Logius - Vastgestelde versie

Digikoppeling Best Practices WUS 1.10

Logius Best practice
Vastgestelde versie 01 oktober 2017

Overzicht standaarden

Deze versie:
https://publicatie.centrumvoorstandaarden.nl/dk/bpwus/1.10

Laatst gepubliceerde versie:
https://publicatie.centrumvoorstandaarden.nl/dk/bpwus/

Laatste werkversie:
https://logius-standaarden.github.io/Digikoppeling-Best-Practices-WUS/

Redacteurs:
Peter Haasnoot (Logius)
Pieter Hering (Logius)

Auteur:
Logius

Doe mee:
GitHub Logius-standaarden/Digikoppeling-Best-Practices-WUS
Dien een melding in
Revisiehistorie
Pull requests

This document is also available in this non-normative format: pdf

---

Samenvatting

Alle Digikoppeling webservices die op WUS gebaseerd zijn, moeten conformeren aan Digikoppeling Koppelvlakstandaard WUS. Dit document beschrijft een aantal Best Practices voor het gebruik van de Koppelvlakstandaard WUS.

Status van dit document

Dit is de definitieve versie van de best practice. Wijzigingen naar aanleiding van consultaties zijn doorgevoerd.

Inhoudsopgave

Colofon

Logius Servicecentrum:	Postbus 96810 2509 JE Den Haag t. 0900 555 4555 (10 ct p/m) e.servicecentrum@logius.nl

1. Inleiding

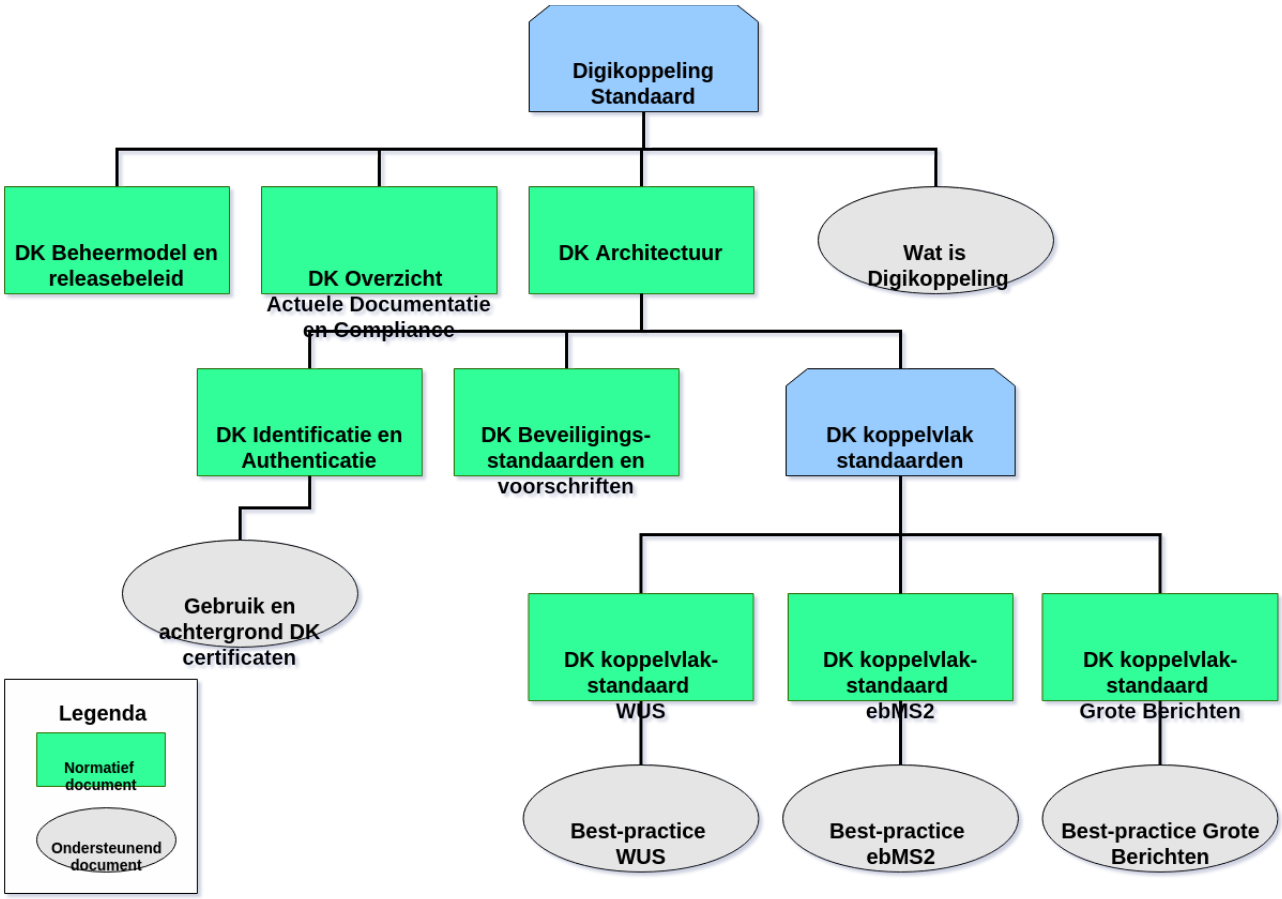
1.1 Doel en doelgroep

Alle Digikoppeling webservices die op WUS gebaseerd zijn, moeten conformeren aan Digikoppeling Koppelvlakstandaard WUS. Dit document is een aanvulling hierop. Het heeft als doel ontwikkelaars te adviseren en te informeren over de huidige werkwijze bij het toepassen van Digikoppeling Koppelvlakstandaard WUS – deze informatie geldt dus alleen voor de WUS-variant.

Het document is bestemd voor ontwikkelaars van webservices, die Digikoppeling toepassen. Het gaat hierbij om zowel service providers als service requesters (clients).

1.2 Opbouw Digikoppeling documentatie

Digikoppeling is beschreven in een set van documenten. Deze set is als volgt opgebouwd:



Figuur 1 Opbouw documentatie Digikoppeling

▶ **Tekstalternatief**

1.3 Digikoppeling

1.3.1 Doel en scope van Digikoppeling

Voor de Overheid als geheel is interoperabiliteit tussen een groot aantal serviceaanbieders en serviceafnemers van essentieel belang. Die grootschalige interoperabiliteit wordt bereikt door een sterke standaardisatie van het koppelvlak tussen de communicatiepartners.

Deze communicatie vindt plaats in het domein van Digikoppeling, en daarbij worden Digikoppeling koppelvlakstandaarden toegepast. Dat is een zeer beperkte set van standaarden waaruit onder gedefinieerde omstandigheden gekozen kan worden.

Digikoppeling biedt de mogelijkheid om op die sterk gestandaardiseerde wijze berichten uit te wisselen tussen serviceaanbieders (service providers) en serviceafnemers (service requesters of clients). Digikoppeling richt zich (in elk geval voorlopig) uitsluitend op uitwisselingen tussen overheidsorganisaties.

1.3.2 Uitwisseling binnen Digikoppeling

De uitwisseling tussen service providers en - requesters is in drie lagen opgedeeld:

- Inhoud: deze laag bevat afspraken over de inhoud van het uit te wisselen bericht, dus de structuur, semantiek en waardebereiken . Digikoppeling houdt zich niet met de inhoud bezig, “heeft geen boodschap aan de boodschap”.
- Logistiek: op deze laag bevinden zich de afspraken betreffende transportprotocollen (HTTP), messaging

(SOAP), adressering, beveiliging (authenticatie en encryptie) en betrouwbaarheid. Dit is de laag van Digikoppeling.

- Transport: deze laag verzorgt het daadwerkelijke transport van het bericht.

Digikoppeling richt zich uitsluitend op de logistieke laag. Deze afspraken komen in de koppelvlakstandaarden en andere voorzieningen.

1.4 Opbouw van dit document

Hoofdstuk 1 bevat een aantal algemene inleidende onderwerpen.

Hoofdstuk 2 bevat aanbevelingen, werkwijzes en Best Practices.

1.5 Gehanteerde terminologie: Digikoppeling Glossary

Voor de definities die binnen het Digikoppeling project gehanteerd worden, zie de ‘Digikoppeling Glossary’ die via Digikoppeling website te vinden is.

2. Werkwijze/Aanbevelingen/Best Practices

2.1 Servicedefinities

Deze paragraaf bevat de aanbevelingen t.a.v. het omgaan met servicedefinities.

2.1.1 WSDL

In WSDL 1.1 is een Authoring Style advies (zie hieronder) opgenomen: “separate the definitions in three documents: data type definitions, abstract definitions, and specific service bindings”. Dit advies, met name het apart beschrijven van de “specific service bindings” (WSDL onderdelen Binding en Service) wordt overgenomen.

Onderstaande tekst is een citaat uit de WSDL 1.1 Standaard. Het vormt een door Digikoppeling overgenomen best practice bij het schrijven van duidelijke WSDL’s.

> *Wijze van notatie (© W3C Note 15 March 2001, vertaald uit het Engels)*

> *“Door gebruik van het importelement wordt het mogelijk, de diverse elementen van een servicedefinitie te scheiden in afzonderlijke documenten, die dan naar behoefte geïmporteerd kunnen worden. Met behulp van deze techniek kun je duidelijker servicedefinities schrijven, omdat de definities gescheiden worden al naar gelang hun abstractieniveau. Het vergroot ook het vermogen om allerlei soorten servicedefinities nogmaals te gebruiken. Het gevolg is, dat WSDL documenten die op deze manier gestructureerd zijn, gemakkelijker te gebruiken en te onderhouden zijn. In het 2e voorbeeld hieronder is te zien hoe deze manier van schrijven gebruikt wordt om de service in het 1e voorbeeld* te definiëren. Hier scheiden we de definities in drie documenten: gegevenstype definities, abstracte definities en specifieke service bindings. Natuurlijk is het gebruik van dit mechanisme niet beperkt tot de definities uit het voorbeeld, dat alleen taalelementen gebruikt die in deze specificatie gedefinieerd zijn. Andere typen definities, die op meerdere taalextensies gebaseerd zijn, kunnen op dezelfde manier gecodeerd en hergebruikt worden.”*

* Het voorbeeld waar deze tekst van de WSDL 1.1 standaard naar verwijst wordt is niet in overeenstemming met het Digikoppeling Koppelvlakstandaard WUS profiel (zie voor details CRF 12). Om een indruk te krijgen van de opdeling (authoring style) van een WSDL wordt verwezen naar de berichtvoorbeelden. De voorbeelden van

berichten zijn gepubliceerd op de Logius website. Voorbeelden van WSDL's zijn beschikbaar als onderdeel van de Digikoppeling Compliance Voorziening.

Voor de specificatie van zaken die buiten het bereik van de WSDL vallen (TLS, WS-Security) wordt aanbevolen om in de WSDL van een service "documentation elements" (<(wsdl:documentation> of <!-- xxx -->) op te nemen die de eisen ten aanzien van metadata verwoorden, of een verwijzing naar betreffende documenten bevat.

2.1.2 Karakterset en codering

Voor communicatie binnen het Digikoppeling WUS kanaal wordt de UCS karakterset (ISO/IEC 10646) gebruikt. Deze karakterset omvat (is superset van) de set Unicode, Latin (ISO/IEC 8859-x) en de GBA karakterset.

Bij berichtenverkeer speelt de gebruikte karakterset een belangrijke rol. Een serviceafnemer kan in de vraagaanroep karakters gebruikt hebben die niet door de serviceaanbieder ondersteund worden. Voor goede communicatie is het dus belangrijk dat hiervoor afspraken gelden. UCS is de meest uitgebreide karakterset. Toepassen van UTF-8 zorgt er voor dat efficiënt omgegaan wordt met het aantal bytes in het bericht.

2.1.3 Versie aanduiding

Er zijn een aantal elementen waaraan een versie aanduiding moet worden toegevoegd. Dit zijn:

WSDL/namespace

WSDL/Servicenaam

WSDL/PortType

WSDL/Type(s) (XSD) namespace

Er zijn een aantal manieren om de versie van een service aan te duiden. De meest gangbare zijn "Major.Minor", "Enkelvoudige versie" (bijv V1) en "YYYY/MM".

Het voorstel is om voor zowel de XSD als de WSDL de Enkelvoudige versie aanduiding te gebruiken.

Waarom gebruiken we geen major.minor? Er zijn verschillende mogelijkheden om Minor wijzigingen backward compatible te houden, deze worden echter als erg omslachtig beschouwd en/of ze vereisen speciale tooling ondersteuning. Daarom wordt voorgesteld geen onderscheid tussen major en minor te maken, en dus alleen met enkelvoudige versies te werken. Dit heeft als resultaat dat de WSDL en XSD namespace dus alleen de "Enkelvoudige" aanduiding, zoals _v1 krijgt.

Een aantal voorbeelden:

WSDL namespace "http://wus.osb.gbo.overheid.nl/wsdl/compliance-v1"

XSD namespace "http:// wus.osb.gbo.overheid.nl/xsd/compliance/xsd/compliance-v1".

Servicenaam "OSBComplianceService_v1"

PortType "IOSBComplianceService_v1"

De aanduiding YYYY/MM slaat ook op een enkelvoudige versie, d.w.z. zonder onderscheid tussen major en minor. Die aanduiding kan dus ook gebruikt worden. Het lijkt echter aan te bevelen om versies van webservices aan te duiden met (oplopende) versienummers, omdat communicatie daarover iets eenduidiger is.

WB005

Voor het onderscheid tussen test- en productieservices heeft het de voorkeur dat deze twee op aparte machines komen met een eigen DNS naam (en dus met verschillende PKI overheid certificaten).

Aan de locatie (uri) van de service is daardoor te zien of het om een productie- of een testservice gaat.

2.1.4 XSD

Gebruik van document/literal wrapped style. In Digikoppeling Koppelvlakstandaard WUS staat de bij voorschrift WW003 dat bij de document literal style de body maar 1 element mag bevatten. Het wordt sterk aangeraden dat dit element de operatie naam bevat voor een bepaald bericht. Deze wordt dus door de xsd beschreven en bevat een beschrijving van de payload. Door deze methode te gebruiken wordt de interoperabiliteit verhoogd, met name tussen Microsoft en andere omgevingen. (zie http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl)

Wsdl definition

```
...
<types>
    <schema>
        <element name="myMethod">
            <complextype>
                <sequence>
                    <element name="x" type="xsd:int">
                    <element name="y" type="xsd:float">
                </element></element></sequence>
            </complextype>
        </element>
        <element name="myMethodResponse">
            <complextype>
        </complextype></element>
    </schema>
</types>
<message name="myMethodRequest">
    <part name="parameters" element="myMethod">
</part></message>
<message name="empty">
    <part name="parameters" element="myMethodResponse">
</part></message>
<porttype name="PT">
    <operation name="myMethod">
        <input message="myMethodRequest">
        <output message="empty">
    </output></operation>
</porttype>
...
```

bericht:

```
...
<envelope>
    <payloadbody>
        <mymethod>
            <x>
                5
            </x>
            <y>
                5.0
            </y>
        </mymethod>
    </payloadbody>
</envelope>
...
```

2.1.5 Namespaces

Met betrekking tot het verkrijgen van eenduidigheid in de WSDL bestanden, wordt sterk aangeraden dat xml namespace prefixen volgens de WSDL 1.1 specificatie gebruikt worden. (Web Services Description Language (WSDL) 1.1).

Het heeft de voorkeur dat een namespace wordt opgebouwd op basis van de domeinnaam van de web service.

2.1.6 Naamgeving conventies

Over het algemeen moet de service naam een goede weerspiegeling zijn van de context waarin de service wordt gebruikt. De operatienamen die door deze service ondersteund worden, moeten passen binnen de context van de service en overdadige lange tekststrings moet worden voorkomen.

2.1.6.1 Contract First vs Contract Last

Het ontwikkelen van webservices kan grofweg ingedeeld worden in twee categorieën, namelijk:

- contract first
- contract last

Met ‘contract’ wordt gewezen op het WSDL contract dat de webservice definieert. Nagenoeg alle webservice toolkits ondersteunen beide manieren van ontwikkeling.

Bij een contract first approach wordt in feite gestart met het handmatig samenstellen van het WSDL. Gebaseerd op het WSDL wordt vervolgens door toolkits de code gegenereerd. De ontwikkelaar maakt gebruik van de genereerde code om vervolgens een implementatie te schrijven. Hierbij maakt de ontwikkelaar gebruik van technieken als interfaces en overerving.

Bij Contract last approach is dit proces net andersom. Een ontwikkelaar begint met het ontwikkelen van de logica en het schrijven de code. Daarna voegt de ontwikkelaar meta informatie toe aan de code waarmee bepaald wordt hoe de WSDL eruit ziet. Meta informatie verschilt per toolkit/framework hoe dit gedaan wordt. Binnen de Javawereld wordt dit vaak gedaan met jaxws annotations. Bij het opstarten van de applicatie wordt vervolgens door de toolkit bepaald welke meta informatie het tot zijn beschikking heeft en zal vervolgens op runtime een WSDL publiceren.

Beide manieren hebben hun voor- en nadelen. Ervaring leert wel dat bij trajecten waarbij strikte eisen worden gesteld met betrekking tot de WSDL dat een contract first approach uiteindelijk de meest robuuste manier is. Doordat een WSDL volledig handmatig wordt samengesteld, is dit ook de meest flexibele en toolkit-onafhankelijke manier van werken.

2.2 Foutafhandeling

Bij berichtenuitwisseling tussen een service requester en service provider kunnen fouten optreden. Het is van belang dat er nagedacht wordt over het nut van het communiceren van een bepaalde foutmelding. De beschrijving van een of andere interne fout bij een service provider zal in het algemeen niet interessant zijn voor een requester; het terugmelden van de specifieke oorzaak heeft dan geen zin.

Als de oorzaak van de fout bij de service requester ligt, dan dient de foutmelding er op gericht te zijn dat de service requester op basis van de foutmelding de fout kan achterhalen en actie ondernemen.

Voor communicatie in het Digikoppeling domein volgens de Koppelvlakstandaard WUS zijn vier verschillende fouttypen te onderkennen: protocolfouten, transportfouten, functionele fouten en technische fouten.

2.2.1 Protocol- en transportfouten (dus TLS of HTTP fouten)

Protocol- en transportfouten zijn in het algemeen in het protocol gedefinieerd. Wijzigingen daarin - dus aanpassing van standaard software - zijn niet wenselijk. Protocol- en transportfouten worden daarom niet beschreven. De hier beschreven foutmeldingen hebben betrekking op situaties waarin het requestbericht door de web service is ontvangen. Deze kan het echter niet goed verwerken en stuurt daarom een foutmelding terug.

2.2.2 Functionele fouten

Functionele fouten zijn in het kader van Digikoppeling moeilijk te standaardiseren. Deze zullen voor veel organisaties verschillen en ook het communiceren van de foutmelding zal niet altijd eenduidig zijn. Dit is weliswaar iets wat om aandacht vraagt, maar het valt buiten de scope van Digikoppeling.

2.2.3 Technische fouten

Voor technische foutmeldingen kan een standaard bericht gedefinieerd worden. In de SOAP specificatie is de SOAP Fault beschreven die je hiervoor goed kunt gebruiken.

Communiceren van een fout via een SOAP Fault heeft een aantal voordelen:

Uitzonderingen op een consistente manier afgehandeld worden;

De SOAPFault wordt beschreven in de SOAP specificatie;

De verschillende elementen waaruit een SOAP Fault is opgebouwd biedt de mogelijkheid tot het communiceren van uitgebreide informatie;

De FaultCode kan aanduiden of de fout was veroorzaakt door Client of Server.

Een aantal nadelen zijn:

Soapfaults kunnen geen binding (HTTP) gerelateerde fout communiceren. In dat geval wordt over het onderliggende protocol de fout gecommuniceerd

Bij een SOAPFault bericht mag geen additionele data toegevoegd worden

Het ‘detail’ element van de SOAP Fault is bedoeld om applicatie specifieke foutmeldingen te communiceren die gerelateerd zijn aan het SOAP ‘Body’ element. Het moet aanwezig zijn in de SOAP Fault indien de fout werd veroorzaakt door het ‘Body’ element. Indien er geen ‘detail’ element in de SOAP Fault aanwezig is, dan betekent dit dat de fout niet is ontstaan bij het verwerken van het ‘body’ element.

Voor een web service in het Digikoppeling domein moeten foutmeldingen gedefinieerd worden

2.3 WS-Addressing

2.3.1 Maak gebruik van MessageID

Binnen WS-Addressing wordt de wsa:MessageID gebruikt om een bericht uniek te definiëren. Dit veld is verplicht binnen de specificatie. De meeste frameworks/toolkits genereren daarom standaard een unieke messageID voor elk bericht indien deze niet is meegegeven door de applicatie.

Hoewel dit in de meeste gevallen prima werkt, is het aan te raden om gebruik van de MessageID en deze via de applicatie te laten bepalen.

Bijvoorbeeld in het scenario dat een bericht wordt verstuurd door een andere interne applicatie proces, kan door gebruik te maken van het interne proces nummer als onderdeel van de messageID, een correlatie over

verschillende processen tot stand gebracht worden. Van een foutbericht kan via de relatesTo eenvoudig bepaald worden welke interne applicatie proces een fout heeft veroorzaakt.

Daarnaast zou het formaat van een messageID ook gebruikt kunnen worden om naast een unieke waarde ook gedeeltelijk aan te vullen met een logische waarde. Indien bijvoorbeeld gewerkt wordt met een interactie waarbij meerdere berichten uitgewisseld worden voor 1 business conversatie, kan het correleren versimpeld worden door een conversationID te verwerken in de messageID.

Voorkeur is om consistentie in de opbouw van de messageID aan te houden. De volgende opbouw heeft de voorkeur: “CUSTOM@UUID@URI “ of “CUSTOM@GUID@URI”. UUID of GUID volgens http://www.ietf.org/rfc/rfc4122.txt

URI is een anyURI volgens http://www.w3.org/2001/XMLSchema

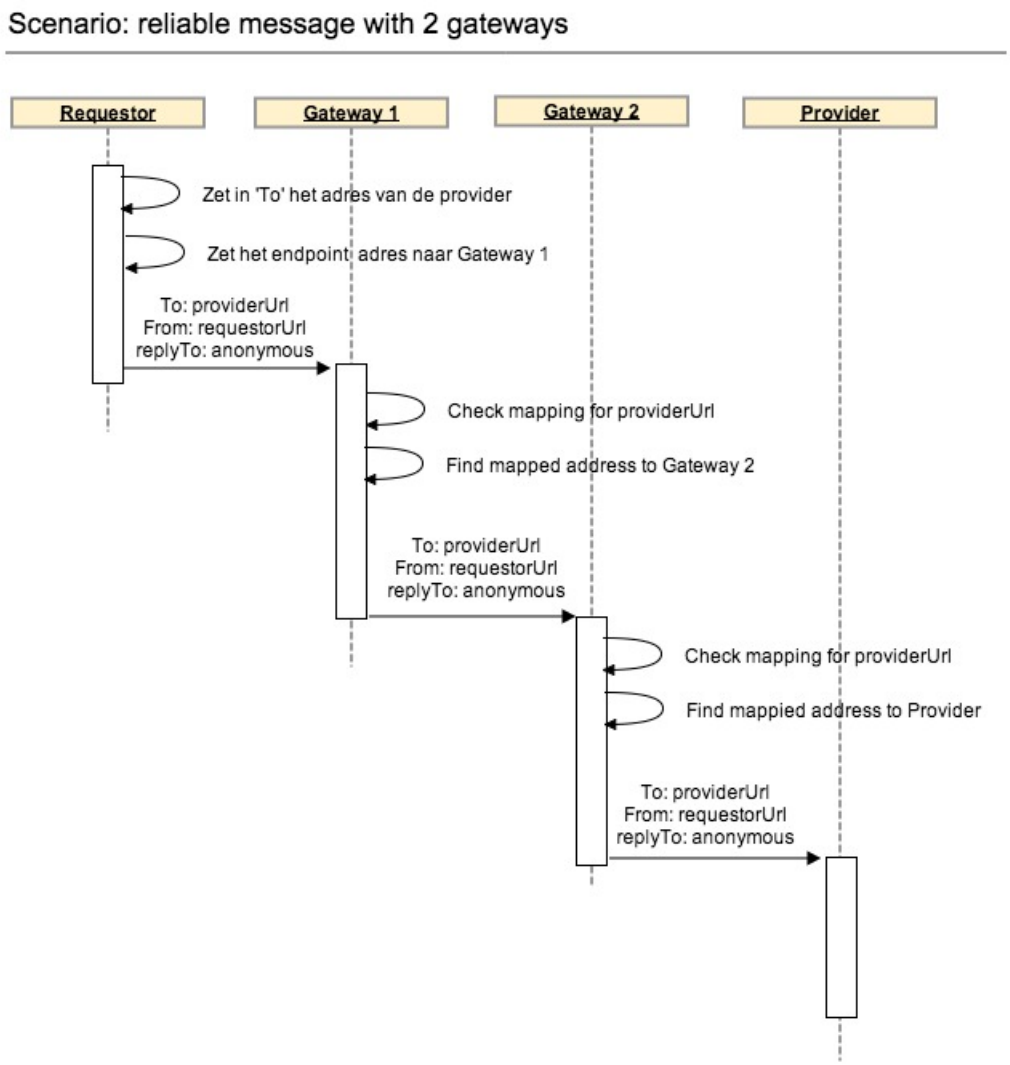
De URI kan de domeinnaam zijn van Digikoppeling messagehandler of de web service namespace.

2.3.2 Routeren van berichten over meerdere intermediairs

WS-Addressing biedt de mogelijkheid om via vaste metadata informatie berichten te voorzien van routeer informatie. Hiervoor gebruikt men met name het ‘TO’ adres. Het is aan te raden om in het TO adres, het beoogde eindadres op te nemen. De endpoint die het bericht ontvangt kan door middel van de waarde van TO adres bepalen hoe het bericht doorgezet wordt. Intern dient de intermediair een mapping tabel bij te houden naar wie een bericht doorgestuurd moet worden afhankelijk van de TO waarde van het bericht. Dit kan dus naar de eindbestemming zijn, of weer naar een andere intermediair. De mapping inrichting dient vooraf afgesproken en ingericht te zijn.

Het is geen optie om het TO adres continu te herschrijven bij elke intermediair, waarin de TO adres de waarde krijgt waar het naar toe moet gaan. Dit is namelijk niet mogelijk, als het bericht ondertekend is want de TO waarde is onderdeel van de ondertekening en mag dus niet gewijzigd worden.

Ter verduidelijking, de volgende sequence diagram:



Figuur 2 sequence diagram routeren van berichten over meerdere intermediairs

Zoals in het diagram getoond wordt, blijft de addressing informatie gelijk tijdens de hele verzending. Indien het bericht ondertekend en versleuteld is, hoeven gateway 1 en gateway 2 het bericht niet te valideren of ontcijferen. Zolang de addressing informatie niet veranderd wordt, is de meegestuurde 'signature' nog steeds valide. De gateway componenten lezen enkel de To address uit, om te bepalen waar het bericht naartoe gestuurd moet worden.

2.3.3 Afhandeling From in combinatie met een proxy/gateway

Een from adres geeft aan waar het bericht vandaan is gekomen. Dit adres wordt vervolgens gebruikt om te bepalen waar het bericht vandaan is gekomen. In complexe infrastructuur oplossingen waarbij gebruik is gemaakt van een reverse proxy kan dit voor bepaalde complicaties zorgen (zie http://en.wikipedia.org/wiki/Reverse_proxy). Een server kan namelijk niet het from adres als endpoint gebruiken omdat deze moet wijzen naar de interne proxy. Extern verkeer dient vaak in dergelijke situaties altijd over de proxy te gaan. Om dit probleem op te lossen dient ook hier gebruik gemaakt te worden van een endpoint mapping functionaliteit.

2.4 WS-Policies

WS-Policies is onderdeel van de WS-* specificaties en is sinds 2007 is versie 1.5 final. De doelstelling van WS-Policies is om een framework aan te bieden waarmee domein specifieke definities in een WSDL opgenomen kunnen worden. Gebaseerd op deze framework kunnen specifieke WS-* standaarden extensies definiëren op

een eenduidig manier. De extensies worden dan in de policy elementen toegevoegd zodat ze vast gelegd zijn in de WSDL. Frameworks/toolkits kunnen tijdens het inlezen van de WSDL dan al bepalen hoe met de policy elementen omgegaan moet worden en kunnen hiervoor al de voorbereidingen treffen. Bijvoorbeeld: indien op WSDL niveau al aangegeven is dat een bericht gesigned moet worden, kan een framework zoals Apache CXF al de benodigde interceptor registreren die het signing proces afhandelt. Dit gebeurt dan transparant voor de ontwikkelaar.

Hoewel de standaarden reeds geruime tijd zijn uitgewerkt en gefinaliseerd, is in de praktijk de implementatie ondersteuning nog wisselvallig. Hierdoor is gekozen om het gebruik van policies niet binnen de koppelvlakstandaard op te nemen maar optioneel te houden. De meerwaarde van het gebruik van policies is daarentegen dusdanig evident dat het binnen de best practices is opgenomen.

Het gebruik van policies kunnen verschillende zaken zowel versimpelen als verduidelijken in de ontwikkelfase maar zijn niet noodzakelijk om ingezet te worden. In de praktijk blijkt wel dat zaken als signing en encryption zonder policies tijdens het ontwikkelen voor de nodige complexiteit zorgt. Uiteraard is het onderhouden van meerdere WSDL varianten voor dezelfde service niet wenselijk. Daarentegen kan de impact daarvan beperkt worden door te werken met een externe policy file dat in de WSDL wordt geïmporteerd en vervolgens met policy references aan specifieke methodes te koppelen. Door de policies extern te houden, is het zelfs mogelijk om verschillende policy varianten te maken indien een veel gebruikte framework een policy niet interoperabel ondersteund. De kans op implementatie fouten zou hiermee verlaagd kunnen worden wat zal resulteren in een sneller adoptieproces. De baten die hiermee behaald kunnen worden zijn hoogstwaarschijnlijk hoger dan de kosten die ermee gemoeid zijn.

Voorbeeld: Partij A besluit om een melding dienst aan te bieden waar andere partijen zich kunnen abonneren en meldingen kunnen ontvangen. Dit houdt in dat alle partijen de service moeten implementeren om de meldingen te kunnen ontvangen. Indien er 1000 verschillende partijen zijn waarbij het bekend is dat 50% gebruik maakt van Oracle Webcenter technologie en 30% gebruik maakt van Microsoft technologie. Dan zou de case om een policy attachment aan te bieden voor beide technologieën dat getest is op een correcte werking zeer goed te maken zijn. Door het inlezen van de WSDL met de bijbehorende policy attachments staan alle webservice instellingen zoals signing onderdelen gelijk goed.

3. Technische foutmeldingen

3.1 Categorieën

Bij gegevensuitwisseling kunnen er fouten optreden door verschillende oorzaken. Fouten kunnen in één van de volgende categorieën ingedeeld worden:

1. syntax fouten, hebben betrekking op de structuur van de berichten (XSD) en standaarden zoals WSA en SOAP
2. inhoudelijke fouten, hebben betrekking op inhoudelijke verwerking en zijn context/domein of sector specifiek en worden niet binnen DK gestandaardiseerd
3. protocolfouten, hebben betrekking op TLS of HTTP
4. fouten doordat een service niet (onvoldoende QoS) beschikbaar is, waaronder ook time-out en autorisatie problemen/fouten.

Per categorie kan op hoofdlijnen een procedure voor de foutafhandeling gedefinieerd worden.

5. bij syntax fouten dient zo mogelijk aangegeven te worden welk element fout is (zoals in foutmeldingen 0005 t/m 0008 aangegeven staat)
6. Zo mogelijk aangeven waarom. Bij inhoudelijke fouten aangeven dat het bericht vanwege inconsistentie niet verwerkt kan worden. (dit is eigenlijk geen transport/koppelvlak probleem, maar veeleer een business

probleem met een bijbehorende afhandelingsprocedure, vgl. de terugmelding in het stelsel).

3.2 Codes

Lijst van technische foutmeldingen met classificatie naar fout-categorieën

Code	Omschrijving	Categorie	toelichting
0001	Invalide soap envelope	1	Voldoet niet aan verwachte syntax. Structuur van de envelope matcht niet met wat er verwacht wordt
0002	Niet geautoriseerd	3	Service niet beschikbaar (QoS). Door gebrek aan bevoegdheden.
0003	Invalide soapaction	1	De inhoud leidt niet tot een voltooide actie, is niet gedefinieerd of onbegrijpelijk. Protocol fout
0004	Niet conform xsd	1	Voldoet niet aan verwachte syntax
0005	WS-Addressing header “to” ontbreekt	1	Ontbreekt of voldoet niet aan verwachte syntax
0006	WS-Addressing header “action” ontbreekt	1	Ontbreekt of voldoet niet aan verwachte syntax
0007	WS-Addressing header “messageID” ontbreekt	1	Ontbreekt of voldoet niet aan verwachte syntax
0008	WS-Addressing header “relates to” ontbreekt	1	Ontbreekt of voldoet niet aan verwachte syntax
0009	Niet volgens UTF	1	Voldoet niet aan verwachte characterset
0010	Headers anders dan WSA-headers	1	Voldoet niet aan verwachte syntax
0011	Header andere waarde dan voorgeschreven	1	Voldoet niet aan verwachte spec/waarde
0051	Service niet beschikbaar	3	Service niet beschikbaar (QoS). Door gebrek aan resources/ verwerkingscapaciteit.

4. Conformiteit

Naast onderdelen die als niet normatief gemarkeerd zijn, zijn ook alle diagrammen, voorbeelden, en noten in dit document niet normatief. Verder is alles in dit document normatief.

5. Lijst met figuren

↑