SuwiML

Transactiestandaard 3.1

Datum	Versienummer	Auteur	Opmerking
04-12-2013	20131202-01	S. Hadiutomo T. Zwaan	Definitief

Inhoudsopgave

1. Inleiding

De SuwiML Transactiestandaard beschrijft hoe de gestructureerde elektronische informatie-uitwisseling in de Suwi keten is ingericht. Dit document geeft gezamenlijk afgesproken richtlijnen waarmee Suwi partijen eenvoudiger nieuwe gegevensuitwisselingen kunnen opzetten. Er worden allerlei aspecten rondom webservices en berichtenverkeer behandeld. De Transactiestandaard dient gebruikt te worden bij alle projecten waarbij gegevensuitwisseling tussen verschillende partijen plaats gaat vinden. De Transactiestandaard dient ook als naslagwerk in het geval van incidenten en problemen bij bestaande elektronische uitwisselingen. De SuwiML Transactiestandaard is bedoeld voor technisch georiënteerde architecten, technisch ontwerpers, programmeurs en applicatie- beheerders.

Dit document biedt ***toepassing-onafhankelijke*** richtlijnen. De Transactiestandaard richt zich op de ***koppelvlakken*** tussen de applicaties van de verschillende partijen. De Suwi Keten gebruikt een ***contract-first*** benadering: bij een nieuwe informatie-uitwisseling worden de koppelvlak specificaties eerst afgesproken en vastgelegd. Vervolgens zijn de partijen zelf verantwoordelijk voor hoe zij er een applicatie voor optuigen. Het platform en de ontwikkelomgeving voor de applicatie is voor iedere partij vrij te kiezen. Als het uiteindelijke Berichtenverkeer tenminste maar aan de afgesproken koppelvlak specificaties voldoet. In de tekst worden wel nog aanwijzingen gegeven voor de wijze waarop de applicaties operationeel met de Berichten om dienen te gaan. Waar mogelijk worden in de tekst voorbeelden gegeven. Als belangrijk voorbeeld dienen ook de specificaties van de VoorbeeldService, zie http://bkwi.nl/SuwiML/Diensten/VoorbeeldService. In de tekst zal daar ook af en toe naar verwezen worden.

De Transactiestandaard gaat dus over gestructureerde elektronische informatie-uitwisseling. De methoden en technieken die in de Transactiestandaard beschreven worden, worden in de Suwi Keten tot nu toe voornamelijk gebruikt voor enkelvoudige berichten. In de berichten zit een enkel Burgerservicenummer, en beschikbare informatie over die ene persoon. Een andere vorm van gestructureerde elektronische informatie-uitwisseling in de Suwi Keten is die van omvangrijke bulkdata transporten, vanuit een historie waarin veel van batch-processen gebruik gemaakt werd. In principe kun je ook dat soort uitwisselingen vormgeven met de hier beschreven methoden en technieken. Wanneer in de nabije toekomst een bepaalde bulkdata uitwisseling wordt herzien, is het raadzaam om alle betrokken systemen te beschouwen, en te overwegen om de uitwisseling een meer real-time service-georiënteerd karakter te geven. Maar, als in de praktijk blijkt dat andere methoden en technieken toch beter en praktischer zijn voor grote omvangrijke uitwisselingen, dan zouden die methoden en technieken ook in een volgende versie van de Transactiestandaard opgenomen dienen te worden.

Dit alles met het oog op een gestroomlijnde informatie-uitwisseling in de Suwi Keten en mede daardoor een goede dienstverlening richting Burgers en Bedrijven vanuit de Keten.

Afspraak 1: Afspraken worden expliciet vastgelegd in genummerde afspraken.

1.1. Afspraken

1.2. Verschillen met 3.0

In Header-v0400 komen de elementen CdKolomSuwi, CdPartijSuwi en CdVestigingSuwi niet meer voor. Om de identiteit van een organisatie, organisatieonderdeel of een applicatie te bepalen wordt nu alleen gebruik gemaakt van distinguished names. Om de identiteit van de oorspronkelijke aanvrager kenbaar te maken, kunnen in de OrigineleBron zowel de UserDN als de ApplicatieDN worden meegegeven. In de Bron en Bestemming kunnen volstaan de ApplicatieDN worden meegegeven.

Bij de specificaties van een koppelvlak, worden geen Envelope-schema's meer geleverd. Van gebruikers wordt verwacht dat zij zelf de benodigde schema's maken.

De structuur van de inhoud van een SOAP-bericht wordt vanuit de WSDL gegenereerd. Specifieke implementatie richtlijnen staan uitgeschreven in deze standaard. Per koppelvlak wordt apart meegegeven welke elementen, en op welke manier, gebruikt worden.

Vanuit de WSDL file wordt de structuur van de berichten vastgelegd. Hiermee sluiten we aan op de internationale W3C-standaarden en het gebruik van deze standaarden binnen de ontwikkel platforms. Door deze methode te hanteren wordt het mogelijk om elementen, bijv. WS-Addressing elementen, van deze standaard vanuit de WSDL aan te roepen voor het genereren van koppelvlakken.

Door deze wijzigingen sluiten wij beter aan bij de nationale standaard Digikoppeling. Het gevolg is dat ontwikkelaars beter gebruik kunnen maken van standaard functionaliteiten uit SOAP-stacks. Toekomstige uitbreidingen zullen op dit gebied sneller gerealiseerd kunnen worden.

1.3. Doorvoeren van een nieuwe versie

Met deze nieuwe versie van de Transactiestandaard veranderen er zaken, met name in de headers van de berichten. Voor nieuwe services zullen we de nieuwe Transactiestandaard en de nieuwe headers gebruiken bij het definiëren van de koppelvlak specificaties. Het is ook raadzaam dat de bestaande services zich zullen gaan conformeren aan de nieuwe Transactiestandaard. Het is de verwachting dat in het kader van programma's als het Digitaal Klantdossier en andere (vervolg-)projecten er ook wel functionele wijzigingen voor de meeste services zullen komen. Wanneer dat het geval is dan zal in een moeite door ook de service conform de nieuwe Transactiestandaard ingericht wordt. Mocht na verloop van tijd blijken dat er te weinig upgrades plaatsvinden, dan kan er een CMK wijzigingsverzoek voor het upgraden van een webservice aangemeld worden. Dat laatste dan liever niet voor alle webservices tegelijk, maar één-vóór-één.

1.4. Historie

In het regeerakkoord van 1998 werd de uitvoeringsstructuur van de publieke arbeidsvoorziening en de sociale zekerheid ingrijpend gewijzigd. De nieuwe structuur kreeg vorm in Kabinetsbesluiten in 1999 en 2000. Onder andere werd daar besloten tot de oprichting van het CWI. De automatisering van de nieuwe structuur werd onder andere vormgegeven door centrale componenten als het Suwi Gegevens Register (SGR) en het bijbehorende XML vocabulaire SuwiML. Om gestructureerde informatie-uitwisseling met behulp van SuwiML Berichten van de grond te krijgen werd er in 2001 al een eerste versie van de SuwiML Transactiestandaard geschreven. Ook in 2001 kreeg CWI de opdracht om de afzonderlijke eenheid Bureau Keteninformatisering Werk en Inkomen (BKWI) op te

richten. Het BKWI kreeg taken rond ontwikkeling, beheer en onderhoud van gemeenschappelijke afspraken en voorzieningen, bijvoorbeeld ook van het SGR, SuwiML en de Transactiestandaard.

In 2003 ging de Suwinet Inkijk applicatie een belangrijke rol spelen bij het ontsluiten van grote hoeveelheden gegevens uit de bron-systemen van UWV, CWI (heden is CWI een onderdeel van UWV) en Gemeentelijke Sociale Diensten. De techniek van het ontsluiten gebeurde conform de Transactiestandaard. In 2004 en 2005 werden er Elektronische Keten Berichten (EKB's) gemaakt waarmee dossiers overgeheveld werden van de éne naar de andere Suwi Partij. De inzichten die voortkwamen uit die trajecten vonden hun weerslag in versie 2.0 van de Transactiestandaard.

Versie 3.0 is enerzijds geïnspireerd door het Digitaal Klant Dossier (DKD) programma dat in 2007 in de Suwi Keten heeft gelopen. Anderzijds hebben er internationaal ontwikkelingen plaatsgevonden op het gebied van standaardisatie van elektronische informatie-uitwisseling. Met deze nieuwe versie sluit de Transactiestandaard ook weer beter bij die ontwikkelingen aan.

Versie 3.1 bevat wijzigingen met betrekking tot het meer conformeren aan Digikoppeling en het niet meer leveren van Envelope-schema’s bij de koppelvlak-specificaties.

1.4.1 Document historie

Versie	Datum	Auteur
1.0	20-12-2001	Mark Backer, Henk Gingnagel, Arjan Loeffen, Astrid Hackenberg
1.1	29-10-2002	Paul Vriend ism Mark Vlems
2.0	09-01-2004	Paul Vriend ism Mark Vlems, Paul Schlotter, Eduard Renger
3.0	16-03-2009	Dirk Temme
3.1	26-06-2013	Shinta Hadiutomo en Tonkie Zwaan

2. Positionering

In de Suwi-keten hebben we te maken met de gedistribueerde systemen van de verschillende Suwi partijen, en met die van externe bronnen, basisregistraties en bedrijfsleven. De SuwiML Transactiestandaard (en de SuwiML Berichtstandaard) tracht een invulling te geven aan gestructureerde gegevensuitwisseling tussen die gedistribueerde systemen. Uiteraard gebeurt iets dergelijks in allerlei sectoren, en er zijn dan ook vele manieren om die problematiek aan te pakken. Om niet iedereen het wiel opnieuw te laten uitvinden is er standaardisatie nodig. Standaarden geven houvast en enige zekerheid voor de toekomst. In dit hoofdstuk een overzicht van de verschillende standaardisatie-initiatieven, en de relatie ervan tot de SuwiML Transactiestandaard.

2.1. Internationale standaarden

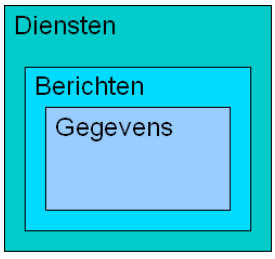
Als resultaat van gezamenlijke internationale samenwerkingsverbanden ontstaan er breed geaccepteerde internationale standaarden, ook op het gebied van gestructureerde gegevensuitwisseling. Sommige van die standaarden hebben inmiddels aanzienlijke draagkracht in de internationale gebruikersgemeenschap, en in de industrie die in bijbehorende software moet gaan voorzien. In de SuwiML Transactiestandaard willen we ons waar mogelijk conformeren aan dit soort internationale standaarden. Voorbeelden van belangrijke en relevante standaardisatie- organisaties zijn W3C, WS-I en OASIS. De SuwiML Transactiestandaard is met name gebaseerd op het WS-I Basic Profile 1.1, en de daarin gerefereerde W3C standaarden SOAP 1.1 en WSDL 1.1.

Afspraak 2: De SuwiML Transactiestandaard conformeert zich aan de Requirements in het WS-I Basic Profile 1.1.

Dit betekent dat er in de SuwiML Transactiestandaard hooguit aanscherpingen van het WS-I Basic Profile 1.1 gedaan zullen worden.

2.2. Nationale standaarden

In de Suwi-keten achten we het ook zinvol om ons aan de principes uit de Nederlandse Overheid Referentie Architectuur (NORA, versie 2.0) te houden. Een van de basis uitgangspunten van de NORA (zie §4.3.2) is dat er voor een Servicegerichte Architectuur (SOA) gekozen wordt. De technieken die daarbij behulpzaam kunnen zijn worden in de Suwi-keten voor een deel (XML, SOAP) al sinds enkele jaren toegepast. De gegevensuitwisseling en de ontwikkeling daarvan is in onze sector echter tot nu toe voornamelijk Bericht-geörienteerd geweest. Door in deze versie van de SuwiML Transactiestandaard ook meer aandacht te schenken aan de Services laag wordt de Informatie-Architectuur in de Keten verder gestandaardiseerd en

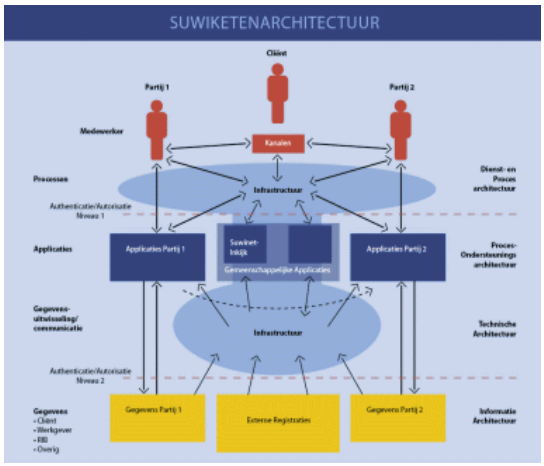


geprofessionaliseerd. De koppelvlakken worden nog beter beschreven, en wordt er een link naar de Proces-architectuur mogelijk gemaakt.

De SuwiML Transactiestandaard beschrijft in feite de servicebus van de Suwi-keten. De toepassing van een zogenaamde servicebus in de scope van overheden en andere sectoren is Digikoppeling (voorheen OverheidsServiceBus). Digikoppeling maakt gebruikt van dezelfde technieken als de SuwiML Transactiestandaard, maar zij maken daarbij andere keuzes. Op termijn zal de SuwiML Transactiestandaard naar de Digikoppeling standaard toegroeien, maar deze versie bevat nog verschillen.

2.3. Standaarden in de keten

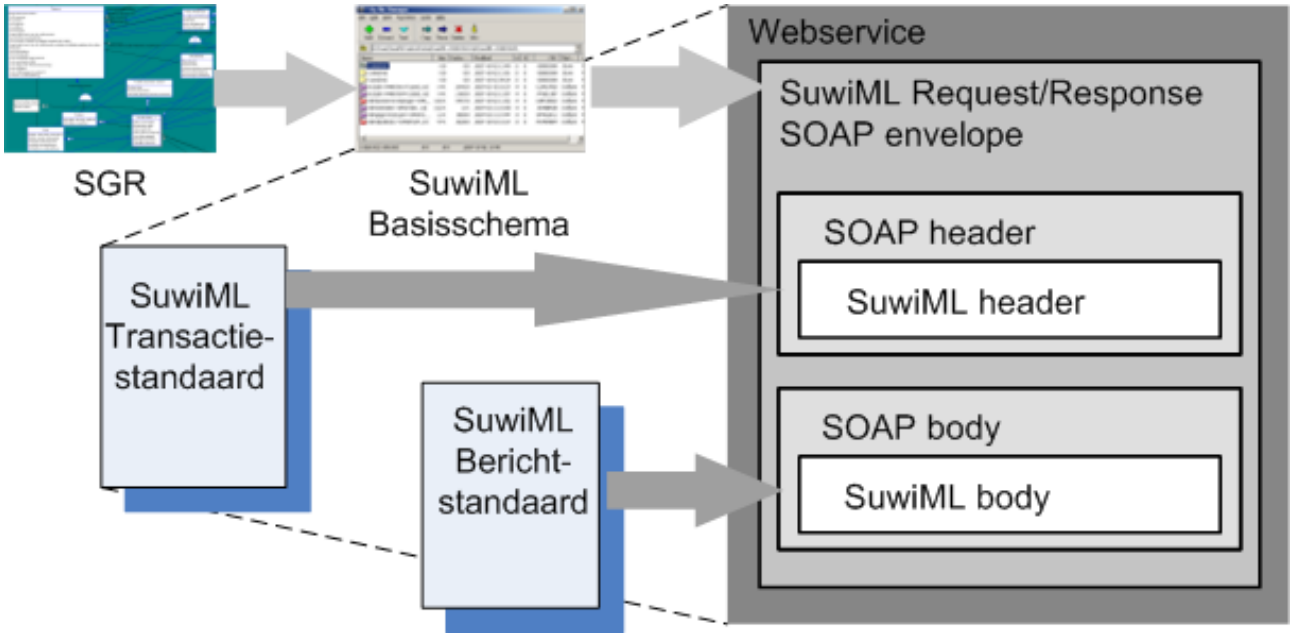
Naast de NORA is er de Ketenarchitectuur Werk en Inkomen (KARWEI). KARWEI versie 2.0 bevat beschrijvingen over Bedrijfsarchitectuur, Informatiearchitectuur en Technische architectuur. De SuwiML Transactiestandaard bevat een technische uitwerking van het Technische Architectuur gedeelte, en dan met name van de in de Suwi-ketenarchitectuur onderkende sessielaag, de transactielaag en de netwerklaag.



In de Suwi-keten kennen we naast de SuwiML Transactiestandaard ook het Suwi Gegevensregister (SGR), het SuwiML Basisschema, en de SuwiML Berichtstandaard. Het SGR is een Entiteiten – Relaties gegevensmodel dat dient als een gemeenschappelijk gedragen weergave van het gehele Suwi werkveld. Als zodanig vormt het ook de basis van de ontwikkelde en te ontwikkelen gegevensuitwisselingen en de koppelvlakken van de diensten. Alle entiteiten en attributen zijn voorzien van gezamenlijk afgesproken definities.

SuwiML is de vertaling in XML van het SGR. Alle entiteiten en attributen uit het SGR hebben een SuwiML XML-tag en datatype. Alle SuwiML bouwstenen tezamen vormen het SuwiML Basisschema. De SuwiML Berichtstandaard is een document met technische richtlijnen voor het definiëren van het inhoudelijke deel van SuwiML berichten. Tenslotte geeft de SuwiML Transactiestandaard dan technische richtlijnen voor het definiëren van de stuurinformatie in de SuwiML berichten, en voor het beschrijven van de koppelvlakken van de bijbehorende webservices.

In Afbeelding 1 worden de relaties tussen de verschillende onderdelen van SGR/SuwiML, noodzakelijk voor de opbouw van een SuwiML dienst, schematisch weergegeven. Het SuwiML basisschema is het XML synoniem voor het Suwi Gegevensregister en is van invloed op alle onderdelen van een SuwiML bericht. De SuwiML transactiestandaard schrijft de SOAP envelope structuur voor, alsmede de stuurgegevens binnen de SuwiML header. De SuwiML berichtstandaard schrijft de opbouw van de SuwiML body voor.



Afbeelding 1 Opbouw berichten vanuit de Suwi standaarden

2.4. Verschillen met Digikoppeling

Deze versie van de Transactiestandaard biedt nog de mogelijkheid om Suwi-specifieke stuurinformatie in de Header van de Berichten op te nemen. Digikoppeling-standaarden staan dat niet toe. Deze versie van de Transactiestandaard staat echter ook toe om de Suwi- specifieke Headers weg te laten, zodat er ook Suwi webservices gemaakt kunnen worden die wèl voldoen aan de Digikoppeling-standaarden.

Digikoppeling maakt onderscheid tussen zogenaamd 'Betrouwbaar' en 'Niet-betrouwbaar' Berichtenverkeer. Gewone Raadplegingen (zoals bijvoorbeeld afkomstig uit Suwinet Inkijk) stellen minder zware eisen aan de betrouwbaarheid rondom aflevering en beantwoording dan berichten die een Signaal, een Trigger, een Melding of een Correctieverzoek bevatten. Digikoppeling heeft gekozen voor de ebMS standaarden omdat die voorzien in een rijkere Header ten behoeve van Betrouwbaar Berichtenverkeer (zie http://www.logius.nl/producten/gegevensuitwisseling/digipoort/koppelvlakken/ebms-20-voor-overheden/).

2.5. Verschillen met Basic Profile 1.1, SOAP 1.1, WSDL 1.1

In de SOAP 1.1 standaard en ook in het Basic Profile 1.1 is het gebruik van een SOAP Header optioneel. In de Suwi keten is er echter een set aan stuurinformatie afgesproken. Op zijn minst gaat er met ieder bericht een MessageID mee in de SOAP Header. Het MessageID is onmisbaar bij het snel verhelpen van problemen. Bovendien dient vanwege het privacy-gevoelige karakter de informatie-uitwisseling in de Suwi Keten tot en met op Bericht niveau gemonitord te kunnen worden.
De standaard SOAP 1.1 en ook Basic Profile1.1 laten de vulling van de SOAP Header geheel vrij. De SuwiML Transactiestandaard perkt die vrijheid in door slechts het gebruik van WS-Addressing Headers en voorgeschreven SuwiML Headers toe te staan.

De WSDL 1.1 standaard kent vier patronen van bericht-uitwisseling: One Way, Request – Response, Sollicit – Response, en Notification. Het Basic Profile 1.1 beperkt de keuze tot One Way of Request – Response. De SuwiML Transactiestandaard verkiest voor alle uitwisselingen het

Request – Response patroon. Ieder Bericht wordt dus beantwoord met een Response Bericht. De naam 'Request' betekent niet dat er in het heengaande Bericht alleen maar een vraag gesteld mag worden. We gebruiken zo'n 'Request' ook om een Melding of een Signaal of een Trigger te versturen.

Het Request – Response patroon kan wat WSDL 1.1 betreft zowel synchroon (Response in het HTTP back-channel van het Request) als asynchroon (Response in een nieuwe HTTP connectie) geïmplementeerd worden. De SuwiML Transactiestandaard verkiest voor alle uitwisselingen de synchrone variant, om daarmee optimale duidelijkheid te verschaffen over wat er precies in het back-channel van HTTP connecties verstuurd dient te worden.

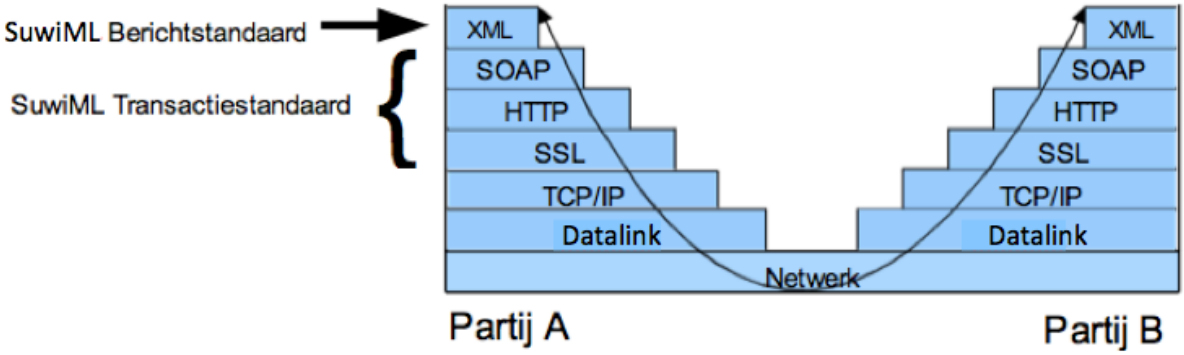
De WSDL 1.1 standaard schrijft voor hoe het gebruik van SOAP 1.1 in een WSDL file beschreven kan worden. De WSDL standaard staat daarbij enkele keuzes toe. Bijvoorbeeld het attribuut 'style' kan de waardes 'rpc' of 'document' bevatten. De SuwiML Transactiestandaard verkiest voor alle uitwisselingen de waarde 'document'. En volgens WSDL 1.1 kan het attribuut 'use' de waardes 'literal' of 'encoded' bevatten. Het Basic Profile 1.1 en de SuwiML Transactiestandaard verkiezen voor alle uitwisselingen de waarde 'literal'.

3. Onderliggende technische basis standaarden

Om tot gestructureerde informatie-uitwisseling tussen verschillende systemen te komen zijn er technische afspraken nodig op verschillende niveaus. Die afspraken worden vastgelegd in protocollen en aanvullend in bijvoorbeeld onderliggende SuwiML Transactiestandaard. In dit Hoofdstuk beschrijven we de basis ingrediënten, de Technische Standaarden waar we gebruik van maken en waar de verdere afspraken in deze Transactiestandaard over zullen gaan.

3.1. Verschillende lagen

Het lagen-model uit Afbeelding 2 toont wat er gebeurt bij informatie-uitwisseling tussen twee partijen. Laag voor laag wordt bij de ene partij de boodschap steeds verder ingepakt totdat het pakket klaar is om over de infrastructuur verzonden te worden. Bij de andere partij wordt het hele pakket dan weer laag voor laag uitgepakt totdat de originele boodschap overblijft.



Afbeelding 2 Technische niveaus en protocollen

De Transactiestandaard beschrijft hoe de lagen 'SOAP', 'HTTP' en 'SSL' ingevuld dienen te worden. De afspraken op de lagen 'SOAP' en 'HTTP' vinden hun weerslag in 'WSDL' beschrijvingen van de diensten van de partijen.

3.2. De XML laag

De bovenste laag is de XML laag, ook wel genoemd de 'Payload'. Op bedrijfs-applicatieniveau is bepaald dat er gegevens moeten worden uitgewisseld tussen verschillende applicaties / systemen. Hier is bekend welke gegevens uitgewisseld gaan worden en voor welke applicatie(s) de gegevens bestemd zijn.

De inhoudelijke functionele informatie die tussen partijen wordt uitgewisseld wordt vormgegeven en gestructureerd met behulp van XML. De XML variant van de Suwi-keten heet SuwiML. In de SuwiML Berichtstandaard staat tot in detail beschreven hoe met behulp van SuwiML de bericht-inhoud samengesteld en vastgelegd wordt. De Transactiestandaard doet daar verder geen uitspraken over.

3.3. De SOAP laag

De volgende laag is de SOAP laag, ook wel genoemd de berichtlaag. Hier wordt bepaald hoe de gegevens uitgewisseld moeten worden. De XML met de inhoudelijke boodschap wordt voorzien van nog wat extra XML-tags conform het SOAP 1.1 protocol. Een SOAP bericht bestaat uit een SOAP Envelope met daarin een SOAP Body en daarin weer de inhoudelijke XML boodschap uit de laag er boven. De SOAP Envelope bevat ook een SOAP Header met stuurinformatie op het gebied van Bericht-Identificatie, Routering, Adressering en Transacties. De precieze invulling van de SOAP Header wordt beschreven in Hoofdstuk 5 SuwiML Berichten.

Afspraak 3: Ieder SuwiML Bericht heeft een SOAP Header met stuurinformatie

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Action>http://bkwi.nl/SuwiML/Diensten/VoorbeeldService/Aanvraag</wsa:Action>
    <wsa:MessageID>5454587841</wsa:MessageID>
    <smlh:SuwiMLHeader xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
      <RouteInformatie> ... </RouteInformatie>
      <BerichtIdentificatie> ... </BerichtIdentificatie>
      <Transactie> ... </Transactie>
    </smlh:SuwiMLHeader>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <smls:AanvraagInfo xmlns:smls="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService">
      <Burgerservicenr>123456789</Burgerservicenr>
    </smls:AanvraagInfo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Afbeelding 3 Structuur van een SuwiML SOAP-Envelope (voorbeeld)

3.4. De http laag

De HTTP laag (ook bekend als de transportlaag) verzorgt de bezorging van de berichten. Het SOAP bericht wordt naar een HTTP adres gestuurd. HTTP is de naam van het Transport protocol. HTTP informatie wordt meegestuurd met het bericht in de vorm van HTTP parameters. Bijvoorbeeld in de parameter Content-Type wordt een indicatie gegeven van de karakterset die gebruikt is in het bericht.

```
POST /axis2/services/VoorbeeldService HTTP/1.1
Content-Length: 1301
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 127.0.0.1:8081

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
      ...
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <AanvraagInfo xmlns="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService">
         <Burgerservicenr>123456789</Burgerservicenr>
      </AanvraagInfo>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Afbeelding 4 Voorbeeld van een SOAP bericht met daarboven de HTTP parameters

Het hangt van de gebruikte ontwikkel-tools af of en in welke mate er nog invloed mogelijk is op de HTTP parameters. De parameter SOAPAction wordt echter vastgelegd door de WSDL beschrijving van een webservice. Zie daarvoor ook 4.1.3 De WSDL Binding.

3.5. De SSL laag

Een HTTP connectie kan versleuteld worden met behulp van SSL (Secure Sockets Layer). Voor websites en webservices waarbij privacy-gevoelige of andere vertrouwelijke informatie verstuurd wordt (dus ook voor bijna alle SuwiML Diensten) is adequate versleuteling een must. SSL is een breed geaccepteerde en geïmplementeerde standaard daarvoor.

Afspraak 4: Vertrouwelijke informatie dient versleuteld verstuurd te worden.

De service provider installeert daartoe een X509 server-certificaat op zijn webserver. De URL waar het request heen gestuurd wordt begint dan met de protocolnaam https://.... Om de informatie-uitwisseling ongestoord automatisch te laten verlopen moet de service requester in zijn component bekend maken dat de service provider vertrouwd kan worden. Dat kan door de publieke sleutel van het certificaat van de service provider in een keystore op te slaan.

Een service provider zal van een service requester verlangen dat die zich identificeert. Dat kan óók met behulp van een X509 certificaat. De service requester installeert een X509 client-certificaat en de service provider installeert de bijbehorende publieke sleutel in zijn keystore. Dit levert dan een zogenaamde 'dubbelzijdige' SSL connectie op. Zeker in het geval van vertrouwelijke informatie dient een service provider de identiteit van de service requester te controleren (authenticatie) en ook moet gecontroleerd worden of die partij recht heeft op toegang tot de gevraagde dienst (autorisatie). SSL voorziet in een mechanisme daarvoor.

Afspraak 5: Vertrouwelijke informatie wordt alleen verstrekt aan geautoriseerde partijen.

In de praktijk wordt de authenticatie en autorisatie van het SSL protocol afgehandeld in aparte hardware (bijvoorbeeld een Content Server Switch) of in aparte software die zich als eerste

toegangspoort in de webserver installeert. De ontwikkelaar van de webservices en de SOAP berichten hoeft zich daar dan als het goed is geen zorgen meer over te maken.

Er bestaan andere methoden dan SSL om te versturen informatie te versleutelen. De standaard WS- Security biedt faciliteiten om op het hogere SOAP niveau de inhoud van het SOAP -bericht te versleutelen. Op het moment van schrijven (2008) wordt in de Suwi-keten nog geen WS-Security toegepast. Deze versie van de Transactiestandaard ondersteunt nog geen WS-Security. Wanneer voor een bepaalde bericht-uitwisseling wèl WS-Security gebruikt gaat worden, dan zou het versleutelen met SSL voor die uitwisseling wellicht niet meer nodig zijn of op z'n minst her-overwogen dienen te worden.

De meest recente versies van SSL heten Transport Layer Security (TLS). De principes en de gebruikte certificaten zijn bij SSL en TLS dezelfde. Maar bijvoorbeeld de technische invulling van de handshake is bij TLS veiliger en meer hackers-proof. De term 'SSL' is ingeburgerd en wordt veel breder gebruikt, ook voor situaties waarbij er technisch gesproken eigenlijk TLS gebruikt wordt. Welke versie van SSL of TLS er precies gebruikt wordt is wat deze versie van de Transactiestandaard betreft niet de belangrijkste kwestie. En dat zal ook afhangen welke versies er ondersteund worden door de betrokken hard- en software. Het belangrijkste op dit gebied is dàt er SSL of TLS gebruikt gaat worden.

3.6. WSDL en XML schema

Conform het WS-I Basic Profile 1.1, zie http://www.ws-i.org/Profiles/BasicProfile-1.1.html, worden de koppelvlakken van Suwi webservices beschreven met behulp van Webservices Description Language (WSDL) 1.1 files. WSDL is net als SuwiML en SOAP ook een gestructureerde XML variant. Een WSDL beschrijving van een webservice laat precies zien hoe de lagen 'SOAP' en 'HTTP' uit bovenstaande Afbeelding 2 ingevuld zijn voor die webservice.

Afspraak 6: De koppelvlak specificaties van een SuwiML Webservice worden vastgelegd in en bepaald door een WSDL beschrijving

De WSDL file is dus 'leading'. Documenten als deze Transactiestandaard en het WS-I Basic Profile dienen 'slechts' ter ondersteuning. Vanuit de WSDL file wordt precies de structuur van de berichten vastgelegd. Hiermee sluiten we aan op de internationale W3C-standaarden en het gebruik van deze standaarden binnen ontwikkel platforms. Door deze methode te hanteren wordt het mogelijk om elementen van deze standaard vanuit de WSDL aan te roepen voor het genereren van koppelvlakken. Denk hierbij aan bijvoorbeeld WS-Addressing functionaliteiten.

In de WSDL file worden 'import's gedaan van een aantal XML Schema files. De XML Schema files beschrijven de afzonderlijke bouwstenen die betrokken zijn bij de webservice. Het betreft bouwstenen uit het SuwiML Basisschema, en bouwstenen voor herbruikbare componenten zoals Headers en Foutafhandeling.

4. SuwiML webservices

Iedere afzonderlijke webservice interface / gegevensuitwisseling, wordt volgens SGR/SuwiML vastgelegd door een technische beschrijving in een WSDL file met bijbehorende SuwiML bericht-schema's. Een webservice wordt uniek geïdentificeerd door middel van de targetNamespace van de WSDL file. In die URI dient ook het versienummer opgenomen te worden. De webservice URI komt ook terug als namespace in de requests die naar de webservice gestuurd worden en de responses die door de webservice terug gestuurd worden.

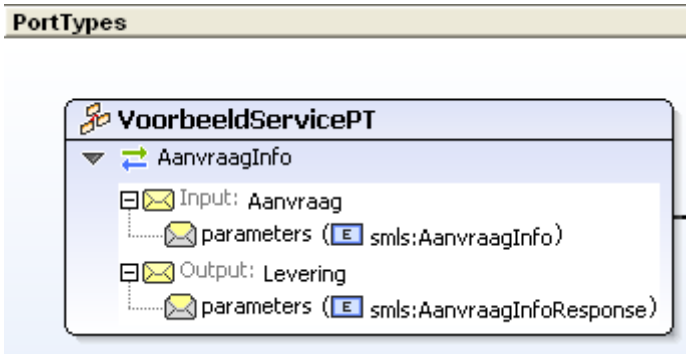
4.1. De WSDL file van een SuwiML webservice

In de WSDL wordt vastgelegd hoe het request er uit moet zien dat door de zender (de client) verstuurd zal worden, èn hoe dan het bijbehorende response van de ontvanger (de server) er uit zal zien.

Indien er een fout optreedt tijdens de gegevensuitwisseling, bijvoorbeeld als de ontvanger niet in staat is de request te verwerken en/of te beantwoorden, dan wordt een foutmelding geretourneerd. Hoofdstuk 6 Foutafhandeling beschrijft in detail op welke wijze de foutafhandeling plaatsvindt binnen de verschillende lagen.

4.1.1 De WSDL PortType

Het centrale onderdeel van een WSDL file is de PortType. Met de PortType wordt de interface van de webservice beschreven, in termen van een of meer Operations voorzien van Input en Output parameters.



Afbeelding 5 De PortType definitie in een WSDL

Afspraak 7: Iedere Operation van een SuwiML webservice heeft zowel een Input als een Output

Dit geldt ook voor het geval van webservices van het type Meldingen (zie 4.2 Inkijk versus Meldingen). Bij Meldingen zal de Output altijd een Acknowledgement zijn, ter bevestiging van de ontvangst van de melding in het request.

```
<wsdl:definitions ... xmlns:smls="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService">
   ...
   <wsdl:message name="Aanvraag">
      <wsdl:part name="parameters" element="smls:AanvraagInfo"/>
   </wsdl:message>
   <wsdl:message name="Levering">
      <wsdl:part name="parameters" element="smls:AanvraagInfoResponse"/>
   </wsdl:message>
   ...
   <wsdl:portType name="VoorbeeldServicePT">
      <wsdl:operation name="AanvraagInfo">
         <wsdl:input message="Aanvraag"/>
         <wsdl:output message="Levering"/>
      </wsdl:operation>
   </wsdl:portType>
   ...
</wsdl:definitions>
```

Afbeelding 6 XML weergave van een PortType

Aan de XML weergave in Afbeelding 6 is te zien dat in de Operation 'AanvraagInfo' voor de Input verwezen wordt naar het Element 'smls:AanvraagInfo' en voor de Output naar het element 'smls:AanvraagInfoResponse'. Die Element namen zullen in het berichtenverkeer terug komen als de eerste tag in de SOAP Body van een Request en van een Response.

4.1.2 Document – literal wrapped stijl

Een document literal wrapped inrichting van een webservice zorgt ervoor dat de XML requests en responses optimaal gemapped kunnen worden op een object structuur die vrij is van overbodige Request en Response objecten. Op dit moment is er met de document literal wrapped stijl en met bijbehorende afspraken ook de beste kans op succes bij het automatisch interpreteren van een WSDL door de verschillende beschikbare SOAP toolkits, bijvoorbeeld die van Microsoft .Net en Apache Axis2. Zie bijvoorbeeld http://pzf.fremantle.org/2007/05/handlign.html en http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/. Bovendien ondersteunt Digikoppeling dit advies.

Afspraak 8: Iedere SuwiML webservice heeft een 'Document – Literal Wrapped' interface

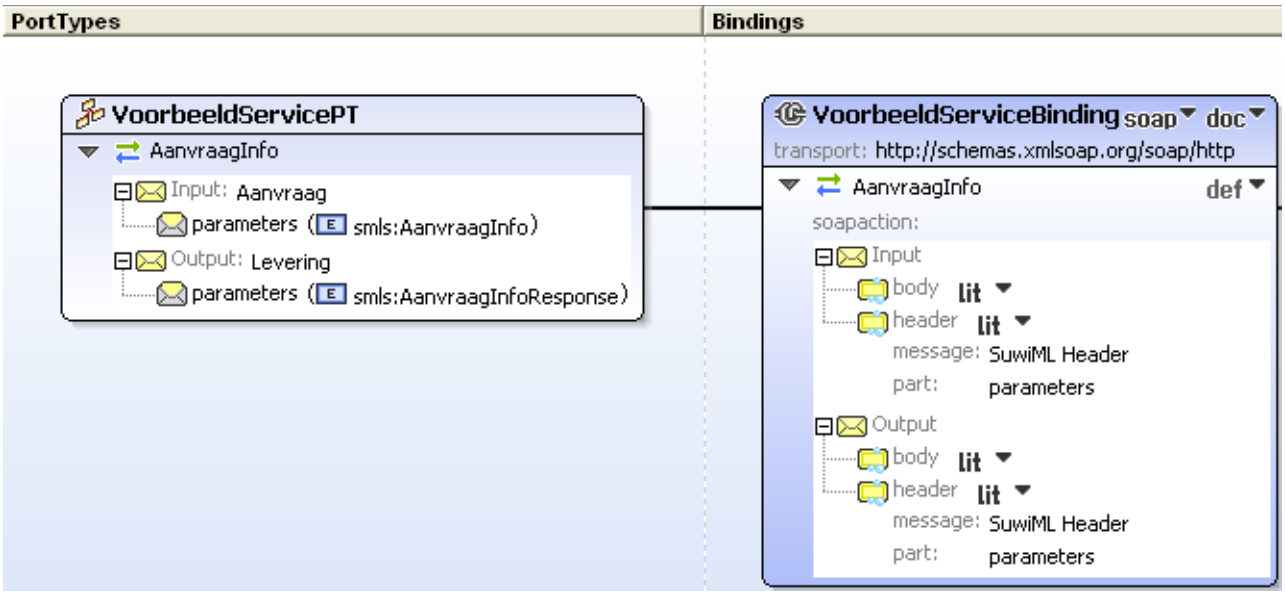
Om een 'document-literal wrapped' interface te krijgen dienen enkele richtlijnen gevolgd te worden. Voor de PortType en de bijbehorende Messages in een WSDL geldt het volgende:

- Iedere Message heeft slechts één Message Part met de naam 'parameters';
- Ieder Message Part verwijst naar een 'element', niet naar een 'type';
- De naam van het Message Part element in de Input is gelijk aan de naam van de Operation;
- De naam van het Message Part element in de Output is gelijk aan de naam van de Operation, met Response' er aan toegevoegd.

4.1.3 De WSDL Binding

De WSDL 1.1 standaard is op zichzelf niet gebonden aan het gebruik van SOAP, maar er is wel in de WSDL 1.1 standaard een SOAP Binding opgenomen. Deze SOAP Binding gebruikt de SOAP

1.1 (zie http://www.w3.org/TR/soap/) standaard. Het WS-I Basic Profile 1.1 (en ook deze versie van de SuwiML Transactiestandaard) stelt het gebruik van deze SOAP 1.1 Binding verplicht. Als gevolg daarvan zullen de berichten van en naar WS-I Basic Profile 1.1 conforme webservices opgesteld zijn in het SOAP 1.1 formaat. Waar het WS-I Basic Profile 1.1 nog wat vrijheid overlaat voor de 'style' van de binding, kiest de SuwiML Transactiestandaard dus voor de 'document – literal' style. Daarom heeft het attribuut 'style' van een SOAP Binding de waarde 'document'.



Afbeelding 7 De SOAP binding van een webservice

In de SOAP Binding wordt ook de waarde van de HTTP parameter SOAPAction vastgelegd. Aangezien er met de introductie van WS-Addressing in de SOAP Headers een veld <wsa:Action> opgenomen kan worden, verliest de HTTP parameter SOAPAction zijn waarde. In de WSDL moet er echter wel iets over gezegd worden. Voor SuwiML webservices kiezen we er voor om de SOAPAction leeg te laten: <soap:operation soapAction=””/>, zie Afbeelding 8

```
<wsdl:definitions ... xmlns:smls="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService"
xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
   ...
   <wsdl:message name="SuwiMLHeader">
      <wsdl:part name="parameters" element="smlh:SuwiMLHeader"/>
   </wsdl:message>
   <wsdl:portType name="VoorbeeldServicePT"> ... </wsdl:portType>
   <wsdl:binding name="VoorbeeldServiceBinding" type="VoorbeeldServicePT">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsaw:UsingAddressing wsdl:required="true"/>
      <wsdl:operation name="AanvraagInfo">
         <soap:operation soapAction=""/>
         <wsdl:input>
            <soap:body use="literal"/>
            <soap:header message="SuwiMLHeader" part="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
            <soap:header message="SuwiMLHeader" part="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   ...
</wsdl:definitions>
```

Afbeelding 8 De XML weergave van een SOAP binding

Afspraak 9: Voor SuwiML webservices is de HTTP parameter SOAPAction leeg: “”

In de Binding dient ook informatie over benodigde stuurinformatie vastgelegd te worden. Om aan te sluiten bij de Digikoppeling-standaarden maken we gebruik van WS-Addressing.

Afspraak 10: SuwiML webservices maken gebruik van WS-Addressing

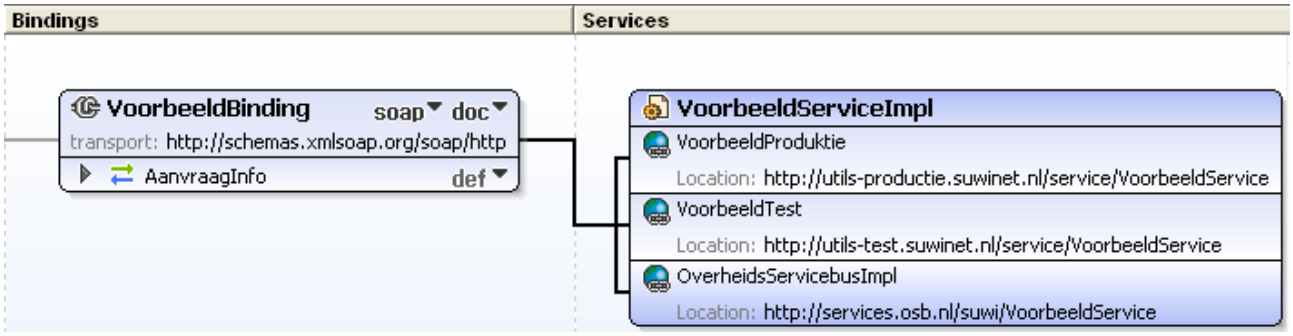
In Afbeelding 8 blijkt het gebruik van WS-Addressing uit het element <wsaw:UsingAddressing .../>. Voor verdere informatie over het gebruik van WS-Addressing, zie 4.8.1 Ondersteuning voor WS-Addressing.

Voorheen waren er in de Suwi-keten zelf-gedefinieerde headers om stuurinformatie met de berichten mee te sturen. Mocht er voor een bepaalde webservice nog gebruik gemaakt worden van die Suwi-specifieke stuurinformatie, dan dient dat in de WSDL koppelvlak specificaties van de webservice kenbaar gemaakt te worden zoals in Afbeelding 8 met het Element <soap:header .../>.

Afspraak 11: Eventueel gebruik van SuwiML stuurinformatie voor een webservice wordt in de WSDL koppelvlak specificaties van die webservice vastgelegd

4.1.4 De WSDL Service

Waar de WSDL PortType de functionele aspecten van de koppelvlak definitie (vrije vertaling van 'interface') van een webservice beschrijft, staat de WSDL Service voor een concrete implementatie van die koppelvlak definitie. De WSDL Binding vormt de verbinding tussen de twee. De WSDL Service bestaat uit één of meer endpoints, voorzien van een URL http(s) adres, waarop de implementatie(s) bereikbaar zijn.



Afbeelding 9 Verschillende implementaties met hun URL's

4.2. Inkijk versus Meldingen

In de Suwi-keten wordt van oudsher onderscheid gemaakt tussen services van het type 'Inkijk' en services van het type 'Meldingen'.

- Inkijk (ook bekend als 'Inlezen'): het opvragen van informatie, op basis van een sleutel-gegeven zoals bijvoorbeeld het Burger Servicenummer.
- Melding: het doorgeven van informatie naar aanleiding van een gebeurtenis (verhuizing, status-wijziging, correctie-verzoek, ...) en bij wijze van signaal.

Bij een webservice van het type Inkijk zit er een sleutel-gegeven in het request, en de bijbehorende gevraagde informatie wordt terug gegeven in het response. Het sturen van nogmaals hetzelfde request zal leiden tot het ontvangen van hetzelfde response. Een Inkijk service wordt vaak synchroon geïmplementeerd (de informatie wordt meteen in dezelfde HTTP-connectie mee terug gestuurd).

Bij een webservice van het type Melding zit de te leveren informatie in het request, dus in het heengaande bericht, en het response (het terugkomende bericht) zal vaak een soort Ontvangst-Bevestiging zijn. Het sturen van een Melding-request zal normaal gesproken bij de partij die het Melding-request ontvangt leiden tot een wijziging in hun systemen, ofwel het in gang zetten van een bedrijfsproces. Als dat allemaal snel en real-time afgehandeld kan worden dan kan de uitslag daarvan al gelijk in de Ontvangstbevestiging-response terug gemeld worden. Vaak kan dat echter niet (bijvoorbeeld omdat er menselijke handelingen nodig zijn, of omdat het Melding-request doorgestuurd moet worden naar nog een andere partij). In dat geval komt er eerst een kale Ontvangstbevestiging terug en pas op een later tijdstip een echt inhoudelijk antwoord. Bij Meldingen is er veel meer aandacht nodig, dan bij Inkijk, voor het afhandelen van herhaalde pogingen met hetzelfde Melding-request: het betreffende bedrijfsproces of het verwerken van de wijziging is immers misschien inmiddels al in gang gezet. Er is ook meer aandacht nodig voor Betrouwbaarheid: de partij die het Melding-request verstuurd heeft moet er van op aan kunnen dat zijn Melding doorkomt en verwerkt wordt, zelfs als dat moet gebeuren in systemen die misschien niet permanent in de lucht zijn. Dit zijn redenen dat er bij Meldingen vaak voor een asynchrone implementatie gekozen wordt (al dan niet van RINIS),inclusief store-and-forward faciliteiten. Een belangrijk deel van de stuurinformatie in de SuwiML Header is noodzakelijk om ook dit soort implementaties te ondersteunen.

Het onderscheid tussen Inkijk en Meldingen is echter minder groot dan het lijkt: ook bij een eenvoudig Inkijk request kan er bij de ontvangende partij een proces gestart worden, en andersom, ook een service van het type Melding kan synchroon geïmplementeerd worden. De SuwiML Transactiestandaard ondersteunt in principe alle varianten, en maakt dus geen streng onderscheid tussen Inkijk en Meldingen. Wel geldt volgens Afspraak 7 dat ieder request-bericht binnen dezelfde HTTP-connectie beantwoord wordt met een response-bericht. Dus dat geldt ook voor Melding-requests, al zal het response-bericht dan vaak slechts een kale Ontvangstbevestiging bevatten, en geen verslag van de verdere afhandeling van het Melding-request.

Het verschil tussen Inkijk en Meldingen zit hem niet zo zeer in de bericht-uitwisseling, als wel in de implementaties van de partijen. Een applicatie die een SOAP bericht stuurt ten behoeve van een Raadpleging door een gebruiker, zal, als hij geen antwoord van de webservice krijgt, de gebruiker een foutmelding voorschotelen. Het is dan aan de gebruiker of hij die Raadpleging nog een keer wil proberen. In het geval van een Melding-bericht kan de applicatie die het SOAP bericht verstuurt (de Client applicatie) zich er niet zo makkelijk vanaf maken. Bij de partij die het bericht moet ontvangen, moet een verwerking gestart worden, en zolang de Client applicatie geen zekerheid heeft dat de Melding bij de webservice aangekomen is, moet hij de Melding opnieuw blijven aanbieden. En wel met hetzelfde Bericht-Id want de ontvangende webservice moet ook kunnen zien of hij die Melding al verwerkt heeft of niet. Ook na een herstart (gepland of niet) moet de Client applicatie daartoe in staat zijn. Dus in de tussentijd moet de Melding bijvoorbeeld in een database opgeslagen zijn.

Afspraak 12: Een applicatie die een Melding-Bericht verstuurt naar een webservice moet dat Melding-Bericht blijven herverzenden (met hetzelfde MessageID) totdat hij een bijbehorend Acknowledgement-Bericht heeft terug ontvangen

Het kan ook voorkomen dat het Acknowledgement-Bericht op de terugweg verloren gaat of te laat komt. De partij die de Melding verstuurd had, heeft dan geen zekerheid dat zijn Melding is aangekomen, en zal de Melding nogmaals versturen. Bij de ontvangende partij is inmiddels een Procedure of Verwerking naar aanleiding van de Melding gestart.

Die Procedure of Verwerking hoeft niet nogmaals gestart te worden, maar wel dient de eerder verstuurde Acknowledgement nogmaals teruggestuurd te worden.

Afspraak 13: Een applicatie die een eerder ontvangen Melding-Bericht nogmaals ontvangt dient hetzelfde eerder teruggestuurde Acknowledgement-Bericht nogmaals terug te sturen.

Het herverzenden van een Melding-bericht dient op realistische tijdsintervallen te gebeuren. Er kunnen verschillende redenen zijn dat het Acknowledgement de eerste keer niet terug ontvangen werd. Misschien had de server-zijde een tijdelijk probleem. Dan heeft het geen zin om overmatig veel herhalingsberichten te sturen. Hoe vaak en hoe frequent er een herhalingsbericht verstuurd dient te worden kan echter niet in de koppelvlak specificaties vastgelegd worden. Dit dient apart te worden afgesproken tussen de betrokken partijen en vastgelegd te worden in de Implementatie-afspraken.

In een herhalingsbericht krijgt in de Header het element <wsa:MessageID> dezelfde waarde als het originele bericht. Het element <smlh:DatTijdAanmaakRequest> krijgt wel een nieuwe waarde om het herhalingsbericht te kunnen onderscheiden van het originele bericht.

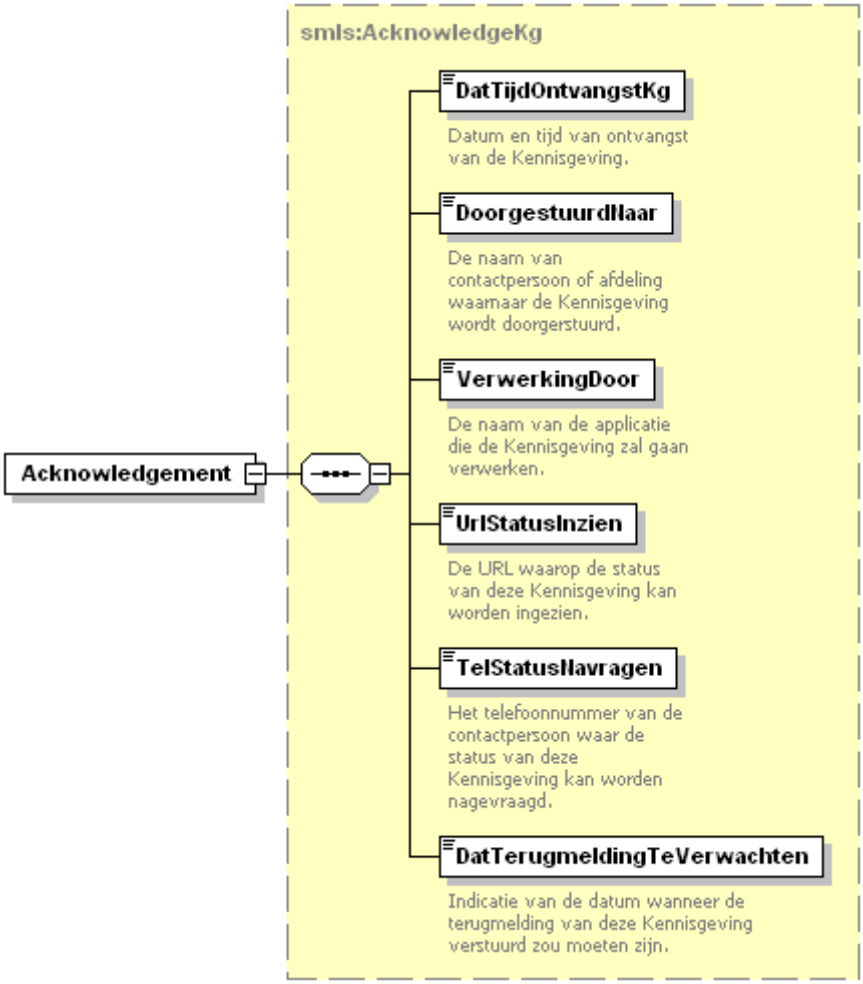
Er zijn wel mogelijkheden denkbaar om ook in de Bericht-uitwisseling betere ondersteuning voor 'betrouwbaar berichtenverkeer' te bieden. Deze versie en eerdere versies van de Transactiestandaard bieden dat nog niet. In de toekomst is dat wel mogelijk bijvoorbeeld door ondersteuning voor de WS-ReliableMessaging standaard er in op te nemen.

Voor degenen die de koppelvlak specificaties opstellen van een webservice die Meldingen gaat ontvangen is het raadzaam om de Melding-berichten 'idempotent' te maken. Idempotent betekent dat meerdere keren sturen van het Bericht hetzelfde effect heeft als één keer sturen. De mededeling “Mevrouw X krijgt 10 % salarisverhoging” is niet idempotent, maar de mededeling “Mevrouw X gaat € 4000 verdienen” wel. Bij idempotente Meldingen zijn bovengenoemde maatregelen voor de implementaties minder dringend noodzakelijk (maar ook weer niet overbodig).

Voor meer informatie omtrent het implementeren van Melding-berichten, zie ook 9.1 Triggers, Signalen en (Terug-)Meldingen.

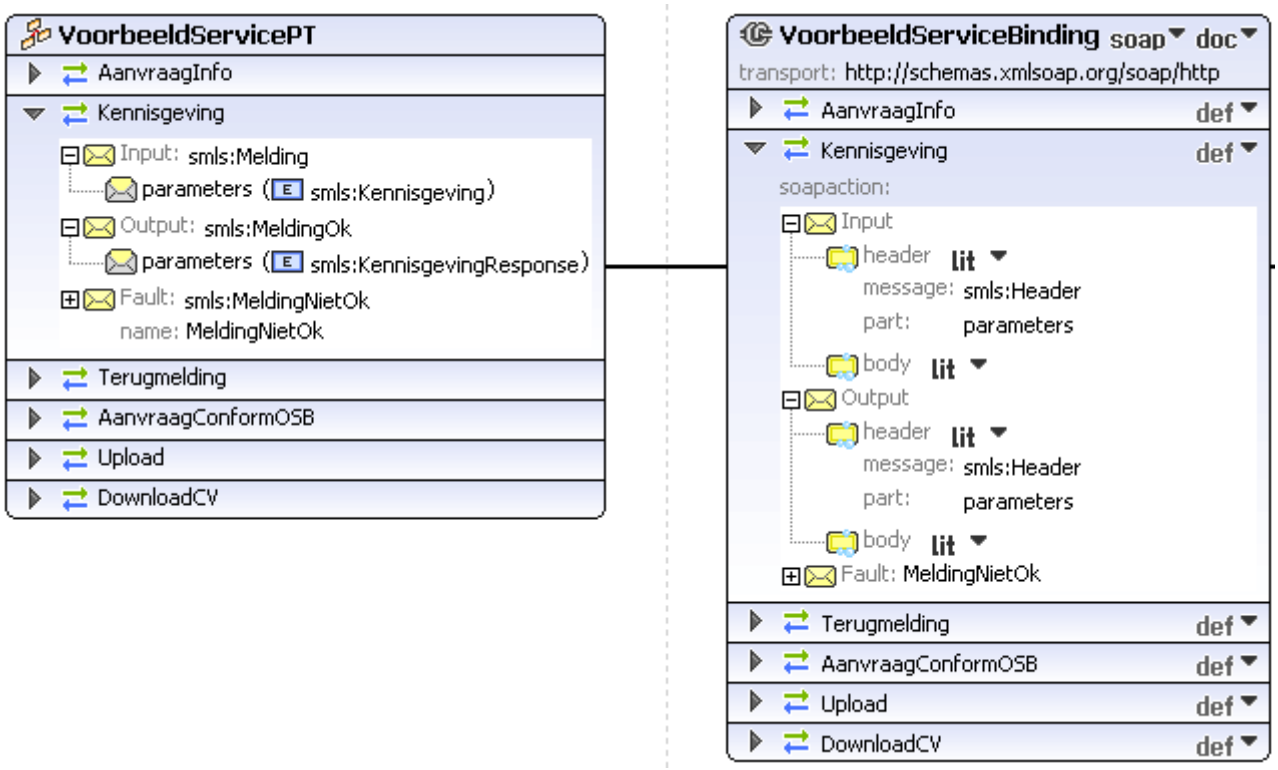
4.3. De Ontvangstbevestiging

Er bestaat de mogelijkheid om in de Ontvangstbevestiging informatie over de afhandeling van de Melding of het Signaal mee terug te geven. Het is bijvoorbeeld mogelijk om een URL terug te sturen waarop de status van de afhandeling van de Melding of het Signaal op te vragen is.



Afbeelding 10 Voorbeeld van een ontvangstbevestiging

De precieze invulling van een Ontvangstbevestiging voor een bepaalde service wordt vastgelegd in de WSDL en de bijbehorende XML Schema's.



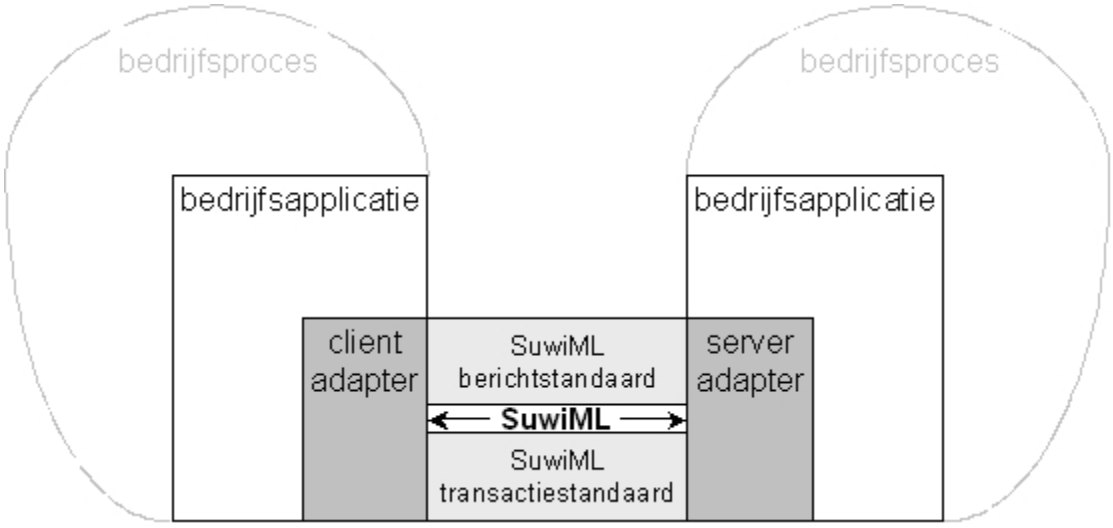
Afbeelding 11 De ontvangstbevestiging-response bij een Meling-request (voorbeeld)

4.4. Brokers en andere tussenstantions

Er kunnen verschillende redenen zijn om een bepaalde service op meerdere plaatsen aan te bieden. Er wordt bijvoorbeeld vaak gebruik gemaakt van een Broker bij wijze van centrale toegangspoort tot allerlei achterliggende services. Deze services worden dan dus zowel geleverd door de achterliggende systemen als ook door de broker, namens de achterliggende services. Een dergelijke opzet kan voordelen bieden op het gebied van hergebruik, caching en het inregelen van autorisaties. Zowel UWV, als UWV Werkbedrijf als de centrale Suwinet Inkijk omgeving maken gebruik van een Broker (UWV gebruikt de Klantbeeldserver (KBS), UWV Werkbedrijf een Oracle ESB, en Suwinet Inkijk de Suwi Broker). Het Sectorloket van het Inlichtingenbureau fungeert ook als een soort Broker voor de achterliggende services (die voor de Elektronische Keten Berichten) van de verspreide Gemeentelijke systemen. Het kan ook voorkomen dat een service op verschillende netwerken aangeboden wordt, bijvoorbeeld zowel op het besloten Suwinet als op het openbare Internet. Tenslotte kan een partij er voor kiezen om alle zorgen op het gebied van connectiviteit, monitoring, beveiliging, enz. uit handen te geven en het transport te laten verzorgen door een derde partij.
RINIS is een partij die dat soort dingen doet door een webservice van een partij op afstand, te implementeren op een RINIS server lokaal bij de vragende partij. Dat is dan ook een voorbeeld van een service die op meerdere plaatsen geïmplementeerd is.

De SuwiML Transactiestandaard ondersteunt alle genoemde situaties. In de SuwiML Header is een gedeelte 'RouteInformatie' opgenomen. Een Broker of een RINIS server of een ander Tussenstation kan een request doorzetten naar een achterliggende partij en in de RouteInformatie van het doorgestuurde request opnemen dat zij als Tussenstation opgetreden is. Voor meer details over de RouteInformatie, zie 5.3 Route-Informatie.

4.5. SOAP adapters



Afbeelding 12 Schematisch beeld van het gebruik van SuwiML bij de gegevensuitwisseling tussen client en server (voorbeeld)

In de praktijk komt het vaak voor dat de SOAP communicatie afgehandeld wordt door een speciale SOAP Adapter, terwijl de functionele inhoud van het SOAP bericht bestemd is voor een achterliggende andere bedrijfs-applicatie. De communicatie tussen de adapter en de bedrijfs-applicatie zal vaak geen SOAP zijn en misschien zelfs geen XML. De adapter transformeert van het interne protocol naar het protocol van de keten, en andersom. Wanneer een partij over een broker beschikt dan zal die broker de rol van adapter op zich nemen. De SuwiML Transactiestandaard gaat over de communicatie tussen de broker / adapter van de client partij en die van de server partij, en niet over de interne communicatie bij een partij tussen de broker / adapter en de bedrijfs-applicatie.

4.6. Verschillende versies

Voor iedere SuwiML webservice kunnen meerdere versies in omloop zijn. Echter iedere request-response-acknowledgement-fout set wordt door precies één en dezelfde SuwiML webservice-specificatie bepaald. Het versienummer wordt opgenomen in de targetnamespace van de WSDL en komt ook terug in de namespace van de body van het Request en het Response. Bij ieder request van een Client applicatie is dus voor de Server applicatie herkenbaar voor welke versie van de webservice het request bedoeld is.

In de naam van het .zip pakket met de WSDL en de bijbehorende XML Schema files staat naast het versie-nummer nog een build-nummer. Het build-nummer wordt ***niet*** opgenomen in de targetnamespace en in de namespace van de body van het Request en het response. Gedurende het ontwikkel- en testtraject van een nieuwe (versie van een) webservice kan naar aanleiding van de bevindingen het build-nummer nog veranderen.

4.7. SOAP toolkits

Iedere partij bouwt zijn eigen implementatie van de SuwiML webservices die zij wensen te gebruiken. Zij kunnen dat doen met zelf te kiezen tools en ontwikkelomgevingen. Voor ieder modern platform en iedere ontwikkelomgeving is er ondersteuning voor webservices beschikbaar. De ontwikkelaar krijgt een Application Program Interface (API) aangeboden waarmee SOAP berichten kunnen worden samengesteld en verstuurd. De details van het SOAP protocol worden

tot op zekere hoogte voor de ontwikkelaar verborgen. Er is geen standaard SOAP API, i.e. de API verschilt per toolkit.

De SuwiML webservice koppelvlakken worden echter ontworpen met interoperabiliteit tussen die verschillende toolkits als belangrijk onderliggend principe. De meest gebruikte ontwikkelomgevingen (Java Axis2, Oracle ESB, Microsoft .Net, SUN Metro, ...) moeten goed met de opgeleverde koppelvlakken overweg kunnen.

4.8. Stuurgegevens

In een WSDL koppelvlak-beschrijving wordt meegegeven welke soort stuurgegevens er van belang zijn voor de webservice. In vorige versies van de SuwiML Transactiestandaard werd er alleen de custom-made SuwiML Header gebruikt. Inmiddels ontstaan er ook internationale standaarden voor bepaalde soorten stuurgegevens, zoals WS-Addressing en WS-ReliableMessaging. Waar mogelijk zullen we in deze en toekomstige versies van de SuwiML Transactiestandaard onderdelen uit de SuwiML Header vervangen door soortgelijke headers van de relevante WS-* en/of andere algemeen geldende standaarden. In deze versie betreft dat alleen nog headers die door WS-Addressing gedekt worden.

4.8.1 Ondersteuning voor WS-Adressing

De standaard Web Services Addressing 1.0 - Core (W3C Recommendation 9 May 2006) bepaalt welke velden er in de header van een bericht meegestuurd dienen te worden ten behoeve van adequate adressering van het bericht. In de specificatie “Web Services Addressing 1.0 - WSDL Binding” (W3C Candidate Recommendation 29 May 2006) wordt aangegeven hoe in een WSDL beschrijving het gebruik van WS-Addressing gestuurd kan worden.

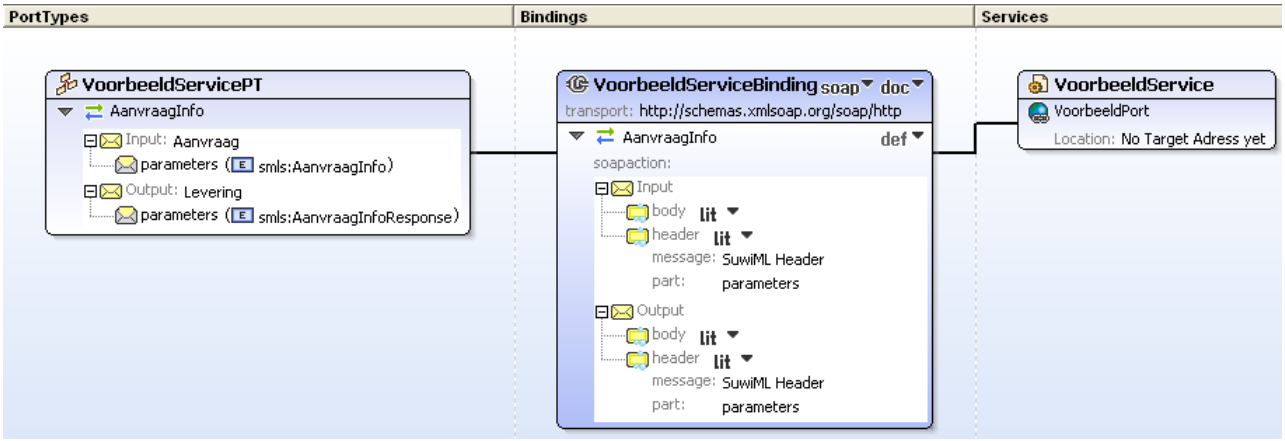
Het element <wsaw:UsingAddressing wsdl:required="true"/> wordt in iedere WSDL toegevoegd als child van het element <wsdl:Definitions> / <wsdl:Binding>, bij wijze van indicatie dat het gebruik van WS-Addressing verplicht is. De prefix wsaw staat voor de namespace http://www.w3.org/2006/05/addressing/wsdl.

Ieder <wsdl:Input> element in een WSDL beschrijving krijgt een attribuut wsaw:Action. Iedere client implementatie dient de waarde van het attribuut wsaw:Action te gebruiken als waarde voor het element <wsa:Action> in elk bijbehorend request.

Ieder <wsdl:Output> element in een WSDL beschrijving krijgt ook een attribuut wsaw:Action. Iedere server implementatie dient de waarde van het attribuut wsaw:Action te gebruiken als waarde voor het element <wsa:Action> in elk bijbehorend response.

Zie verder 5.2 Adressering voor de gevolgen van WS-Addressing voor de afzonderlijke Berichten.

4.8.2 De SuwiML Header



Afbeelding 13 De SuwiML Header vastgelegd in een WSDL file

In de WSDL beschrijving van een SuwiML webservice is er in de Binding sectie de mogelijkheid om aan te geven dat er een SuwiML Header verwacht wordt voor het Input bericht en/of voor het Output bericht.

NB: het opnemen van een SuwiML Header in de WSDL is geen verplichting! Als we bijvoorbeeld een webservice beschrijven die Digikoppeling-compliant moet zijn, dan mag daar ***niet*** een SuwiML Header bij zitten. We zullen dus voortaan per webservice moeten bepalen òf er een SuwiML Header in de specificaties meegenomen wordt. Ook kan er dan in de koppelvlak specificaties vastgelegd worden welke onderdelen van de SuwiML Header er gebruikt zullen worden.

Afspraak 14: Als er voor een webservice een SuwiML Header gebruikt wordt dan wordt er in een service-specifiek SuwiMLHeader.xsd file vastgelegd welke onderdelen uit de SuwiML Header gebruikt worden

```
<wsdl:definitions ... xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
   <wsdl:types>
      ...
      <xs:schema targetNamespace="http://bkwi.nl/SuwiML/Header/v0400">
         <xs:include schemaLocation="SuwiMLHeader.xsd"/>
      </xs:schema>
   </wsdl:types>
   ...
   <wsdl:message name="Header">
      <wsdl:part name="parameters" element="smlh:SuwiMLHeader"/>
   </wsdl:message>
   <wsdl:portType name="VoorbeeldServicePT">
      <wsdl:operation name="AanvraagInfo">
         <wsdl:input message="Aanvraag"/>
         <wsdl:output message="Levering"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="VoorbeeldServiceBinding" type="VoorbeeldServicePT">
      ...
      <wsdl:operation name="AanvraagInfo">
         <soap:operation soapAction=""/>
         <wsdl:input>
            <soap:body use="literal"/>
            <soap:header message="Header" part="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
            <soap:header message="Header" part="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   ...
</wsdl:definitions>
```

Afbeelding 14 De SuwiML Header vastgelegd in een WSDL file (voorbeeld)

4.9. Ondersteuning voor binaire bestanden

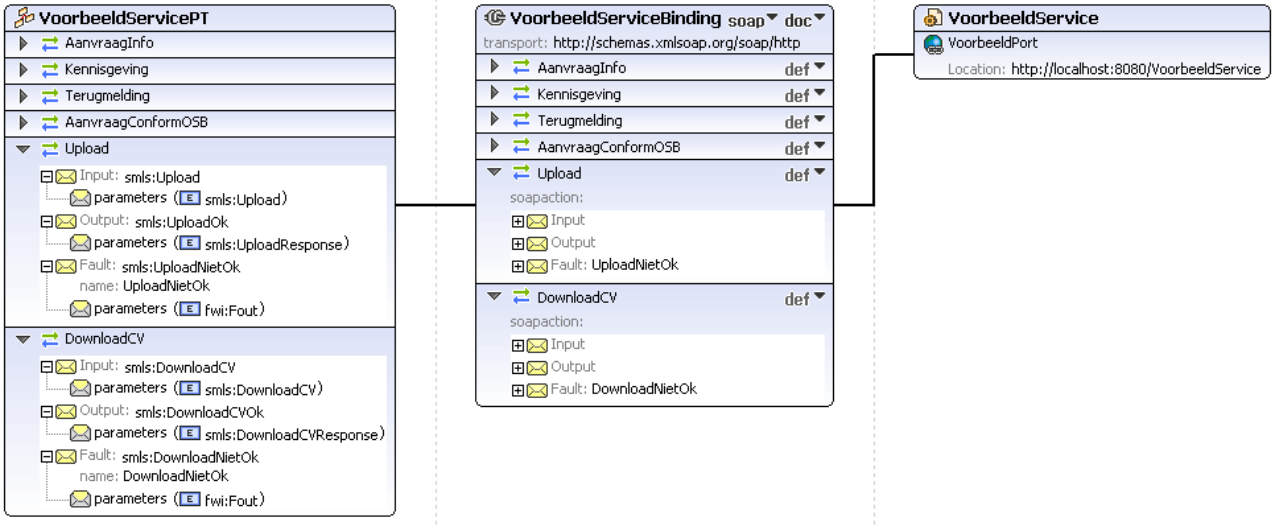
Niet alle data leent zich er voor om in XML formaat verstuurd te worden. Er zijn webservices denkbaar, ook in de Werk en Inkomen keten, die bij de te versturen data een bestand zoals een CV, een Diploma of een Foto zouden willen meesturen. In de WSDL van een dergelijke webservice kan dat aangegeven worden door het gebruik van elementen van het type “xmime:base64Binary”.

Bijvoorbeeld:

```
<element name="Foto" xmime:expectedContentTypes="image/jpeg" type="xmime:base64Binary"/>
<element name="CV" xmime:expectedContentTypes="application/pdf" type="xmime:base64Binary"/>
```

Per mee te leveren bestand dient dus ook het mime-type (image/jpeg, image/png, application/pdf, ...) van het bestand vermeld te worden.

In de koppelvlak specificaties van de SuwiML webservices zal het meeleveren van binaire bestanden ook op deze manier worden vormgegeven. Zie ter illustratie de operations “Upload” en “DownloadCV” van de VoorbeeldService specificaties op http://bkwi.nl/SuwiML/Diensten/VoorbeeldService.



Afbeelding 15 De operations "Upload" en "DownloadCV als voorbeeld voor binaire bestanden in een webservice

In de file VoorbeeldService.wsdl wordt voor het Request van de operation “Upload” verwezen naar het element “smls:Upload”. En voor de het Response van de operation “DownloadCV” wordt verwezen naar het element “smls:DownloadCV”. Het element “smls:Upload” wordt gedefinieerd in de XML Schema file BodyAction.xsd:

```
<<xs:element name="Upload">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Burgerservicenr" type="sml:Burgerservicenr"/>
         <xs:element name="Foto" type="xmime:base64Binary"
                        xmime:expectedContentTypes="image/jpeg, image/png"/>
         <xs:element name="CV" type="xmime:base64Binary"
                        xmime:expectedContentTypes="application/pdf"/>
         <xs:element name="Diploma" type="xmime:base64Binary"
                        xmime:expectedContentTypes="application/pdf"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Afbeelding 16 Voorbeeld van elementen van het type "xmime:base64Binary"

Ook het element “smls:DownloadCV”, dat wordt gedefinieerd in de XML Schema file BodyReaction.xsd, bevat een dergelijk subelement van het type “xmime:base64Binary”. Met het attribuut xmime:expectedContentTypes wordt aangegeven wat voor type binair bestand er met dit element verstuurd mag worden.

De aanwijzing xmime:expectedContentTypes="..." in de WSDL dwingt ***niet*** af dat er in de praktijk ook daadwerkelijk altijd een bestand van dat type verstuurd wordt. Er zou in de praktijk misbruik gemaakt kunnen worden van een dergelijke webservice door een (mogelijk kwaadaardig) bestand van een ander type te voorzien van een extensie die door de webservice geaccepteerd wordt. De partijen die de Koppelvlak specificaties implementeren dienen zich daarvan bewust te zijn, de risico's in te schatten en eventuele voorzorgsmaatregelen te nemen.

De aanwijzingen in de WSDL zeggen niets over de maximale grootte van de bij te voegen bestanden. Ook voor dat aspect is het dus van belang dat de partijen die de webservice implementeren de grenzen van hun eigen systemen kennen, en dat ze eventueel voorzorgsmaatregelen nemen. Aan de HTTP parameter “Content-Length” is bijvoorbeeld wel te zien hoe groot een inkomend bericht is.

Zie verder 5.9 Berichten met Binaire Bestanden voor een bespreking van de gevolgen van dit soort koppelvlak specificaties voor de bijbehorende XML berichten.

5. SuwiML berichten

Een SuwiML Bericht is een request gericht aan een SuwiML webservice ofwel een response afkomstig van een SuwiML webservice. Een SuwiML Bericht bestaat uit:

- Een SOAP Envelope
- Met in de SOAP Envelope
  - Een SOAP Header met stuurgegevens conform deze Transactiestandaard en de bij de service horende koppelvlak specificaties;
  - En een SOAP Body met de applicatie-specifieke bericht-inhoud conform de bij de service horende XML-Schema's, en opgebouwd volgens de SuwiML Berichtstandaard.

De header wordt volgens de richtlijnen van deze Transactiestandaard gevuld met stuurgegevens ten behoeve van adressering, routering, bericht-identificatie, gegarandeerde aflevering en transactie-management. Waar mogelijk en waar van toepassing maken we voor de stuurinformatie gebruik van geaccepteerde internationale standaarden. Op dit moment betreft dat de standaard WS-Addressing 1.0 voor de tags <wsa:MessageId>, <wsa:Action>, <wsa:To> en <wsa:From>.
Voor stuurinformatie die (nog) niet gedekt wordt door deze Internationale standaarden is er nog een eigen apart SuwiML header XML-Schema met een eigen namespace. Per bericht wordt er ten tijde van het vaststellen van de koppelvlak-specificaties in een bericht-specifiek header schema vastgelegd welke onderdelen van de SuwiML headers er voor dat bericht gebruikt worden.
Afbeelding 17 toont de inbedding van de stuurgegevens in de SOAP Envelope. Een exacte XML specificatie van de SuwiML Header is te zien in Header-v0400.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:To>xs:anyURI</wsa:To>
    <wsa:From><wsa:Address>xs:anyURI</wsa:Address></wsa:From>
    <wsa:Action>xs:anyURI</wsa:Action>
    <wsa:MessageID>xs:anyURI</wsa:MessageID>
    <smlh:SuwiMLHeader xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
      <RouteInformatie>
        <Tussenstation> ... </Tussenstation>
      </RouteInformatie>
      <BerichtIdentificatie> ... </BerichtIdentificatie>
      <Transactie> ... </Transactie>
    </smlh:SuwiMLHeader>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <smls:AanvraagInfo xmlns:smls="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService">
      <Burgerservicenr>String</Burgerservicenr>
    </smls:AanvraagInfo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Afbeelding 17 Inbedding van de stuurgegevens in de SOAP envelope

Tabel 5 tot en met Tabel 10 beschrijven in detail de inhoud en structuur van de beschikbare onderdelen van de SuwiML header. Per stuurelement wordt een definitie gegeven. Zoals ook in Afbeelding 17 is te zien, is de SuwiML header verdeeld in drie logische blokken, te weten: RouteInformatie, BerichtIdentificatie en Transactie. Per service wordt in de koppelvlak specificaties vastgelegd welke van de blokken gebruikt worden.

5.1. Identificatie van partijen / componenten / applicaties

Het identificeren van partijen, componenten en applicatie wordt gedaan aan de hand van een DistinguishedName (DN). Deze regels zijn vertaald in de specificatie van SuwiML-Header-v0400.

```
<soap:Header>
   ...
   <smlh:RouteInformatie xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
      <OrigineleBron>
         <UserDN>ou=GSD, o=Gemeente Breda,c=nl</UserDN>
         <ApplicatieDN>cn=GWS4all,o=Centric,c=nl</ApplicatieDN>
      </OrigineleBron>
      <Bron>
         <ApplicatieDN>cn=GWS4all,o=Centric,c=nl</ApplicatieDN>
      </Bron>
      <Bestemming>
         <ApplicatieDN>cn=Sectorloket,o=Inlichtingenbureau,c=nl</ApplicatieDN>
      </Bestemming>
   </smlh:RouteInformatie>
   …
</soap:Header>
```

Afbeelding 18 Distinguished Name identificatie in <smlh:Bron>

De twee DN's in de OrigineleBron, User-DN en Applicatie-DN, maken onderdeel uit van de authenticatie en autorisatie van de combinatie applicatie en afnemer (user).

DistinguishedName's spelen een belangrijke rol in LDAP Directory's en X509 Certificaten PKI Overheid certificaten). Omdat PKI certificaten de basis zijn voor de identificatie van overheidsinstanties, -applicaties, is deze methode overgenomen in de SuwiML Header.

5.2. Adressering

Voor stuurinformatie met betrekking tot Adressering maken alle nieuwe services verplicht gebruik van de WS-Addressing standaard, zie http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/. In '4.8.1 Ondersteuning voor WS-Addressing' werd besproken hoe het gebruik van WS-Addressing in de WSDL kenbaar wordt gemaakt. In deze versie van Transactiestandaard wordt geadviseerd om in de headers van de berichten tags als <wsa:MessageID>, <wsa:Action> en <wsa:RelatesTo> te vullen, afhankelijk van de implementatie-afspraken (dit is de waarde conditioneel in de tabellen van 5.2.1 en 5.2.2). Voor de exacte definities van de tags verwijzen we naar het officiële bovengenoemde document.

In deze paragraaf wordt aangegeven welke velden we van de WS-Addressing gebruiken binnen de keten. In de WS-Addressing standaard wordt echter niet bepaald welke elementen verplicht zijn en in welke volgorde deze moeten voorkomen.
De optionele elementen kunnen weggelaten worden in de berichten. Het niet beschrijven van deze optionele elementen is omdat de optionele elementen vooralsnog geen functie hebben binnen de keten W&I.
Maar als een bepaalde implementatie ze toch wenst op te nemen, dan gelden er vaste waarden. Dit om het gedrag van de diverse implementaties voorspelbaar te houden. Het gedrag zal dan ook hetzelfde zijn als wanneer deze elementen niet in de headers van de berichten opgenomen zouden zijn.

Ieder bericht krijgt een Id mee in de vorm van het element <wsa:MessageID>, te vullen met een door iedere partij zelf te kiezen unieke URI. In een response-bericht dient met het element <wsa:RelatesTo> verwezen te worden naar het bijbehorende request-bericht. Dus het element <wsa:RelatesTo> in het Response heeft dezelfde waarde als <wsa:MessageID> uit het bijbehorende Request. In de WSDL beschrijving van de service staat verder geen informatie met betrekking tot de vulling van de elementen <wsa:MessageID> en <wsa:RelatesTo>.

Ieder bericht heeft een element <wsa:Action> van het type xs:anyURI. De URI die in een bericht ingevuld moet worden, wordt bepaald door en vastgelegd in de WSDL beschrijving van de webservice. Een Request , een Response en een Fout-Bericht horende bij één webservice hebben ieder een eigen waarde voor <wsa:Action>. Het element <wsa:Action> komt in de plaats van de HTTP parameter SOAPAction. De HTTP parameter SOAPAction krijgt voortaan de lege waarde “”.

Note: Alle elementen WS-Adressing zijn volgen de standaard optioneel behalve <wsa:Action> en mogen in willekeurige volgorde in de header voorkomen. De implementatie bepaalt of elementen gebruikt moeten worden of optioneel zijn.

5.2.1 Toepassing in de synchrone gegevensuitwisseling

Dit gedeelte beschrijft hoe WS-Addressing-elementen in de synchrone gegevensuitwisseling binnen de keten worden geïmplementeerd.

Tabel 1 Implementatierichtlijnen WS-Adressing elementen voor een synchroon request

Tag	Verplichtingen: V/C/O	Vulling	Functie	Bron
<wsa:MessageID>	Conditioneel	URI ter identificatie van een enkel request: http://afzender/ someuniquestring	Unieke identificatie van het bericht t.b.v. traceerbaarheid.	Te bepalen door de afzender
<wsa:Action>	Verplicht	URI ter identificatie van het input-bericht zoals gedefinieerd in de WSDL	Unieke identificatie zoals bepaald in de WSDL voor het Request, Response en Fout.	wsdl:definitions / wsdl:portType / wsdl:operation / wsdl:input / @wsaw:Action

De overige elementen, zoals <wsa:To>, <wsa:From>, <wsa:ReplyTo>, zijn optioneel. Met een verwijzing in de WSDL naar het regel: <wsaw:UsingAddressing wsdl:required="true"/>, hoeven de optionele elementen niet expliciet te worden uitgeschreven. Zie hierover in § 4.8.1

Tabel 2 Implementatierichtlijnen WS-Addressing elementen voor een synchroon Response op een synchroon Request

Tag	Verplichtingen: V/C/O	Vulling	Functie	Bron
<wsa:MessageID>	Conditioneel	URI ter identificatie van een enkel response: http://beantwoorder/so meother uniquestring	Unieke identificatie van het bericht t.b.v. traceerbaarheid.	Te bepalen door de zender van het Response

<wsa:Action>	Verplicht	URI ter identificatie van het output-bericht zoals gedefinieerd in de WSDL	Unieke identificatie zoals bepaald in de WSDL voor het Request, Response en Fout.	wsdl:definitions / wsdl:portType / wsdl:operation / wsdl:output / @wsaw:Action
<wsa:RelatesTo>	Conditioneel	URI ter identificatie van het bijbehorende request: http://afzender/someuniquestring	Een verwijzing naar de Identificatie van het bericht uit het bijbehorende Request	<wsa:MessageId> uit het Request

5.2.2 Toepassing in de asynchrone gegevensuitwisseling

Dit gedeelte beschrijft hoe WS-Addressing-elementen in de asynchrone gegevensuitwisseling binnen de keten worden geïmplementeerd.

Tabel 3 Implementatierichtlijnen WS-Addressing elementen voor een asynchroon Request

Tag	Verplichtingen: V/C/O	Vulling	Functie	Bron
<wsa:MessageID>	Conditioneel	URI ter identificatie van een enkel request: http://afzender/ someuniquestring	Unieke identificatie van het bericht t.b.v. traceerbaarheid.	Te bepalen door de afzender
<wsa:Action>	Verplicht	URI ter identificatie van het input-bericht zoals gedefinieerd in de WSDL	Unieke identificatie zoals bepaald in de WSDL voor het Request, Response en Fout.	wsdl:definitions / wsdl:portType / wsdl:operation / wsdl:input / @wsaw:Action
<wsa:To>	Conditioneel	URL van de service, zoals bepaald door de waarde van: <wsdl:definitions>/ <wsdl:service>/ <wsdl:port>/ <soap:address>/ @location in de bijbehorende WSDL	Dit wordt gebruikt voor de routering van het bericht.	Te bepalen door de afzender

De overige elementen, zoals <wsa:To>, <wsa:From>, <wsa:ReplyTo>, zijn optioneel. Met een verwijzing in de WSDL naar het regel: <wsaw:UsingAddressing wsdl:required="true"/>, hoeven de optionele elementen niet expliciet te worden uitgeschreven. Zie hierover in 4.8.1

Tabel 4 : Implementatierichtlijnen WS-Addressing elementen voor een asynchroon Response op een asynchroon Request

Tag	Verplichtingen: V/C/O	Vulling	Functie	Bron
<wsa:MessageID>	Conditioneel	URI ter identificatie van een enkel response: http://beantwoorder/someother uniquestring	Unieke identificatie van het bericht t.b.v. traceerbaarheid.	Te bepalen door de zender van het Response
<wsa:Action>	Verplicht	URI ter identificatie van het output-bericht zoals gedefinieerd in de WSDL	Unieke identificatie zoals bepaald in de WSDL voor het Request, Response en Fout.	wsdl:definitions / wsdl:portType / wsdl:operation / wsdl:output / @wsaw:Action
<wsa:RelatesTo>	Conditioneel	URI ter identificatie van het bijbehorende request: http://afzender/someuniquestring	Een verwijzing naar de Identificatie van het bericht uit het bijbehorende Request	<wsa:MessageId> uit het Request
<wsa:To>	Conditioneel	URI in de vorm van van een DN ter identificatie van de afzender van het Request	Ter identificatie van de afzender van het Request	<wsa:From>/ <wsa:Address> uit het Request
<wsa:From>/ <wsa:Address>	Optioneel	URI in de vorm van van een DN ter identificatie van de afzender van het Response	Ter identificatie van de afzender van het Response	<wsa:To> uit het Request

Mocht er in een bepaalde implementatie de optionele elementen toch geleverd worden, dan worden deze in de verwerking genegeerd. Het gedrag zal dan ook hetzelfde zijn als wanneer deze elementen niet in de headers van de berichten opgenomen zouden zijn.

De waarde van het optionele element <wsa:To> in een request-bericht wordt bepaald door de waarde van het attribuut <wsdl:definitions> / <wsdl:service> / <wsdl:port> /<soap:address> / @location in de bijbehorende WSDL. Gevolg is dat een client implementatie 'gewoon' de URL van de webservice in de <wsa:To> zal zetten. Dit is ook conform de Digikoppeling-standaard. Het element <wsa:To> heeft dus ook een wezenlijk andere invulling dan het element <smlh:RouteInformatie> / <smlh:Bestemming> uit de SuwiML header en kan dus ook niet als vervanger van die laatste dienen.

De waarde van de optionele elementen <wsa:From> / <wsa:Address> en <wsa:ReplyTo> / <wsa:Address> in een request-bericht worden gevuld met de standaard URI “http://www.w3.org/2005/08/addressing/anonymous”. Dit komt overeen met het Request – Response karakter van alle conform deze Transactiestandaard te definieren webservices. Wanneer er wèl een betekenisvolle URL in <wsa:From> of <wsa:ReplyTo> zou staan, dan zou dat suggereren dat de server het Response in een nieuwe HTTP-connectie naar die URL zou moeten sturen. En dat is niet de bedoeling, want strijdig met de keuze voor Request – Response operations.

In het response-bericht wordt het element <wsa:To>, indien aanwezig, gevuld met de standaard URI “http://www.w3.org/2005/08/addressing/anonymous”. Het element <wsa:From> / <wsa:Address>, indien aanwezig in het Response, wordt gevuld met de waarde van <wsa:To> uit het Request.

Het is toegestaan om alle WSA-elementen op te nemen omdat bij sommige toolkits het genereren van deze elementen niet onderdrukt kan worden. Hierbij geldt wel de beperking dat de waarde voor deze elementen het routeringsmechanisme niet verstoort.

5.3. RouteInformatie

Ten behoeve van Tracking en Tracing kan er voor bepaalde services de behoefte bestaan om de hele afgelegde weg van een bericht in de stuurinformatie van dat bericht zelf op te nemen en te kunnen zien. De SuwiML Header <smlh:RouteInformatie> biedt daar ruimte voor in de vorm van subelementen <smlh:OrigineleBron>, <smlh:Bron>, <smlh:Tussenstation> en <smlh:Bestemming>.

```
<smlh:RouteInformatie xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
   <OrigineleBron>
      <UserDN>ou=GSD, o=Gemeente Breda,c=nl</UserDN>
      <ApplicatieDN>cn=GWS4all,o=Centric,c=nl</ApplicatieDN>
   </OrigineleBron>
   <Bron>
      <ApplicatieDN>cn=GWS4all,o=Centric,c=nl</ApplicatieDN>
   </Bron>
   <Bestemming>
      <ApplicatieDN>cn=Sectorloket,o=Inlichtingenbureau,c=nl</ApplicatieDN>
   </Bestemming>
   <Tussenstation>
      <ApplicatieDN>cn=Rinis-box 1,o=RINIS,c=nl</ApplicatieDN>
      <Volgordenr>1</Volgordenr>
      <DatTijdOntvangstBericht>2007-12-17T09:27:14+01:00</DatTijdOntvangstBericht>
      <DatTijdVersturenBericht>2007-12-17T09:29:23+01:00</DatTijdVersturenBericht>
   </Tussenstation>
   <Tussenstation>
      <ApplicatieDN>cn=Rinis-box 2,o=RINIS,c=nl</ApplicatieDN>
      <Volgordenr>2</Volgordenr>
      <DatTijdOntvangstBericht>2007-12-17T09:30:47+01:00</DatTijdOntvangstBericht>
      <DatTijdVersturenBericht>2007-12-17T09:30:53+01:00</DatTijdVersturenBericht>
   </Tussenstation>
</smlh:RouteInformatie>
```

Afbeelding 19 Voorbeeld van RouteInformatie-gegevens

Voor iedere nieuwe service dient in de koppelvlak specificaties bepaald te worden òf er <smlh:RouteInformatie> aan de berichten toegevoegd zal worden. De keuze wordt vastgelegd in en moet blijken uit de WSDL beschrijving van de service en de bijbehorende service-specifieke header XML Schema's.

Voor iedere SuwiML webservice waarvoor het opnemen van <smlh:RouteInformatie> is afgesproken, zal de applicatie die het initiatief neemt tot het versturen van een SuwiML bericht horend bij die service, zichzelf als <smlh:Bron> vermelden.

Tabel 5 Inhoud en structuur van de SuwiML header - RouteInformatie - Bron

RouteInformatie				
Bron	Unieke logische aanduiding van organisatie, organisatieonderdeel en applicatie van wie het bericht afkomstig is. De Bron is verplicht voor ieder bericht, kan slechts 1x voorkomen en bestaat uit de volgende onderdelen:			
		Element	Status	Definitie
		ApplicatieDN	Verplicht	String in de vorm van een X509 DistinguishedName

Het kan voorkomen dat de eigenlijke OrigineleBron van een verzoek nog weer een andere partij is, bijvoorbeeld een Gebruiker van een website (zoals bijvoorbeeld Suwinet Inkijk). In dat geval heeft de Bron (Suwinet Inkijk) ook nog de mogelijkheid om die Gebruiker (of het Organisatie-onderdeel waar zij/hij bij hoort) te vermelden als <smlh:OrigineleBron>. Voor de uiteindelijk Bestemming van het Bericht kan dat gegeven van belang zijn in verband met Autorisatie en Logging.

Tabel 6 Inhoud en structuur van de SuwiML header - RouteInformatie - OrigineleBron

RouteInformatie				
OrigineleBron	Unieke logische aanduiding van organisatie, organisatieonderdeel en applicatie van wie het bericht afkomstig is. De OrigineleBron is optioneel voor ieder bericht, kan slechts 1x voorkomen en bestaat uit de volgende onderdelen:			
		Element	Status	Definitie
		UserDN	Verplicht	String in de vorm van een X509 DistinguishedName
		ApplicatieDN	Verplicht	String in de vorm van een X509 DistinguishedName

De Bron-applicatie hoeft niet per se te weten wat de eindbestemming van een Request is. Er kunnen zelfs meerdere eindbestemmingen zijn wanneer een Request opgesplitst wordt in meerdere Requests. De Broker van het Inlichtingenbureau doet dat bijvoorbeeld met verzoeken afkomstig van de Suwi Broker. Indien nodig wordt het verzoek opgesplitst in meerdere verzoeken die naar verschillende Gemeentelijke applicaties gestuurd worden. In dat geval kan de Suwi Broker volstaan met het noemen van de Inlichtingenbureau Broker als <smlh:Bestemming>.

Ingeval van een Response-, Fout-, of Acknowledgement-bericht, wordt de identificatie van de ‘Bestemming’ van het Response-, Fout-, of Acknowledgement-bericht gevuld met een exacte kopie van de identificatie van de ‘Bron’ van het oorspronkelijke requestbericht.

Tabel 7 Inhoud en structuur van de SuwiML header - RouteInformatie - Bestemming

RouteInformatie				
Bestemming	Unieke logische aanduiding van organisatie, organisatieonderdeel en applicatie aan wie het bericht geadresseerd is. De Bestemming is verplicht voor ieder bericht, kan slechts 1x voorkomen en bestaat uit de volgende onderdelen:			
		Element	Status	Definitie

		ApplicatieDN	Verplicht	String in de vorm van een X509 DistinguishedName

Alle tussenliggende betrokken Brokers en Tussenstations kunnen zichzelf vermelden als <smlh:Tussenstation> in het door te sturen bericht. Het vastleggen van de Tussenstations welke gaande het transport worden aangedaan is optioneel. Per interface / gegevensuitwisseling wordt in de implementatie-afspraken bepaald of Tussenstations wel of juist niet vastgelegd zullen worden. De Tussenstations die tijdens het transport van een SuwiML bericht worden aangedaan, worden dus niet vooraf door de Bron (zender) bepaald en aan de SuwiML header toegevoegd. Ieder bezocht Tussenstation of Broker doet dat zelf.

Tabel 8 Inhoud en structuur van de SuwiML header - RouteInformatie - Tussenstation

<table>
<tr><th colspan="5">RouteInformatie</th></tr>
<tr><td>Tussenstation</td><td colspan="4">Unieke logische aanduiding van organisatie, organisatieonderdeel en applicatie door wie het bericht doorgestuurd is. Tussenstation is optioneel voor ieder bericht, kan meermaals voorkomen en bestaat uit de volgende onderdelen:</td></tr>
<tr><td></td><td></td><td>Element</td><td>Status</td><td>Definitie</td></tr>
<tr><td></td><td></td><td>ApplicatieDN</td><td>Verplicht</td><td>String in de vorm van een X509 DistinguishedName</td></tr>
<tr><td></td><td>èn</td><td>Volgordenr</td><td>Verplicht</td><td>PositiveInteger; Volgordenummer van het Tussenstation, oplopend en aansluitend genummerd. Bepaalt de volgorde waarin de Tussenstations zijn aangedaan. Het eerste Tussenstation stelt het volgordenummer op 1, ieder volgend Tussenstation hoogt het volgordenummer met 1 op.</td></tr>
<tr><td></td><td></td><td>DatTijdOntvangstBericht</td><td>Optioneel</td><td>DateTime; Datum en tijdstip waarop het Bericht is ontvangen door het Tussenstation.</td></tr>
<tr><td></td><td></td><td>DatTijdVersturenBericht</td><td>Optioneel</td><td>DateTime; Datum en tijdstip waarop het Bericht is doorgestuurd door het Tussenstation.</td></tr>
</table>

5.4. Berichtidentificatie

Naast de elementen <wsa:MessageId> en <wsa:RelatesTo> die als gevolg van het gebruik van WS-Addressing in de SOAP Header van een bericht voorkomen, biedt SuwiML nog de mogelijkheid om extra informatie op het gebied van BerichtIdentificatie in de header mee te sturen. Het betreft de tijdstippen van aanmaak en/of ontvangst van een bericht, en er kan specifieke applicatie-informatie toegevoegd worden.

```
<smlh:BerichtIdentificatie xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
   <DatTijdAanmaakRequest>2008-07-03T09:34:47+01:00</DatTijdAanmaakRequest>
   <ApplicatieInformatie>Suwinet Inkijk</ApplicatieInformatie>
</smlh:BerichtIdentificatie>
```

Afbeelding 20 Voorbeeld van Berichtidentificatie-gegevens in een Request

```
<smlh:BerichtIdentificatie xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
   <DatTijdOntvangstRequest>2008-07-03T09:34:50+01:00</DatTijdOntvangstRequest>
   <DatTijdAanmaakResponse>2008-07-03T09:34:51+01:00</DatTijdAanmaakResponse>
   <ApplicatieInformatie>UWV Klantbeeldserver</ApplicatieInformatie>
</smlh:BerichtIdentificatie>
```

Afbeelding 21 Voorbeeld van Berichtidentificatie-gegevens in een Response

Ook een Tussenstation of Broker dient deze velden in een door te sturen Request of Response in te vullen. Het Tussenstation zal in de Header van het door te sturen Request de Datum en Tijd vermelden dat hij zelf het door te sturen Request heeft aangemaakt. Dus het is niet de bedoeling dat het Tussenstation dat veld vult met de info die in het originele Request stond. En het Tussenstation zal in de Header van het door te sturen Response de Datum en Tijd vermelden dat hij zelf het doorgestuurde Request ontvangen had, en de Datum en Tijd dat hij zelf het door te sturen Response heeft aangemaakt. Dus het is niet de bedoeling dat het Tussenstation dat veld vult met de info die in het originele Response stond.

Voor iedere nieuwe service dient in de koppelvlak specificaties bepaald te worden of er volstaan wordt met de bericht-identificatie headers van WS-Addressing, of dat er <smlh:BerichtIdentificatie> aan de berichten toegevoegd zal worden. De keuze wordt vastgelegd in en moet blijken uit de WSDL beschrijving van de service en de bijbehorende service- specifieke header XML Schema's.

Tabel 9 Inhoud en structuur van de SuwiML header - BerichtIdentificatie

BerichtIdentificatie			
DatTijdAanmaakRequest	Optioneel	Te gebruiken in een (door te sturen) Request	DateTime
DatTijdOntvangstRequest	Optioneel	In een door te sturen Request en in een Response	DateTime
DatTijdAanmaakResponse	Optioneel	In een (door te sturen) Response	DateTime
DatTijdOntvangstResponse	Optioneel	In een door te sturen Response	DateTime
ApplicatieInformatie	Optioneel	In alle Berichten. Specifieke informatie (inclusief versie-aanduiding) van de applicatie die het bericht heeft aangemaakt	String, maxlength = 256

5.5. Transactiegegevens

SuwiML berichten kunnen worden voorzien van een transactie-identificatie <smlh:Transactie>. Afzonderlijke berichten die wel bij elkaar horen kunnen zo herkend worden. De berichten die bij een Transactie horen krijgen ook een Volgordenummer mee. Daarmee wordt het mogelijk om ontbrekende eerdere berichten te detecteren, en om een juiste verwerkingsvolgorde te realiseren.

```
<smlh:Transactie xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">
   <TransactieReferentienr>00000010-0000-0010-8000-00AA000BBBCC</TransactieReferentienr>
   <Volgordenr>5</Volgordenr>
   <IndLaatsteBericht>1</IndLaatsteBericht>
</smlh:Transactie>
```

Afbeelding 22 Voorbeeld van Transactiegegevens in een bericht

De partij die een nieuwe Transactie start bepaalt het <TransactieReferentienr> middels een zelf te implementeren functie die een GlobalUniqueIdentifier (GUID) genereert. De Bron stelt bovendien, bij aanvang van de Transactie, de <IndLaatsteBericht> op 2 (=nee) en het <Volgordenr> op 1. Ieder

volgend bericht dat door de Bron naar dezelfde Bestemming wordt gestuurd, en tot dezelfde Transactie behoort, krijgt hetzelfde <TransactieReferentienr> met een oplopend en aansluitend <Volgordenr>. De Bron zet bovendien de <IndLaatsteBericht> op 1 (=ja) als het betreffende bericht de laatste in de Transactie is.

Op deze wijze kan de Bestemming, bij ontvangst van de berichten, aan de hand van de <smlh:Transactie> bepalen in welke volgorde de ontvangen berichten verwerkt moeten worden. De Bestemming weet dus altijd of, ten opzichte van het laatst ontvangen bericht, alle qua volgorde daarvoor gelegen berichten van een bepaalde Bron reeds ontvangen zijn. De Bestemming kan met de verwerking van een bericht beginnen zodra alle qua volgorde daarvoor gelegen berichten reeds ontvangen en verwerkt zijn.

Tabel 10 Inhoud en structuur van de SuwiML Header – Transactie

Transactie			
TransactieReferentienr	Globaal uniek referentienummer van de transactie waar het bericht onderdeel van is. Een transactie bevat alle berichten die als een geheel en in de juiste volgorde verwerkt moeten worden.	Verplicht	datatype = xs:string maxlength = 100 Global unique identifier (GUID), gegenereerd door de initiërende bron
Volgordenr	Volgordenummer van het bericht binnen de transactie, oplopend en aansluitend genummerd.	Verplicht	datatype = xs:positiveInteger
IndLaatsteBericht	Aanduiding of dit het laatste bericht is binnen de transactie (ja, 1), of niet (nee, 2).	Optioneel	‘1’ (Ja) of ‘2’ (Nee)

Het gebruik van <smlh:Transactie> ondersteunt dus het in de juiste volgorde verwerken van berichten door de Bestemming. De Transactie doet echter geen uitspraak over en stelt dus ook geen eisen aan de volgorde waarin de berichten worden verstuurd door de Bron en/of worden ontvangen door de Bestemming.

Als de Bron geen gebruik maakt van <smlh:Transactie> en dus ook geen <Volgordenr> vermeldt, is de volgorde van verwerking blijkbaar niet relevant en mag de Bestemming het bericht direct na ontvangst verwerken.

Het ligt voor de hand om berichten die onderdeel uitmaken van een <smlh:Transactie> in één WSDL samen te voegen als verschillende 'operations' van een enkele webservice. Andersom dient voor iedere nieuwe webservice in de koppelvlak specificaties bepaald te worden of er gebruik gemaakt zal worden van <smlh:Transactie>. De keuze wordt vastgelegd in en moet blijken uit de WSDL beschrijving van de service en de bijbehorende service-specifieke header XML Schema's.

Het betreft hier trouwens niet het soort Transacties dat bij databases gangbaar is en die halverwege 'teruggedraaid' kunnen worden. 'Terugdraaien' van een eerder aangezwengelde bewerking kan in SOAP berichtenverkeer alleen maar met een apart bericht met de tegengestelde actie er in vermeld.

Voorbeeld: Voor een klant die een Aanvraag voor een WWB uitkering doet bij het UWV worden er twee berichten binnen één Transactie gestuurd vanuit Sonar van UWV Werkbedrijf naar de applicatie van de Gemeentelijke Sociale Dienst (via het Inlichtingenbureau): het bericht

AanvraagWWB (met <Volgordenr>1</Volgordenr> en <IndLaatsteBericht>2</IndLaatsteBericht>) en het bericht IntakeOverdrachtWWB (met <Volgordenr>2</Volgordenr> en <IndLaatsteBericht>1</IndLaatsteBericht>).

In de toekomst zal een dergelijk scenario wellicht vormgegeven gaan worden met behulp van de WS-ReliableMessaging standaard, in plaats van met <smlh:Transactie>.

5.6. Andere stuurgegevens

De Transactiestandaard biedt geen stuurgegevens voor de implementatie van bijvoorbeeld een 'two- phase commit'. Transacties kunnen niet teruggedraaid worden. Er kunnen wel tegengestelde transacties uitgevoerd worden om eventuele foute transacties recht te zetten.
Zodra er een behoefte kenbaar wordt gemaakt aan versleuteling of een digitale handtekening op applicatie-niveau dan zullen daarvoor ook stuurgegevens in de header opgenomen worden. De SuwiML Transactiestandaard zal zich in dat geval conformeren aan het WS-I Basic Security Profile, zie http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html. Op dit moment is die behoefte nog niet gesignaleerd en wordt volstaan met versleuteling en identificatie op transport-niveau door middel van SSL.

5.7. Valideren van een inkomend Request

Het valideren van een bericht dat binnenkomt bij een adapter (of een broker, of een applicatie die SOAP 'praat') gaat enigszins anders dan voorheen. In de huidige service-georiënteerde opzet moeten de volgende stappen doorlopen worden:

I. Parse het inkomende request
II. Verifieer dat het een SOAP bericht betreft door te valideren tegen het XML-Schema http://schemas.xmlsoap.org/soap/envelope/.
III. Verwerk de informatie uit de WS-Addressing elementen in de header. Hier wordt geen gebruik gemaakt van een schema, waarin vastgelegd is welke elementen moeten/mogen voorkomen. Hiervoor wordt gebruik gemaakt van het WS-Addressing schema zoals is voorgeschreven door W3C. De service moet zelf de implementatierichtlijnen zoals beschreven in 5.2 implementeren.
IV. Verwerk de informatie uit de SuwiML Header conform het schema van de SuwiML-Header.
V. Neem het eerste element uit de SOAP Body. Verifieer dat de namespace van dat element voorkomt in het lijstje van webservices dat de adapter ondersteunt
VI. Verifieer dat de naam van dat element een van de operations is van de desbetreffende webservice.
VII. Valideer het element met alles wat er onder zit tegen het schema dat in de WSDL geïmporteerd wordt
VIII. Verwerk de informatie uit de body.

Bij iedere bovengenoemde stap kan het proces fout gaan. Zie Hoofdstuk 6 Foutafhandeling voor de gewenste afhandeling van de diverse foutsituaties.

5.8. Diakrieten, karaktersets en encoderingen

Een veel voorkomend verschijnsel is dat er in (computer-)teksten tekens voorkomen als 'Ã©'. Vaak kun je wel raden wat er had moeten staan: 'AndrÃ©' had natuurlijk 'André' moeten zijn. Dit verschijnsel is altijd het gevolg van encoderings-fouten. Alle diakritische tekens (zoals 'é', 'ë', ...), het € teken, en alle andere niet-ASCII tekens zijn hier gevoelig voor. In het geval van de string

'AndrÃ©' heeft er kennelijk eerst een vertaling van de string naar bytes plaatsgevonden (bijvoorbeeld ten behoeve van opslag in een file, of voor transport over het netwerk) conform de encodering 'UTF-8'. Daarbij krijgt het karakter é een representatie in twee bytes, namelijk C3 (hexadecimaal, ofwel 11000011 binair) en A9 (ofwel 10101001). Vervolgens is van die reeks bytes weer een string gemaakt, maar nu conform een andere encodering (in dit geval ISO-8859-1, ook bekend als Windows Latin 1, die is berucht voor deze problemen). Daarbij wordt namelijk de byte C3 geïnterpreteerd als het karakter Ã en de byte A9 wordt geïnterpreteerd als het karakter ©.

Het ligt daarom voor de hand om enkele afspraken te maken op het gebied van karaktersets en encodering. We zullen daarbij aansluiten bij wat er internationaal gangbaar is.

Ten eerste kun je je afvragen welke karakters je allemaal zou willen toestaan: alleen die van de Westerse talen? Of ook Arabische tekens? Cyrillische, Chinese tekens? Inmiddels hebben echter al die karakters een plaats gekregen in de 'Unicode' karakterset. Unicode is ook bekend als 'Universal Character Set' of ook als 'ISO 10646'. Unicode wordt breed ondersteund: in de gangbare Besturingssystemen, in XML en HTML, in Java en .Net. Wikipedia zegt: “Unicode has become the dominant scheme for internal processing of text”. Dus ook al is de Unicode karakterset erg groot, en zullen heel veel karakters nooit gebruikt worden, er is geen reden om ons te beperken tot een bepaalde subset. De keuze voor een bepaalde subset zou ook tamelijk arbitrair zijn, en vroeg of laat niet blijken te voldoen.

Afspraak 15: Er gelden geen beperkingen aan de te gebruiken karakters anders dan dat ze tot de Unicode karakterset moeten behoren.

Bovendien is er voor Unicode een bijbehorende encodering, namelijk UTF-8: voor alle Unicode karakters biedt UTF-8 een representatie in bytes. Die representatie in bytes is nodig wanneer een tekst moet worden opgeslagen of verstuurd over een netwerk. UTF-8 is de internationale standaard voor email, webpagina's en webservices.

De bij het versturen van een SOAP bericht gebruikte encodering wordt vermeld in de HTTP parameter Content-Type:

```
Content-Type: text/xml; charset="UTF-8"
```

en bovendien in de 'XML-Prolog':

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

Indien nodig, als er een andere encodering gebruikt is, dan dient die encodering in plaats van UTF-8 vermeld te worden. Het ten onrechte vermelden van UTF-8, terwijl er feitelijk een andere encodering gebruikt is, leidt onherroepelijk tot foute interpretaties bij de tegenpartij. Het gebruik van UTF-8 heeft wel duidelijk de voorkeur.

Afspraak 16: Bij het versturen van een Bericht dient bij voorkeur de UTF-8 encodering gebruikt te worden.

5.9. Berichten met binaire bestanden

Zoals besproken in 4.9 Ondersteuning voor binaire bestanden dienen binaire bestanden in een bericht vervoerd te worden met behulp van elementen van het type “xmime:base64Binary”. Base64 conversie is een oude en zeer bekende techniek die in alle programmeertalen

geïmplementeerd is. Met Base64 kun je iedere binaire file representeren als een string met daarin alleen maar 'veilige' karakters.

Zie bijvoorbeeld het volgende willekeurige met Paint gemaakte plaatje:  . De Base64 string voor dit plaatje is als volgt:


```
/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
AQEBAQEBAQEBAQEBAQEBAQEBAQH/2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEAQEBAQEBAQEBA
QEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAARCAAWACcDASIAAhEBAxEB/8QAGQABAAMBAQAAAAAAAAA
AAAAAAAclCQYK/8QAJRAAAgICAQMFAQEBAAAAAAAABgcFCAQJAwIVFgABChcYExQS/8QAFAEBAAAAAAAAAAAAAAAAAAA
AAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/ALdlhBPXQ+znW3Xrt/gKirXSDaGwjqJmgNaVItpn1Try/Vh
YuANMQyCUhVhcH6LWVPbFyBR0sSKB2SuSNTLPFwTtRcJnM4xvwDZ2+PyXUVlimqz/ANv8qdVPYFN4lty9ncH49KZrjEB1rXYFC
wFPZb0jU2RYXKv3iss8il0nZddQjaRMRizy9ehdny6FzeSbl6ftDSZ5PadMatMhF1Aa9mhDSCUOWYZWHP8A4qRK6e04xVnV0Idw5
KVvqYU2atIwFfNLPMPDXouKzz4OsIYtY/NBgSqjC9HGo5mgGzrWlWequxKvSkm601AqqP8AL+E2Ebobn2i2EIkmyUSzLRPpQPQ
wLnvYSr6/Zov/AFzYxPDTFylOQc5jX5RYBE9Q1OvEf63QZ1jDV9Tm83cp81BsLKTvyHrYlIxd9g1TsMt2W96j0DXuNZ9U627lyLFH6
9VrRVu62m1UnAM42Jgk0375h8Kj/us8h4LQ6ZrOaxjJj542Ps6zauQV66dUyH9r8CxCtP2HsPagJtyx9RDZUgdVXDCwnEs17sSzkH1
WVts+nAsq7M1ArUJUZxa3MtcB14YNdDbjKJBdwSlHT7KDOrRWcvBABJDr01gL6uTWv+9ozKRUlvAsJmAiu+r5jaEPyFg5kalVXkgo
T5shcaqyoqhcQuBmR5JMIasBUxU+wwy4rrUnTMEJqGC2lT26tvgTWfWCGr5XhP7XspfWMjtqD9Yoq9ZEPebP5ka66zQYUsizJOB+
iwSijBRh8U8MBKit0o47DmyWyHBye+dkQob/ACe1+WRutdB44drb27nqxORS0goLjEBCAuii1Sypw85c1tgPFLTOaHUA4XUqfdPwx
SHkecmxk9OiRygciWtCLKSwtG88MmOt6zg2CQT6T9v3FVyPfS/132asBUCoJSsrHNL5EGyfLBE+CLSwFy5c6NJ1lWCWy9+8/NP+
MVGRleJXu0wn5hoR7zSgoTQpJaQvT70G31sdQVBav6XrLDn5HqAcO2u+sByQn6X/ACclxlpmDTUlUiPA+9O/dhIC0fYBAWj/AJ/3fz
Wbl4ojy/8AZ5TJSeP3bmgCQr0064PC0Is7m2v382i9gfGGYR+9A1MkaWl2yRy+7V3wmYYNmII3e9+/sDk8c6IiNyhKbA1YAKyJWiLT
SdVyIUYULYj09BL9vaDv3gNKTV6IW3SBx18bm1+5DrBVfYrXsaOOOHCp/ILaVb7Mj3Nh5144IJeg+DZJsYjIhiw4Cm5HFKsZcsvller
CJQQNMgaMapX6LnEhR5UNGkAGm2pSDdnTLFt700mNOu4T0FQG7Vba2Wl5rJCQ3aETFuogAmtKZEbSQ643l0Y6FRkTke7EV
EMSGPWLhr09B6HtcdYPybfq8668aqAK96qBroNe30pqH+LlZzdxdGzeC/2z6t+6nv39gcnjn8JU/8ALo7ug5jig745h+K9zmHp6eg//9k
```


Afbeelding 23 Voorbeeld van een Base64 string

Een dergelijke Base64 string kan in een XML bericht worden opgenomen omdat er geen teken-combinaties in kunnen voorkomen die de XML structuur in de war kunnen sturen. Een bericht waarmee bovenstaand plaatje naar de VoorbeeldService (conform de koppelvlak specificaties op (http://bkwi.nl/SuwiML/Diensten/VoorbeeldService) gestuurd kan worden ziet er dan als volgt uit:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
   <SOAP-ENV:Header>
      <smlh:SuwiMLHeader
         xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">...</smlh:SuwiMLHeader>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <smls:Upload xmlns:smls="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService">
         <Burgerservicenr>000000000</Burgerservicenr>
         <Foto
xmime:contentType="image/jpeg">/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQ
EBAQEBAQEBAQEBAQEBAQEB
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEB
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAARCAAWACcDASIA...
hR5UNGkAGm2pSDdnTLFt700mNOu4T0FQG7Vba2Wl5rJCQ3aETFuogAmtKZEbSQ643I0Y6FRkTke7
EVEMSGPWLhr09B6HtcdYPybfq8668aqAK96qBroNe30pqH+LlZzdxdGzeC/2z6t+6nv39gcnjn8J
U/8ALo7ug5jig745h+K9zmHp6eg//9k=</Foto>
      </smls:Upload>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Afbeelding 24 Voorbeeld van een Upload bericht met een binair bestand

Merk op dat ook in het bericht (net als in het XML Schema met het attribuut `xmime:expectedContentTypes` met het attribuut `xmime:contentType="image/jpeg"` aangegeven wordt wat het type is van het meegeleverde binaire bestand.

Een aantal gangbare webservices-toolkits, waaronder Axis2 en .Net, heeft de mogelijkheid om voor berichten met binaire bestanden gebruik te maken van de techniek MTOM (Message Transmission Optimization Mechanism). MTOM maakt gebruik van MIME, bekend van attachments van email. Wanneer MTOM gebruikt wordt, dan komt er, in de plaats van het element dat de base64Binary string bevat, een verwijzing-element

```
<xop:Include href="cid:1.A91D6D2E3D7AC4D580@apache.org"
      xmlns:xop="http://www.w3.org/2004/08/xop/include">
```

in het Bericht. Het Bericht is dan een Multipart MIME Bericht geworden. In het eerste MIME part zit de SOAP Envelope, met de XML payload en daarin een of meer `<xop:Include ...>` elementen. Het attribuut `href="..."` van een `<xop:Include ...>` element verwijst dan naar een ander MIME part waar het desbetreffende binaire bestand in zit. Ter illustratie, de MTOM-versie van het voorbeeld van Afbeelding 25 is als volgt:

```
Content-Type: Multipart/Related; type="application/xop+xml"; boundary="----
=_Part_0_1744155.1118953559416"
Content-Length: 3453
SOAPAction: ""

------=_Part_1_4558657.1118953559446
Content-Type: application/xop+xml; type="text/xml"; charset=utf-8

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xmime="http://www.w3.org/2005/05/xmlmime" xmlns:xop="http://www.w3.org/2004/08/xop/include">
   <SOAP-ENV:Header>
      <smlh:SuwiMLHeader xmlns:smlh="http://bkwi.nl/SuwiML/Header/v0400">...</smlh:SuwiMLHeader>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <smls:Upload xmlns:smls="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService">
         <Burgerservicenr>000000000</Burgerservicenr>
         <Foto>
            <xop:Include href="cid:5aeaa450-17f0-4484-b845-a8480c363444@example.org" />
         </Foto>
      </smls:Upload>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

------=_Part_1_4558657.1118953559446
Content-Type: image/jpeg
Content-ID: <5aeaa450-17f0-4484-b845-a8480c363444@example.org>

   ... binary data ...

   ------=_Part_1_4558657.1118953559446
```

Afbeelding 25 Bericht met een binair bestand in MTOM formaat

Het gebruik van MTOM heeft voordelen zoals een betere scheiding tussen XML payload en bijgeleverde bestanden, minder belasting voor de XML parser, en beter zicht op de grootte van de afzonderlijke bijgeleverde bestanden.

Maar, het eventuele gebruik van MTOM is een ***keuze*** in de implementaties. In de koppelvlak specificaties van een webservice wordt er geen uitsluitsel over gegeven. Axis2 kent een parameter 'enableMTOM' waarmee in de configuratie van Axis2 het gebruik van MTOM aangezet kan worden. In .Net is er daarvoor een element <mtom> in de configuratie-file web.config. Het eventuele gebruik van MTOM voor een bepaalde uitwisseling moet tussen de betrokken partijen nader afgesproken worden. Beide partijen moeten MTOM-aware software hebben, en beide partijen moeten de MTOM optie aanzetten, anders werkt het niet.

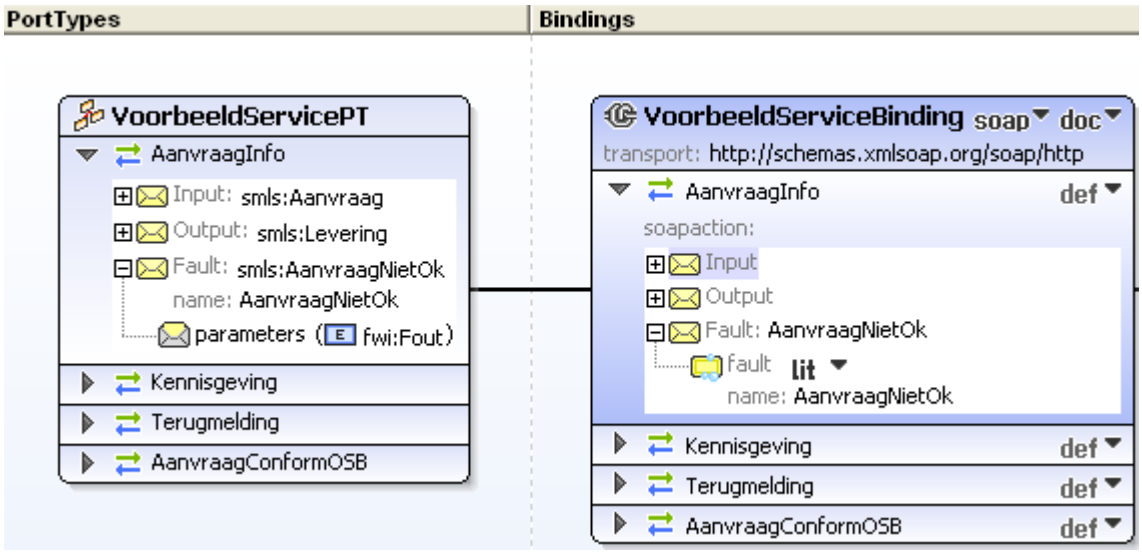
6. Foutafhandeling

Een webservice moet conform de koppelvlak specificaties reageren op inkomende requests. In de WSDL is ook ruimte voor het specificeren van voor de hand liggende foutsituaties. Fouten kunnen optreden op verschillende niveaus, grofweg overeenkomend met de lagen uit Afbeelding 2. Enkele veel voorkomende fout-situaties worden onder besproken. Er wordt getoond hoe een Axis2 implementatie in die gevallen zal reageren. De precieze reactie bij andere implementaties zal net iets anders zijn, maar er wel op lijken.

Het is raadzaam dat de besproken situaties ook in het test-traject van een webservice applicatie (client of server) aan bod komen. Er dient gecontroleerd te worden of er in deze situaties door de applicatie het gewenste gedrag vertoond wordt. En door het te testen zullen dergelijke situaties sneller herkend worden wanneer ze onverhoopt in de Productie-omgeving optreden.

6.1. Foutafhandeling in de WSDL

Wanneer er ten tijde van het opstellen van de specificaties voor een webservice bepaalde uitzondering-situaties onderkend worden die een aparte afhandeling vergen, dan kan dat in de WSDL vermeld worden.



Afbeelding 26 Specificatie van de Fout 'AanvraagNietOk' in de WSDL

In het Voorbeeld van Afbeelding 27, zie ook http://bkwi.nl/SuwiML/Diensten/VoorbeeldService, is een generieke Fault 'AanvraagNietOk' opgenomen, maar indien gewenst kan daar ook een specifiekere fout-situatie naast vermeld worden. De partijen die de webservice implementeren dienen ook de in de WSDL vermelde Fout-situaties te implementeren, zowel in de Client-applicatie (hoe reageert de applicatie als er zo'n Foutmelding terug komt) als in de Server applicatie (in de bedoelde uitzondering-situatie dient de gespecificeerde Foutmelding terug gestuurd te worden).

```
<wsdl:definitions ... xmlns:fwi="http://bkwi.nl/SuwiML/FWI/v0201" ...>
   <wsdl:types>
      <xs:schema targetNamespace="http://bkwi.nl/SuwiML/FWI/v0201">
         <xs:include schemaLocation="../../FWI/v0201/FWI.xsd"/>
      </xs:schema>
      ...
   </wsdl:types>
   ...
   <wsdl:message name="AanvraagNietOk">
      <wsdl:part name="parameters" element="fwi:Fout"/>
   </wsdl:message>
   <wsdl:portType name="VoorbeeldServicePT">
      <wsdl:operation name="AanvraagInfo">
         ...
         <wsdl:fault name="AanvraagNietOk" message="smls:AanvraagNietOk"
          wsaw:Action="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService/Fout"/>
      </wsdl:operation>
      ...
   </wsdl:portType>
   <wsdl:binding name="VoorbeeldServiceBinding" type="smls:VoorbeeldServicePT">
      ...
      <wsdl:operation name="AanvraagInfo">
         ...
         <wsdl:fault name="AanvraagNietOk">
            <soap:fault name="AanvraagNietOk" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      ...
   </wsdl:binding>
</wsdl:definitions>
```

Afbeelding 27 Specificatie van de Foutmelding 'AanvraagNietOk' in de WSDL

In de <wsdl:message>'s voor de Foutmeldingen wordt verwezen naar het Element <fwi:Fout>. Dat Element wordt gespecificeerd in het FWI.xsd schema, zie http://bkwi.nl/SuwiML/FWI. Met het FWI.xsd schema maken we hergebruik van een vaste structuur voor SuwiML Foutmeldingen. FWI staat voor 'Fouten, Waarschuwingen en andere Informatie'. Het element <fwi:Fout> bevat sub-elementen 'Code', 'Tekst', 'Detail' en 'Bron'.

Er wordt door de Server-applicatie een AanvraagNietOk Foutmelding gegenereerd om meer gedetailleerde informatie omtrent de precieze fout-situatie in een SOAP Fault te kunnen terugkoppelen. Zie ook 6.4 Fouten in de stuurinformatie.

6.1.1 Afhandeling van ‘Burgerservicenr niet gevonden’

Een regelmatig terugkerend discussiepunt is wat de Server applicatie zou moeten doen wanneer het opgevraagde Burgerservicenummer niet voorkomt in de database. Tot nu toe zijn de specificaties zo ingericht dat deze situatie niet als een Fout wordt behandeld, het is tenslotte heel normaal dat een Burgerservicenummer niet in een database van een van de Ketenpartijen voorkomt. In de specificaties van het Response-bericht is daartoe het element <smls:ClientSuwi> optioneel gesteld. Mocht het Burgerservicenummer niet in de database voorkomen, dan kan de Server applicatie volstaan met een leeg <smls:AanvraagInfoResponse> element, dus met weglating van het optionele element <smls:ClientSuwi>:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
      ...
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <AanvraagInfoResponse xmlns="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService"/>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Afbeelding 28 Response in het geval dat een Burgerservicenummer niet voorkomt

6.2. Fouten in de HTTP Headers

Als alles goed gaat dan zien de HTTP Headers van het Response er als volgt uit:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 20 Oct 2008 08:26:11 GMT

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   ...
</soapenv:Envelope>
```

Afbeelding 29 Een succesvolle HTTP Response

De HTTP Code 200 duidt op een succesvolle HTTP dialoog. Als er een andere HTTP Code terug gestuurd wordt dan is er iets speciaal aan de hand.
Als de hostnaam van het verzoek fout is, dan zal het verzoek niet bij de bedoelde webserver uitkomen.

```
POST /axis2/services/VoorbeeldService HTTP/1.1
Content-Type: text/xml
User-Agent: XML Spy
SOAPAction: ""
Host: foutehost.nl
```

Afbeelding 30 Verkeerde hostnaam in het HTTP Request

Het verzoek kan dan niet getransporteerd worden en in de software van de client zal een “UnknownHostException: foutehost.nl” optreden.

Als de hostnaam wel goed is, maar de server staat uit of heeft het te druk, dan zal er in de software van de client een “ConnectException: Connection timed out” optreden. Een dergelijke time- out verschijnt ook wanneer de hostnaam naar een verkeerd IP-Adres vertaald wordt.

Als de hostnaam van het verzoek goed is maar het pad van de webservice is fout (in de eerste regel van de HTTP Header van het Request staat dan bijvoorbeeld “POST /foutpad/VoorbeeldService HTTP/1.1” dan zal de webserver een HTTP Code 404 terugsturen: “HTTP/1.1 404 Not Found”, eventueel samen met een HTML pagina met een toelichting.

6.3. Fouten in de SOAP structuur

Wanneer er in het Request <soapenv:Envelop> in plaats van <soapenv:Envelope> staat, dan kan de webservice het verzoek niet interpreteren:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 20 Oct 2008 10:18:38 GMT Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Action>http://www.w3.org/2005/08/addressing/soap/fault</wsa:Action>
   </soapenv:Header>
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode/>
         <faultstring>First Element must contain the local name, Envelope , but found
            Envelop</faultstring>
         <detail/>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

Afbeelding 31 SOAP Fault naar aanleiding van 'Envelop' in het Request

Een dergelijke foutmelding verschijnt ook wanneer er XML in het Request opgestuurd wordt die helemaal niet lijkt op een SOAP Bericht, bijvoorbeeld <EenFoutSOAPBericht/>.
Wanneer er in het XML gedeelte van het Request helemaal geen XML staat, maar bijvoorbeeld “<geenXML” (dus zonder sluithaak '>'), dan zegt de server zoiets als:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
Date: Mon, 20 Oct 2008 10:15:34 GMT
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Action>http://www.w3.org/2005/08/addressing/soap/fault</wsa:Action>
   </soapenv:Header>
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode/>
         <faultstring>com.ctc.wstx.exc.WstxEOFException: Unexpected end of input block;
            expected an identifier&#13; at [row,col {unknown-source}]: [1,8]</faultstring>
         <detail/>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

Afbeelding 32 SOAP Fault naar aanleiding van onbegrijpelijke XML in het Request

Merk op dat in beide gevallen de Foutmelding vergezeld gaat van een HTTP Code 500 “Internal Server Error”. En merk ook op dat in beide gevallen de Foutmelding verpakt zit in een <soapenv:Fault> structuur. Dit is het standaard voorgeschreven gedrag van een SOAP 1.1 webservice. Bovendien heeft door het gebruik van WS-Addressing het element <wsa:Action> een speciale waar http://www.w3.org/2005/08/addressing/soap/fault”.

Ook het opsturen van bijvoorbeeld <soap:Headers> in het Request in plaats van <soap:Header> leidt tot een dergelijke SOAP Fault:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Action>http://www.w3.org/2005/08/addressing/soap/fault</wsa:Action>
   </soap:Header>
   <soap:Body>
      <soap:Fault>
         <faultcode>Sender</faultcode>
         <faultstring>SOAP Envelope can not have children other than SOAP Header and Body</faultstring>


            <detail/>

      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Afbeelding 33 SOAP Fault naar aanleiding van <soap:Headers

6.4. Fouten in de stuur-informatie

In de WSDL beschrijving van een webservice staan instructies voor de vulling van het WS-Addressing Header element <wsa:Action>. Een voor de server onbekende <wsa:Action> in het request leidt tot een <soap:Fault>:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Action>http://www.w3.org/2005/08/addressing/fault</wsa:Action>
    <wsa:RelatesTo>urn:uuid:F430D2CE8910294F3D1219047809955</wsa:RelatesTo>
    <wsa:FaultDetail>
      <wsa:ProblemHeaderQName>wsa:Action</wsa:ProblemHeaderQName>
    </wsa:FaultDetail>
  </soap:Header>
  <soap:Body>
    <soap:Fault xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <faultcode>wsa:ActionMismatch</faultcode>
      <faultstring>A header representing a Message Addressing Property is not valid and
        the message cannot be processed</faultstring>
      <detail/>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Afbeelding 34 SOAP Fault naar aanleiding van onjuiste <wsa:Action>

Het element <soapenv:Fault> / <detail> biedt meer ruimte voor nadere specificatie van de opgetreden Fout-situatie. Een fout in de SuwiML Header van het Request kan bijvoorbeeld vermeld worden. Het element <fwi:Fout> dat in de WSDL vermeld wordt (zie 6.1 Foutafhandeling in de WSDL) dient daarvoor gebruikt te worden. Het gebruik van <fwi:Fout> in alle SuwiML webservices maakt het mogelijk dat Client applicaties daar een generieke afhandeling voor kunnen implementeren.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Action>http://bkwi.nl/SuwiML/Diensten/VoorbeeldService/Fout</wsa:Action>
    <wsa:RelatesTo>34343</wsa:RelatesTo>
  </soapenv:Header>
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>Fout in de SuwiML Header</faultstring>
      <detail>
        <fwi:Fout xmlns:fwi="http://bkwi.nl/SuwiML/FWI/v0201">
          <Code>ElementOnbekend</Code>
          <Tekst>Onbekend Element aangetroffen in de SuwiML Header: ...</Tekst>
          <Bron>
            <DN>cn=KBS,o=UWV,c=nl</DN>
          </Bron>
        </fwi:Fout>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Afbeelding 35 SOAP Fault met fwi:Fout naar aanleiding van onjuiste SuwiML Header

6.5. Fouten in de SOAP Body

De namespace en de naam van het eerste element van de SOAP Body moeten overeenkomen met die van het element dat verwacht wordt door de operation van de webservice, zoals vastgelegd in de WSDL van de webservice. Als het request een onverwachte namespace bevat,

dan klopt de context van het verzoek niet. De server kan het verzoek niet accepteren en genereert een SOAP Fault:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Action>http://www.w3.org/2005/08/addressing/soap/fault</wsa:Action>
      <wsa:RelatesTo>urn:uuid:42C1B28D128C21FC3A1219051489480</wsa:RelatesTo>
   </soap:Header>
   <soap:Body>
      <soap:Fault>
         <faultcode>soap:Server</faultcode>
         <faultstring>namespace mismatch require http://services.samples/xsd found
http://services.samples/xsd1</faultstring>
         <detail/>
      </soap:Fault>
   </soap:Body>
</soap:Envelope>
```

Afbeelding 36 SOAP Fault naar aanleiding van onbekende namespace

6.6. Fouten in de Payload

Vaak zal de Payload van een Bericht (de echte inhoudelijke functionele informatie) door een ander systeem afgehandeld worden. Het systeem dat de webservice implementeert (laten we zeggen de 'SOAP Handler') krijgt wel de SOAP Requests binnen maar geeft de inhoud van dat Request dan door aan het Functionele systeem. Het Functionele systeem kan een probleem hebben met de inhoud van het Request, of met de afhandeling van het Request. Voorbeelden van functionele fouten zijn ‘Burgerservicenr voldoet niet aan 11-proef’, ‘Ongeldigepostcode’, etc. Het Functionele systeem zal daarover een terugkoppeling doen richting de SOAP Handler. De communicatie tussen de Client applicatie en de SOAP Handler is in zo'n geval voor wat betreft het SOAP niveau gewoon succesvol verlopen. Dus daarom is voor dit geval een SOAP Fault in het Response niet op z'n plaats. De SuwiML webservices specificaties worden echter zo opgesteld dat de SOAP Handler toch de terugkoppeling van het Functionele systeem in het Response door kan sturen naar de Client applicatie. Het generieke element <fwi:FWI> dient daar voor gebruikt te worden. Het element <fwi:FWI> wordt ook gespecificeerd in het FWI.xsd schema. Het element <fwi:FWI> bevat subelementen 'Fout', 'Waarschuwing' en/of 'Informatie', ieder op zijn beurt voorzien van subelementen 'Code', 'Tekst', 'Detail' en 'Bron'. Zie http://bkwi.nl/SuwiML/FWI/ voor de precieze specificaties van <fwi:FWI>.
Iedere Client applicatie dient te beslissen òf, en hóe, FWI informatie getoond wordt op het scherm. In het geval van Suwinet Inkijk verschijnt er een rode balk voor iedere <fwi:FWI>/<Fout> en een gele balk voor iedere <fwi:FWI>/<Waarschuwing>.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
      …
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <smls:AanvraagInfoResponse xmlns:smls="http://bkwi.nl/SuwiML/Diensten/VoorbeeldService">
         <ClientSuwi>...Hier komt het (gedeeltelijke) antwoord dat door het Functionele systeem gemaakt kon
worden...</ClientSuwi>
         <fwi:FWI xmlns:fwi="http://bkwi.nl/SuwiML/FWI/v0201">
            <!-- Maar hier komen enkele kanttekeningen bij het antwoord: -->
            <Fout>
               <Code>AntwoordOnvolledig</Code>
               <Tekst>Niet alle databases waren bereikbaar</Tekst>
               <Bron>...</Bron>
            </Fout>
            <Waarschuwing>
               <Code>OudeVersie</Code>
               <Tekst>Dit was een verzoek voor een oude versie van deze webservice.
                  Graag upgraden naar versie ...</Tekst>
               <Bron>...</Bron>
            </Waarschuwing>
            <Informatie>
               <Code>Onderhoud</Code>
               <Tekst>Donderdag vind er groot onderhoud plaats vanaf ... uur</Tekst>
               <Bron>...</Bron>
            </Informatie>
         </fwi:FWI>
      </smls:AanvraagInfoResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Afbeelding 37 Succesvolle SOAP Response met FWI kanttekeningen bij het antwoord

7. Logging

De beheerders van een systeem hebben taken op het gebied van monitoring, voorkomen en herstellen van incidenten, en het signaleren van problemen. Niet alleen voor hun eigen Partij, maar ook voor andere Partijen waarmee zij elektronisch informatie uitwisselen. Het vastleggen van informatie rondom de uitwisselingen maakt het mogelijk dat zij aan Diagnose en Foutherstel kunnen werken. Bovendien dienen er Rapportages gemaakt te kunnen worden op basis van de gelogde informatie. De gelogde informatie dient ook een bepaalde tijd bewaard te worden zodat het verloop van de informatie-uitwisseling achterhaald kan worden wanneer er later een vermoeden bestaat van misbruik.

Onderscheid moet worden gemaakt tussen het loggen tijdens de gegevensuitwisseling en het loggen tijdens de gegevensverwerking. Dit Hoofdstuk heeft betrekking op het loggen tijdens de gegevensuitwisseling. Het loggen tijdens de gegevensverwerking is een zaak van de afzonderlijke partijen zelf.

7.1. Logging ten behoeve van Diagnostiek en Foutherstel

Voor eventuele Diagnostiek en Foutherstel is het van belang dat achteraf in de logging van een systeem door Beheerders nagekeken kan worden hoe het Berichtenverkeer verlopen is. Het loggen kan echter op meer of minder uitgebreide manier gebeuren. Afzonderlijke onderdelen die al dan niet in combinatie met elkaar gelogd zouden kunnen worden zijn de volgende:

- De HTTP Parameters
- De URL waar een Bericht naar toe gestuurd werd
- Het IP-Adres waar een Bericht vandaan kwam
- De DistinguishedName van het Client Certificaat waarmee een Request gedaan is
- De gehele (inhoud van de) SOAP Header
- Afzonderlijke velden in de SOAP Header
- De inhoud van de SOAP Body
- Afzonderlijke velden in de SOAP Body
- Attachments (indien niet vervat in de SOAP Body)
- Datum en tijdstip van ontvangst van een Bericht
- Datum en tijdstip van verzenden van een Bericht
- Het bereiken van een time-out bij het wachten op een Response

Al te enthousiaste grootschalige logging vormt een bedreiging voor de schaalbaarheid van een systeem. Het is daarom ook wenselijk dat de koppelvlak specificaties zo zijn opgesteld dat er niet al te zware Berichten verstuurd worden, en dat eventuele binaire bestanden met behulp van MTOM in een apart MIME attachment meegestuurd worden, zodat de binaire bestanden zelf niet gelogd hoeven te worden.

Maar, in ieder geval is het van belang dat de Logging van alle partijen samen zo is ingericht dat, indien nodig, in gezamenlijk overleg tussen de Beheerders van de verschillende bij een bepaald Bericht betrokken Partijen, met de Logs kan worden achterhaald wat het traject, end-to-end, van het Bericht geweest is en wat er onderweg allemaal mee gebeurd is.

Omdat veel SuwiML Berichten vertrouwelijke persoonsgegevens van Burgers bevatten is voor de “inhoud van de SOAP Body” de Wet Bescherming Persoonsgegevens (WBP) van toepassing. De

WBP stelt (zie bijvoorbeeld http://www.cbpweb.nl/documenten/inf_va_bewaartermijnen.shtml) onder andere dat:

- persoonsgegevens gedeeld mogen worden als dat vereist is op grond van een wettelijk voorschrift, in het kader van een publiekrechtelijke taak of als er daarvoor een gerechtvaardigd belang is
- persoonsgegevens alleen verwerkt mogen worden voor welbepaalde, uitdrukkelijk omschreven en gerechtvaardigde doeleinden
- persoonsgegevens niet langer bewaard mogen worden dan noodzakelijk is voor de doeleinden waarvoor zij zijn verzameld of worden gebruikt

Voor de logging van de inhoud van de SOAP Body maken we onderscheid tussen

- Sleutelwaardes (zoals een BSN) in een Bericht
- Vertrouwelijke informatie (zoals Persoonsgegevens van Klanten / Burgers)
- Niet-vertrouwelijke informatie (zoals de inhoud van een Ontvangstbevestiging)

In verband met de WBP dienen systemen die 'slechts' tot doel hebben om informatie dóór te sturen (zoals een Broker of een SOAP Adapter), of om overzichten te tonen (zoals Suwinet Inkijk), voor wat betreft het loggen van Vertrouwelijke informatie in de Berichten een veel kortere bewaartermijn te hanteren dan voor Stuurinformatie, Sleutelwaardes en andere Niet-vertrouwelijke informatie.

Bovenstaande overwegingen leiden tot de volgende Afspraken:

Afspraak 17: Alle partijen loggen de inhoud van een Bericht, de inhoud van de stuurinformatie, het tijdstip, en het adres waar een Bericht naar toe gaat of vandaan komt

Afspraak 18: Persoonsgegevens worden voor een termijn van maximaal ... in de logs bewaard. (Deze termijn wordt nog Ketenbreed, op basis van het advies van de DPB, vastgesteld)

Afspraak 19: Stuurinformatie, Sleutelwaardes en andere Niet-vertrouwelijke informatie wordt tenminste 18 maanden bewaard

De bewaartermijnen zijn Ketenbreed vastgesteld. De bewaartermijn van 18 Maanden is overgenomen uit het Rapport Diagnostiek: “In verband met de wettelijke eis tot het kunnen leveren van bewijslast, is in overleg tussen BKWI, de juridische dienst CWI en het CBP, de bewaartermijn van de logbestanden van Suwinet vastgesteld op 18 maanden. Dit betreft het end-to-end kunnen volgen van het bericht.”

De bewaartermijn van 18 Maanden sluit ook aan op het heersende beleid bij de GBA. In het Logisch Ontwerp GBA v3.5 (blz 93) staat:

> ### 4.2.5 Bewaartermijn protocolgegevens
>
> Om invulling te kunnen geven aan het recht van de burger om opgave te verlangen van de gegevens die in het afgelopen jaar over hem zijn verstrekt (artikel 103 Wet GBA), dienen de protocolgegevens over verstrekkingen minimaal één jaar na het moment van verstrekken, te worden bewaard. Voor de bewaartermijn van de protocolgegevens van de andere procedures wordt geadviseerd een zelfde termijn aan te houden.

Meer in detail betekent bovenstaande dat door een Verzender van een Request en door een Ontvanger van een Response de volgende onderdelen gelogd dienen te worden:

Tabel 11 Te loggen informatie en termijnen

Te loggen onderdeel		Door de Verzender van een	Door de Ontvanger van een
De URL waar een Request naar toe gestuurd werd		> 18 Maanden	
Het IP-Adres waar een Request vandaan kwam			> 18 Maanden
De DN van het Client Certificaat			> 18 Maanden
De inhoud van de SOAP Header van het Request		> 18 Maanden	> 18 Maanden
De inhoud van de SOAP Body van het Request	Sleutelwaardes, niet-vertrouwelijke informatie	> 18 Maanden	> 18 Maanden
	Vertrouwelijke informatie	< XXX	< XXX
Datum en tijdstip van verzenden van een Request		> 18 Maanden	
Datum en tijdstip van ontvangst van een Request			> 18 Maanden
Datum en tijdstip van verzenden van een Response			> 18 Maanden
De HTTP Code van het Response		> 18 Maanden	> 18 Maanden
De inhoud van de SOAP Header van het Response		> 18 Maanden	> 18 Maanden
De inhoud van de SOAP Body van het Response	Sleutelwaardes, niet-vertrouwelijke informatie	> 18 Maanden	> 18 Maanden
	Vertrouwelijke informatie	< XXX	< XXX
Datum en tijdstip van ontvangst van een Response		> 18 Maanden	
Een time-out bij het wachten op een Response		> 18 Maanden	

Lange bewaartermijnen stellen natuurlijk eisen aan de opslagcapaciteit van de betrokken systemen. En het heeft ook zijn weerslag op de schaalbaarheid. Wanneer een systeem per jaar 500000 SuwiML Berichten zou verwerken, van gemiddeld 10 kB per Bericht, dan zou er over een termijn van 18 maanden ***7,5 GB*** aan opslagruimte nodig zijn voor de logging.

7.2. Management Informatie

Naast het bieden van mogelijkheden voor Diagnostiek en Foutherstel dient iedere Partij ook op verzoek Management Informatie te kunnen leveren. In de diverse Service Level Agreements worden daarover afspraken gemaakt. Het betreft het kunnen leveren van aantallen Berichten, Fouten die optraden, Time-Outs, Response-tijden en Down-tijden. Bovendien zal op gezette tijden uitgerekend moeten worden in hoeverre door het systeem aan de afspraken in de SLA's voldaan is.

8. Versiebeheer

De koppelvlak specificaties van een SuwiML webservice worden opgeleverd in een .zip file. De .zip file bevat de WSDL beschrijving van de webservice, en alle bouwstenen (in XML Schema .xsd files) die er bij horen. Het versienummer (bijvoorbeeld v0100) van een webservice wordt vastgelegd in de targetnamespace van de WSDL file. Het versienummer staat ook in de naam van de .zip file. In de naam van de .zip file staat daarnaast ook nog het buildnummer (bijvoorbeeld b01). Tijdens het ontwikkel- en implementatietraject van een webservice zullen er meestal meerdere .zip files opgeleverd worden voor een bepaalde versie van een webservice. Iedere .zip file zal dan een hoger buildnummer hebben. Uiteindelijk dienen de implementaties te voldoen aan de specificaties in de .zip file met het hoogste buildnummer.

Wanneer er nieuwe functionele wensen komen zal dat leiden tot een nieuwe versie van de koppelvlak specificaties. Het versienummer in de WSDL beschrijving wordt dan opgehoogd. Behalve dat de nieuwe functionele wensen in de specificaties verwerkt zullen worden, zal op zo'n moment ook van alle benodigde bouwstenen de meest actuele versie aan de .zip file toegevoegd worden. Het betreft in zo'n geval bijvoorbeeld bouwstenen uit een nieuwere versie van het SuwiML Basisschema, of een nieuwere versie van de SuwiML Header, of van FWI.xsd ten behoeve van de Foutafhandeling.

Afspraak 20: Een nieuwe versie van de koppelvlak specificaties van een webservice bevat de meest actuele versie van de benodigde bouwstenen

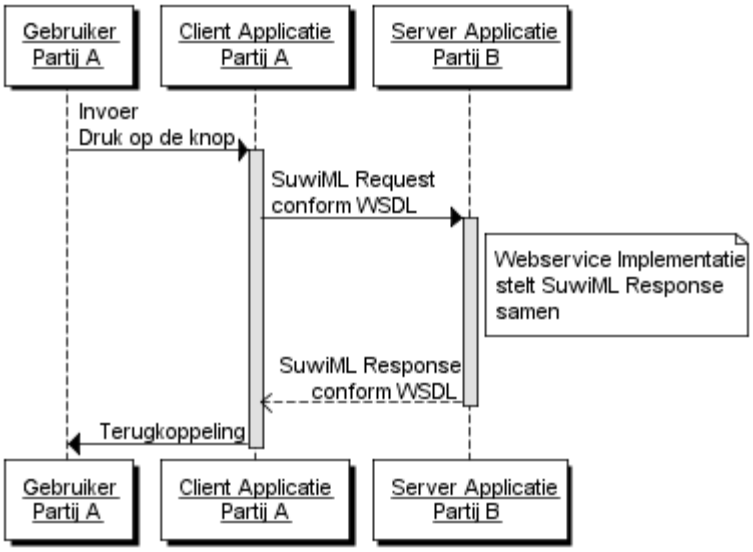
Het invoeren van een nieuwe versie van een webservice betekent niet per se dat de oude versie per diezelfde datum uitgefaseerd moet worden. Er kan door een implementerende partij voor gekozen worden om (liefst tijdelijk) meerdere versie van een webservice te ondersteunen. De oude versie kan definitief uitgefaseerd worden zodra de laatste gebruiker van de webservice over is naar de nieuwe versie.

Afspraak 21: Een partij kan tegelijkertijd meerdere versies van een webservice ondersteunen, maar niet meerdere builds van eenzelfde versie

9. Scenario’s en Sequence diagrammen

In dit hoofdstuk wordt een overzicht geboden van alle Bericht-uitwisseling scenario's die in de Suwi keten voorkomen. De scenario's worden in Sequence Diagrammen weergegeven. In en bij de Sequence Diagrammen worden instructies en aandachtspunten vermeldt die bij het implementeren van de koppelvlak specificaties ter harte genomen dienen te worden. Alle besproken scenario's worden door deze versie van de Transactiestandaard ondersteund.

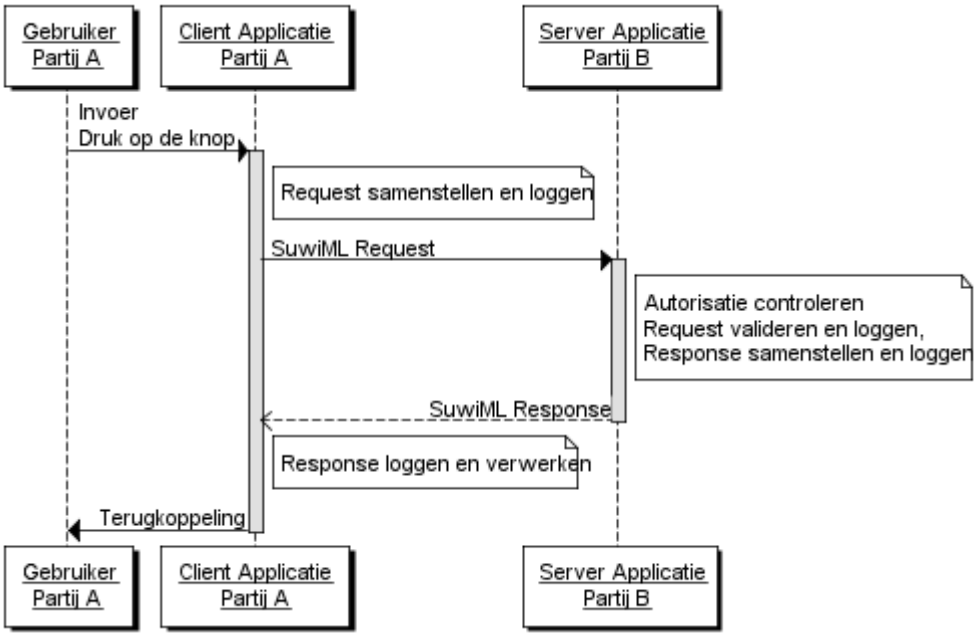
9.1. Raadplegingen



Afbeelding 38 Meest eenvoudige SuwiML Request - Response scenario

In de basis bieden SuwiML webservices een eenvoudige methode voor communicatie tussen een Client Applicatie van de ene partij (Partij A) en een Server Applicatie van een andere partij (Partij B), zie Afbeelding 38. Deze Transactiestandaard gaat met name over het koppelvlak tussen de Client Applicatie van Partij A en de Server Applicatie van Partij B.

Bijna alle SuwiML webservices leveren privacy-gevoelige informatie. Voor alle Services waarbij vertrouwelijke informatie getransporteerd wordt zijn er extra maatregelen noodzakelijk op het gebied van Autorisatie en Logging, zie Afbeelding 39. De Client Applicatie en de Server Applicatie dienen beiden daarvoor te zorgen.

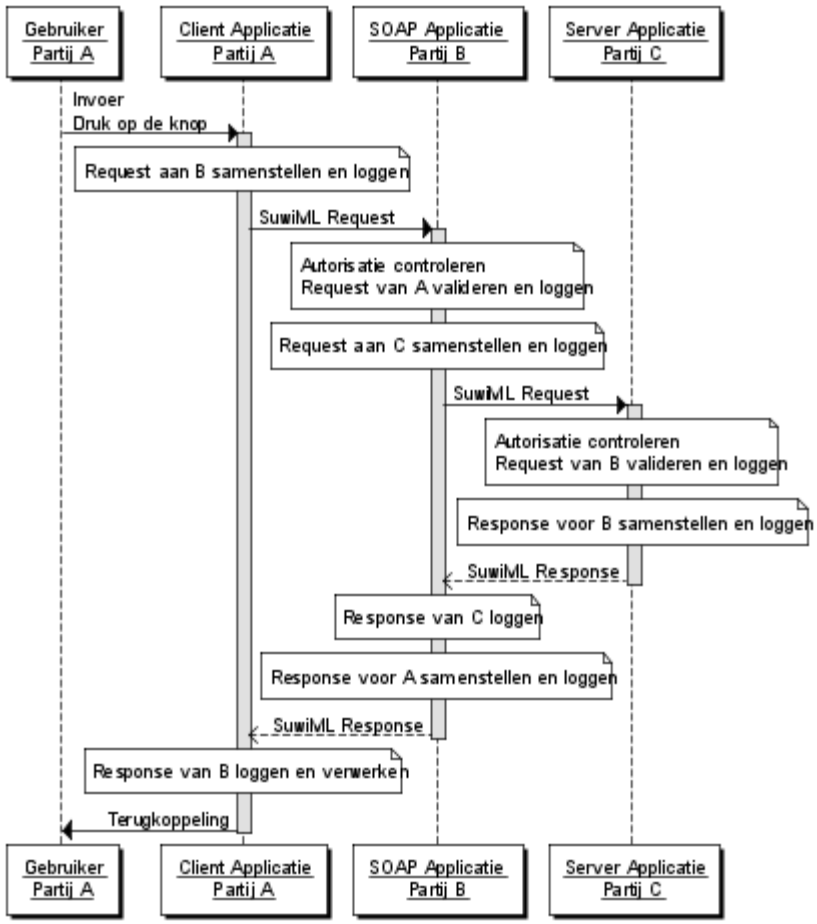


Afbeelding 39 Zorg voor Logging, Validatie, Autorisatie

Het kan zijn dat Partij B voor het samenstellen van het Response informatie nodig heeft van nog andere Partijen. Op basis van het Request van Partij A wordt door Partij B een Request naar Partij C gestuurd, zie Afbeelding 40.

In het scenario van Afbeelding 40 is het vaak zo dat het koppelvlak tussen Partij A en Partij B niet precies hetzelfde is als het koppelvlak tussen Partij B en Partij C. Bekende voorbeelden uit de Suwi praktijk zijn de Suwi Broker en de Broker (Sectorloket) van het Inlichtingenbureau. Beiden spelen de rol van Partij B in Afbeelding 40. De Requests die door Inlezende applicaties naar de Suwi Broker gestuurd worden zijn vragen van de vorm “Geef mij alle info voor dit Burgerservicenummer van alle achterliggende Partijen”. De Requests die door de Suwi Broker naar het Sectorloket van het Inlichtingenbureau gestuurd worden zijn vragen van de vorm “Geef mij alle info voor dit Burgerservicenummer van alle Gemeenten waar deze Persoon bekend is”. En de Requests die door het Sectorloket naar een bepaalde Gemeente gestuurd worden zijn van de vorm “Geef mij alle info van uw Gemeente over de persoon met dit Burgerservicenummer”. Het Response van de Gemeente bevat alleen de informatie van die Gemeente over de persoon in kwestie. Het Response van het Sectorloket bevat alle info van alle Gemeenten waar de persoon bekend. En het Response van de Suwi broker bevat alle info van de verschillende Gemeenten maar ook de info van de andere achterliggende Partijen. De koppelvlakken tussen de verschillende partijen zijn dus verschillend, ook al ziet de vraag er steeds bijna hetzelfde uit.

Wanneer er sprake is van verschillende koppelvlakken, dan dient dit ook in de koppelvlak specificaties tot uiting gebracht te worden. Er dienen aparte WSDL beschrijvingen gemaakt te worden voor de verschillende koppelvlakken.

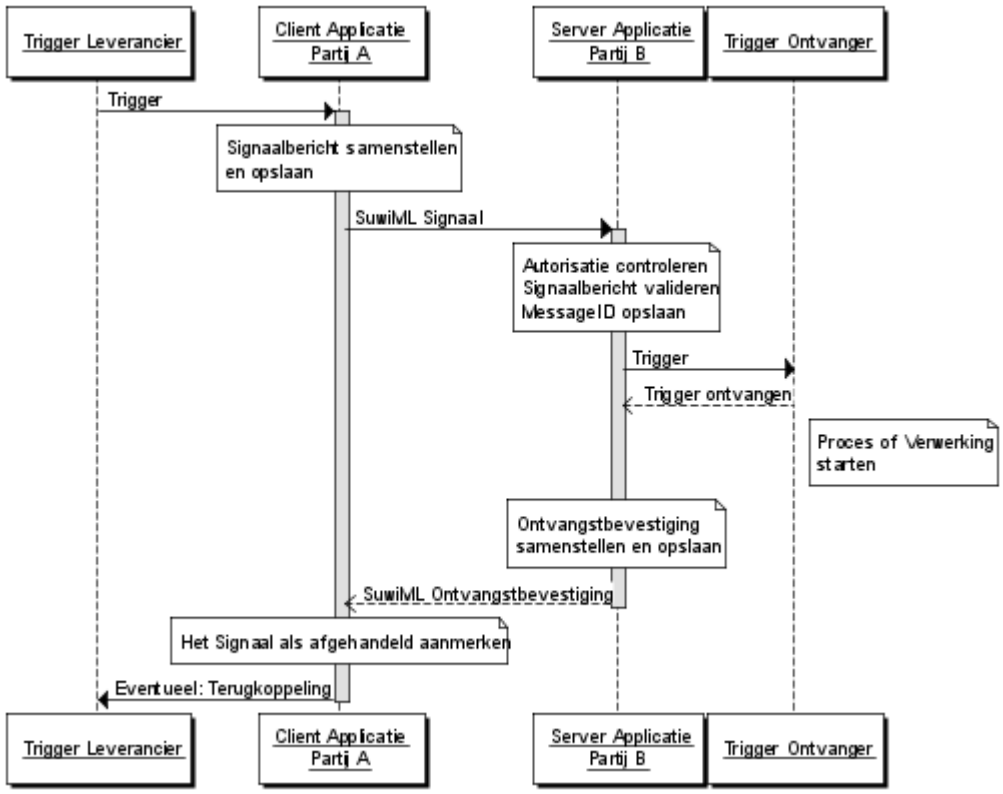


Afbeelding 40 Een SuwiML Request van Partij A dat door partij B doorgestuurd wordt naar Partij C

9.2. Triggers, Signalen en (Terug-)Meldingen

Wanneer het Request-Bericht van Partij A een Melding, Signaal of een Trigger bevat dan zal er bij Partij B een Proces of een Verwerking opgestart worden. Het Response Bericht krijgt dan het karakter van een Ontvangstbevestiging, zie Afbeelding 41. Voor de overzichtelijkheid zijn in Afbeelding 41 de opmerkingen over het Loggen weggelaten, maar het loggen dient net zo te gebeuren als in Afbeelding 39.

Bij dit type Berichten is het van belang dat het Signaalbericht vóór het versturen door de Client Applicatie opgeslagen wordt. In het geval van een systeemcrash of wanneer door andere oorzaken de Ontvangstbevestiging niet ontvangen wordt, dan kan de Client Applicatie het Signaalbericht daardoor nogmaals opsturen. Het kan echter zo zijn dat de Server applicatie het Signaalbericht wel degelijk had ontvangen. Doordat de Server Applicatie tenminste het MessageID van het Signaalbericht had opgeslagen zal de Server het opnieuw verzonden Signaalbericht herkennen en niet nogmaals de Verwerking of het Proces op starten. Wèl dient de Server Applicatie dan de Ontvangstbevestiging nogmaals terug te sturen, kennelijk was die de eerste keer niet aangekomen bij de Client Applicatie.



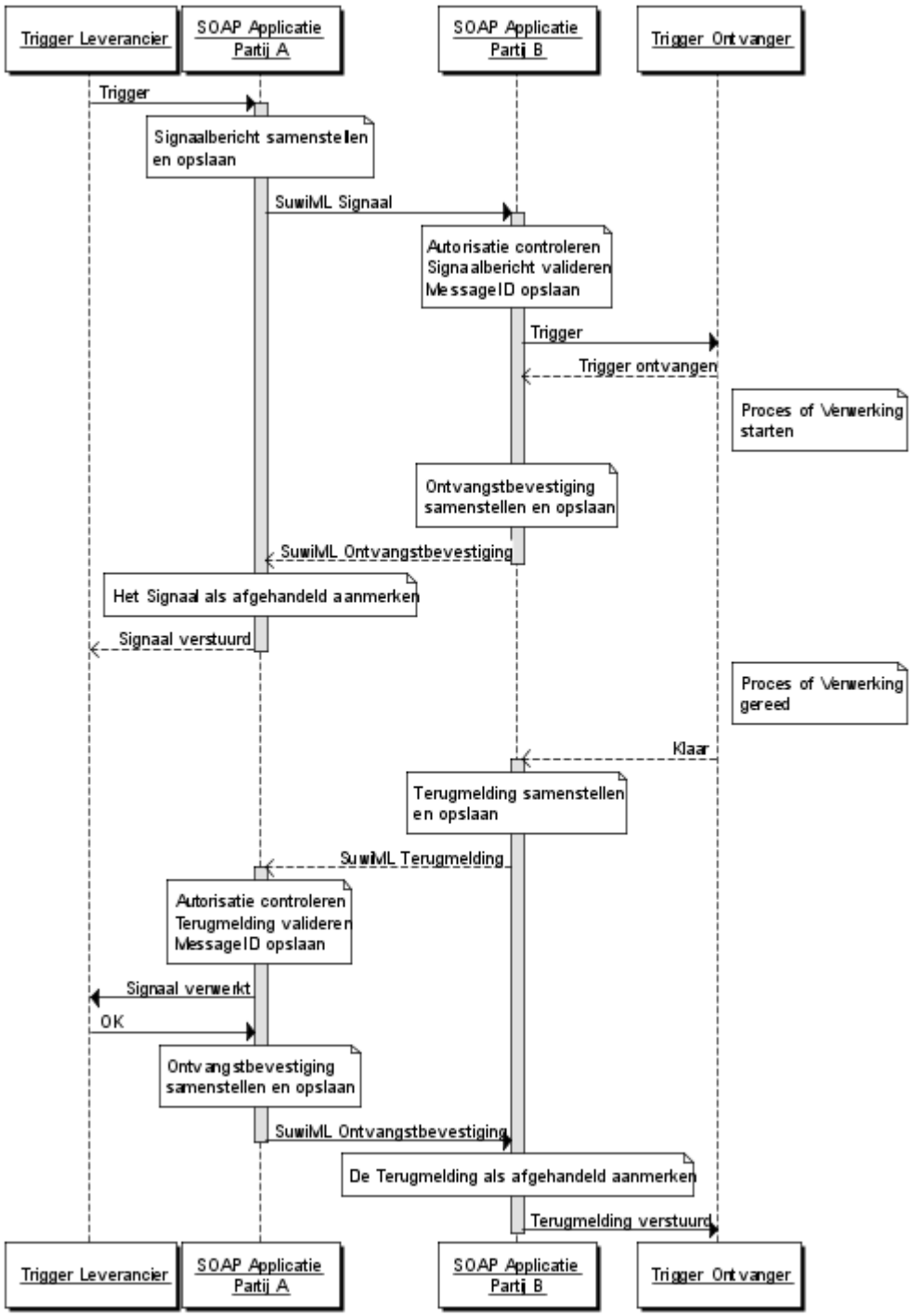
Afbeelding 41 SuwiML Signaalbericht naar aanleiding van een Trigger

Het is denkbaar dat Partij A een Terugkoppeling wenst te ontvangen na afronding van het Proces of de Verwerking bij Partij B. In dat geval treedt Partij A niet alleen op als Client voor het versturen van het Signaalbericht, maar ook als Server voor het ontvangen van een bij behorende Terugmelding, zie Afbeelding 42.

Vaak worden de Client Applicatie en de Server Applicatie in zo'n situatie verenigd in één SOAP Applicatie, en in Afbeelding 42 is het ook zo weergegeven. Technisch gezien is dat echter niet per sé noodzakelijk.

Het is belangrijk dat het implementeren van een SuwiML Terugmelding niet ten koste gaat van de SuwiML Ontvangstbevestiging. De Ontvangstbevestiging en de Terugmelding hebben namelijk een verschillende betekenis zoals uit Afbeelding 42 zal blijken.

Voor de implementatie van de Terugmelding gelden voor Partij A en Partij B dezelfde eisen op het gebied van opslag en eventueel opnieuw verzenden (nu door Partij B) en ontdubbelen (nu door Partij A) als bij het implementeren van het Signaalbericht.

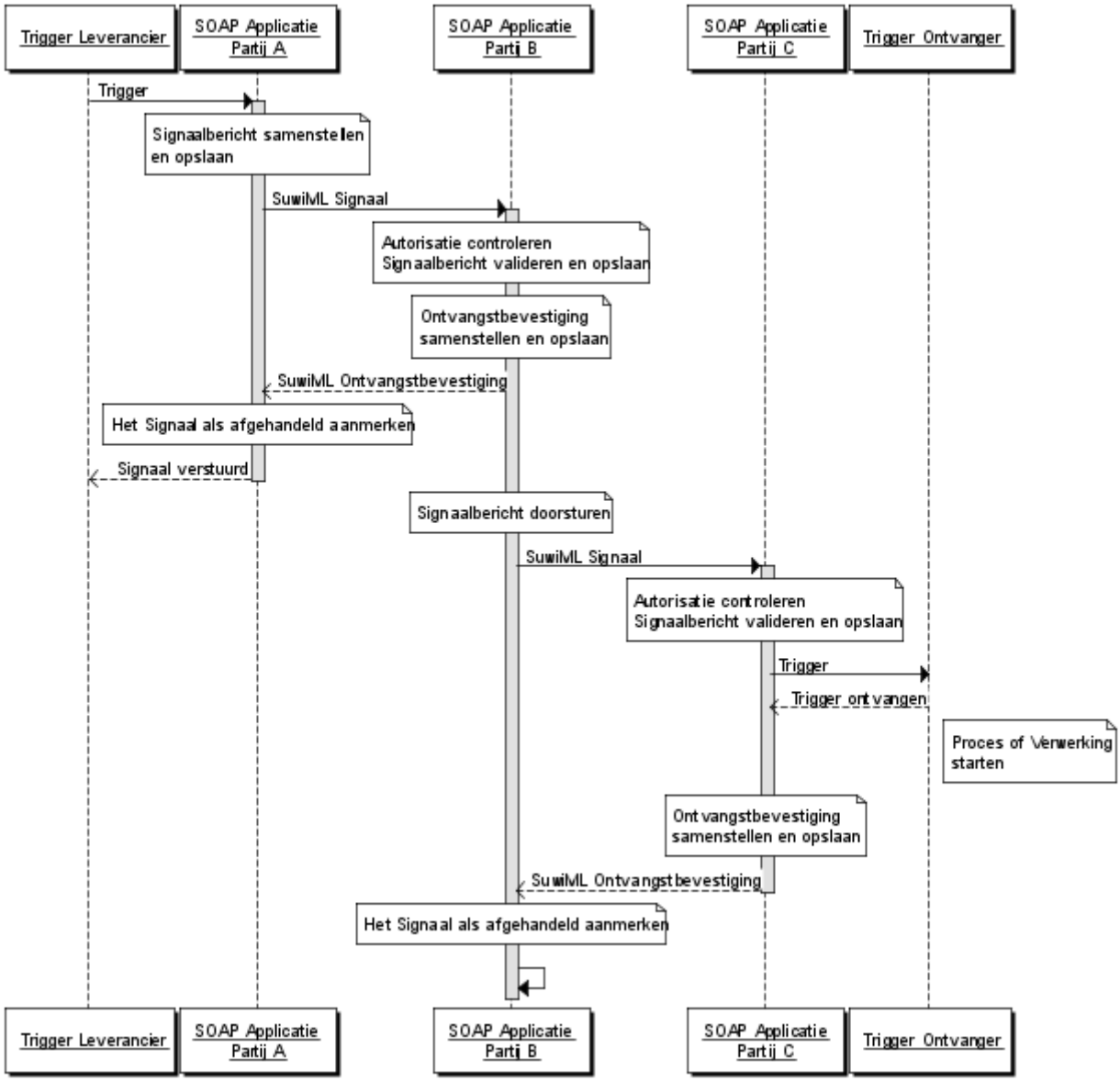


Afbeelding 42 SuwiML Signaal met een bijbehorende SuwiML Terugmelding

Het kan zijn dat het 'Proces of Verwerking' bij Partij B slechts het dóórsturen van het Signaalbericht naar nog een andere Partij C behelst. Partij B fungeert in dat geval als een soort SOAP Tussenstation, zie Afbeelding 43.

Een voorbeeld van het scenario van Afbeelding 43 is een Bericht-uitwisseling waarbij er gebruik gemaakt wordt van RINIS. RINIS speelt dan de rol van Partij B. RINIS levert extra diensten, bovenop het transport van een Bericht, op het gebied van Logging, Beveiliging en Store-and-Forward. RINIS verandert niets aan de inhoud van het Bericht. RINIS vermeldt hoogstens zichzelf als Tussenstation in de SuwiML Header van het door te sturen Signaal, indien gewenst. Verder veranderen zij ook de SOAP structuur niet van het Bericht. Het koppelvlak tussen Partij A en Partij B (=RINIS) en het koppelvlak tussen Partij B en Partij C zijn daarom in het RINIS voorbeeld

precies hetzelfde. Het koppelvlak kan in dat geval dan ook met één WSDL beschreven worden. De koppelvlak-specificaties worden in dat geval op twéé plekken geïmplementeerd. Ten eerste op een RINIS server die lokaal bij Partij A in huis geplaatst wordt. De interne inrichting van RINIS bestaat er uit dat het Bericht vervolgens getransporteerd wordt over het RINIS netwerk naar een RINIS server die lokaal bij Partij C in huis staat. Aldaar zal de RINIS server in huis bij Partij C de tweede implementatie (gebouwd door Partij C) van de koppelvlak-specificaties aanspreken op een server van Partij C.



Afbeelding 43 Een SuwiML Signaalbericht van Partij A dat door Partij B doorgestuurd wordt naar Partij C

10. Afsluiting

Ter afsluiting een klein overzicht van de technische hulpmiddelen die nuttig waren bij het schrijven van de Transactiestandaard, en die ook van waarde kunnen zijn bij het implementeren van webservices door de verschillende partijen.

Tenslotte ook een blik op de verder toekomst voorbij deze versie van de SuwiML Transactiestandaard.

10.1. Gebruikte middelen

Er zijn experimenten uitgevoerd met Axis2, soapUI, TCPMon en de Digikoppeling CompliancyService van Digikoppeling.
Met het Axis2 script wsdl2java.bat kun je Java code genereren op basis van een WSDL file. Om server-side code te genereren:

```
wsdl2java.bat -uri <pad>\SuwiML\Diensten\VoorbeeldService\VoorbeeldService.wsdl  -ss -sd -p
nl.bkwi.SuwiML.diensten.voorbeeldservice
```

Naast de Java Code wordt er een build.xml Ant script gegenereerd. Door 'ant' op die build.xml los te laten wordt de Java code gecompileerd en in een .aar Axis2 Archive gestopt, tezamen met een configuratiefile services.xml. Vanuit de Axis2 web-applicatie kan die .aar file ge-upload worden zodat de service beschikbaar komt in de webserver.

De Sequence Diagrammen in Hoofdstuk 9 Scenario's en Sequence Diagrammen zijn gemaakt met behulp van de handige online tool op http://www.websequencediagrams.com/.

10.2. Groeipad

Het is wenselijk dat de SuwiML Transactiestandaard doorgroeit naar de relevante nationale en internationale standaarden. De eerste stappen daarvoor worden met deze versie gezet door bijvoorbeeld het invoeren van WS-Addressing ten koste van de desbetreffende velden in de oude SuwiML Header. Er worden nu ook al wenselijke vervolgstappen gezien die in volgende versies van de SuwiML Transactiestandaard beschreven zullen worden. Mogelijke vervolgstappen kunnen zijn:

- Invoeren ondersteuning voor WS-ReliableMessaging
- Invoeren ondersteuning voor WS-Security
- Asynchrone bericht-uitwisseling
- Gebruik van Fast Infoset
- REST-full webservices

Of de rol van de SuwiML Transactiestandaard geheel overgenomen zal worden door bijvoorbeeld de standaarden van Digikoppeling, of door een set van WS-* standaarden, of door een WS-I Basic Profile, valt nog te bezien. De SuwiML Transactiestandaard zal de komende jaren nog wel van waarde blijven, om de afhankelijkheid van externe ontwikkelingen te beperken, om een eigen tempo te kunnen bepalen, en om de mogelijkheid te behouden eigen keuzes in de Suwi keten te maken.