geparseerd wordt tegen het bijbehorende XSD schema.

Bij het verwerken van asynchrone berichten die ontdaan worden van de SOAP envelope en zonder deze envelope verder verwerkt worden kan zich het probleem voor doen dat de service provider geen retourbericht kan sturen omdat de afzender verloren is gegaan. Dit is een nadelig maar logisch gevolg van het deels weggooien van berichten (in dit geval de SOAP envelope). Om dit te voorkomen kan in een dergelijk koppelvlak gekozen worden voor het behouden van de stuurgegevens in de payload of moet het ontvangende systeem de stuurgegevens uit de SOAP header halen en zelf op nemen in het bericht of anderszins bij het bericht opslaan.

**Voorgestelde oplossing**

De volgende oplossing volgt de GEMMA2 model architectuur en legt de verantwoordelijkheden bij de referentiecomponenten die “daar over gaan”.



* Autorisatie wordt ingevuld door de [Gebruikersrechtenbeheercomponent](http://www.gemmaonline.nl/index.php/GEMMA_2/0.9/id-2f84f8d9-f3dc-403d-9919-d030abf1e72f)
* Authenticatie wordt ingevuld door de [Authenticatiecomponent](http://www.gemmaonline.nl/index.php/GEMMA_2/0.9/id-5fc41df6-6e3a-4102-ad64-32f1f7cc36c8)
* Service Bus (ESB) wordt ingevuld door de [Servicebuscomponent](http://www.gemmaonline.nl/index.php/GEMMA_2/0.9/id-8c2cf498-380a-4fc1-b8ce-becf802d6371)
* De rol van consumer kan bijvoorbeeld worden ingevuld door een Zakenbeheercomponent
* De rol van provider kan worden ingevuld door Zaakregistratiecomponent

Maar ook andere referentie componenten kunnen hier van toepassing zijn.

Een applicatie die een service provider aanspreekt mag niet zelf de authorisatie regelen. Dit moet gebeuren via een authenticatie service (AS) die headers kan lezen en een SAML token leveren. De consumer krijgt van de AS een SAML token mee wat doorgestuurd wordt naar de intermediair (service bus). De intermediair kan de stuurgegevens in de header die door de applicatie gevuld worden overschrijven met gegevens uit het SAML token. Vervolgens worden de stuurgegevens door de ontvangende partij (service provider) uitgelezen en gevalideerd met behulp van de autorisatie en authenticatie services.

Om dit mogelijk te maken wordt het volgende voorgesteld:

1. Maak het gebruik van stuurgegevens optioneel in berichten
2. Bepaal per koppelvlak waar de stuurgegevens opgenomen worden (header OF payload).
3. Definieer het koppelvlak scherp.
4. Sluit voor het gebruik van stuurgegevens zoveel mogelijk aan bij internationale standaarden en best practices