Ontwerpregels en best practices voor StUF-berichten

Auteur: KING
Versie: 1.05
Status: In gebruik

Inhoudsopgave

Versie	Wijzigingen
0.1	Eerst versie
0.2	Feedback uit de StUF Expertgroep, zie het document 'feedback_best_practices.pdf' in de folder http://www.kinggemeenten.nl/gemma/gegevens-en-berichten-(stuf)/documenten/stuf/1_stuf_expertgroep/20110720
1.0	Document is goedgekeurd door de StUF Expertgroep. Alleen de metagegevens van het document zijn aangepast (datum, versie, status, etc.).
1.01	Wijziging in de inleiding van hoofdstuk 3 n.a.v. erratum ERR231.
1.02	Wijziging in paragraaf 2.2. n.a.v. Erratum ERR200 van de StUF Expertgroep.
1.03	Wijziging in de inleiding van hoofdstuk 3 en in hoofdstuk 4 n.a.v. erratum ERR232.

1.04	Paragraaf 2.1 toegevoegd n.a.v. het erratum ERR317 Aan de inleiding van hoofdstuk 3 n.a.v. het erratum ERR318 dat een koppelvlak mag afwijken van het default berichttype van een sectormodel waarop het is gebaseerd. Paragraaf 5.2 toegevoegd n.a.v. ERR0350.
1.05	N.a.v. ERR0403 is paragraaf 5.3.1 toegevoegd.

1 Inleiding

Op basis van de StUF-standaard kunnen concrete berichten gedefinieerd worden voor een bepaald toepassingsgebied of een bepaalde sector. Zo'n berichtenspecificatie wordt in StUF-terminologie ook wel een sectormodel genoemd. Vooralsnog schrijft StUF geen regels voor hoe een sectormodel moet worden ontworpen. De regels van StUF spelen op het niveau van XML-berichten en de ontwerper is vrij om zelf de bijbehorende schema's (XSD) te bedenken zolang het schema geldige StUF-berichten niet afkeurt. Als een sectormodel een StUF-bericht valideert dan betekent dat niet noodzakelijkerwijs dat het een geldig bericht is. Niet alle StUF regels kunnen op een natuurlijke wijze op schema niveau worden afgedwongen. Het is zelfs mogelijk om een sectormodel te definiëren zonder XML schema's. Het is wel een best practice, zoals later zal worden beschreven, om zoveel mogelijk in het schema af te dwingen.

In de afgelopen jaren zijn een aantal belangrijke best practices naar voren gekomen die sterk worden aanbevolen bij het ontwerpen van schema's op basis van StUF met betrekking tot sectormodellen, berichtcatalogi of koppelvlakken. Dit document is een aanzet om dergelijke ontwerpregels in kaart te brengen. Het is de bedoeling dat dit document gaande weg zal worden uitgebreid met nieuwe best practices.

Het staat de ontwerper van een sectormodel in bepaalde gevallen vrij om af te wijken van de in dit document beschreven best practices. Het is dan wel verplicht deze afwijkingen te vermelden in het keuzenVerStUFfing-document.

Bij het toevoegen van berichtcatalogi of koppelvlakken aan een bestaand sectormodel is de ontwerper niet vrij af te wijken van de best practices. In dat geval zijn de best practices voorschriften die moeten worden opgevolgd.

De best practices dragen bij aan de volgende in de praktijk belangrijke kenmerken van een sectormodel:

- Zo weinig mogelijk software aanpassingen bij uitbreidingen
- Flexibiliteit: uitbreidingen van functionaliteit hoeven niet in alle gevallen tot nieuwe namespaces te leiden. Geen wildgroei van sectormodellen die een veel zwaardere statuur hebben dan berichtcatalogi.
- Maximaal hergebruik van schema-onderdelen
- Optimale opdeling van schema's waardoor bij hergebruik niet teveel geïmporteerd hoeft te worden. Dit leidt tot betere performance bij validatie en parsing van schema's.

In sectie 2 worden best practices beschreven om vanuit een informatiemodel herbruikbare entiteitschema's af te leiden. In sectie 3 wordt beschreven hoe je bericht- en serviceschema's (wsdl's) kunt genereren of ontwerpen op basis van entiteitschema's. De best practices worden geïllustreerd aan de hand van een concreet voorbeeld in sectie 6.

2 Van informatiemodel naar entiteitschema

De StUF-standaard gaat ervan uit dat het domein waarvoor berichten gemaakt worden, beschreven is in een informatiemodel met de relevante objecttypen en hun eigenschappen. Voor de gemeentelijke basisgegevens is het RSGB zo'n model en voor de zaakgegevens het RGBZ. Voor de sector WOZ is de catalogus WOZ het informatiemodel. Zo hoort er onder elk sectormodel een

informatiemodel te liggen met de definitie van de gegevens.

De StUF-standaard gaat ervan uit dat de XML-elementen in de berichten één-op-één te herleiden zijn tot attributen en relaties van objecttypen in het informatiemodel. Voorwaarde hiervoor is dat in het informatiemodel de te gebruiken xml-tag voor een attribuut of relatie wordt gedefinieerd en dat in het informatiemodel een korte aanduiding of mnemonic voor alle onderkende objecttypen (een objecttype kan ook een relatie zijn) wordt gedefinieerd. Als een informatiemodel onverhoopt geen xml-tags en mnemonics definieert, dan zal dit in een bijlage bij het keuzenVerStUFfing-document gedefinieerd moeten worden.

2.1 De verStUFfing van het informatiemodel

StUF schema's zijn over het algemeen niet 1 op 1 te mappen op het bijbehorende informatiemodel. Bij het omzetten van een informatiemodel naar StUF schema's moeten namelijk keuzes gemaakt worden die verband houden met het verwachtte gebruik van de te modelleren entiteit, de voor de entiteit benodigde berichttypen en de praktijk van alle dag. Wordt de entiteit direct benaderd of altijd via een andere entiteit, zijn alleen kennisgevingsberichten nodig of ook vraagAntwoordberichten, zal men bij de verwerking van de entiteit tegen problemen aanlopen die met een speciale constructie in het bericht te voorkomen is. Zo kan besloten worden bepaalde entiteiten uit het informatiemodel plat te slaan in andere entiteiten en kan aangegeven worden hoe relaties tussen entiteiten gelegd moeten worden.

Het is een best practice om dit soort keuzes vast te leggen in een verStUFfingsdocument.
Een belangrijk onderdeel van een verStUFfingsdocument zijn de relatiegrafieken. Daarin worden de StUF-entiteiten grafisch weergegeven door middel van boomstructuren. Daarbij worden de volgende conventies gehanteerd:

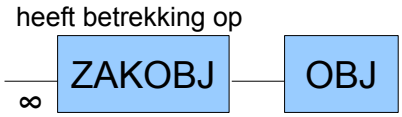
- Een fundamenteel entiteittype wordt opgenomen als een lichtblauw blokje met als naam een mnemonic XXX van drie letters.

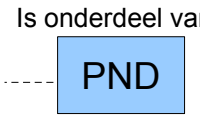
- De mnemonic voor een entiteittype kan worden voorafgaan door een aanduiding van het sectormodel waarin die mnemonic voorkomt. Bijvoorbeeld 'ztc:' om entiteittypen uit ImZTC aan te duiden en 'bg:' om entiteittypen uit het RSGB aan te duiden.
- Een relatie-entiteittype wordt opgenomen als een blokpijl met als naam een mnemonic XXXYYY /XXXYYYZZZ van zes of negen letters. De mnemonic XXX staat voor de entiteit van waaruit de relatie wordt gelegd. YYY staat voor de entiteit waarnaar de relatie wordt gelegd. In geval er meerdere relaties tussen XXX en YYY bestaan, dan kunnen deze worden onderscheiden door er een extra mnemonic ZZZ aan toe te voegen.

- Een relatie wordt alleen als lijn in de relatiegrafiek opgenomen, wanneer de relatiesoort geen eigen attribuutsoorten of relatiesoorten bevat. Wanneer de relatiesoort wel eigen attribuutsoorten of relatiesoorten bevat, dan wordt deze als blok in de relatiegrafiek opgenomen. Ook als de relatie ligt tussen dezelfde objecttypen en de betekenis verschilt afhankelijk van de richting van de relatie, wordt altijd een blok voor de relatie opgenomen.



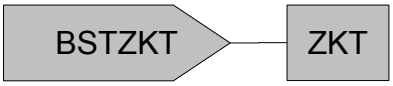
- Een gestreepte lijn voor een relatie wil zeggen dat de relatie alleen voorkomt in vraag/antwoord berichten en niet in kennisgevingberichten.



- Een gestreept blokje wil zeggen dat er voor de entiteit geen kennisgevingen zijn alleen vraagAntwoord berichten.

- Een groen blokje binnen een blauw blokje wil zeggen dat gegevens van de objectsoort in het groene blokje zijn platgeslagen in de StUF-entiteit waarin het groene blokje zich bevindt.

- Van een fundamenteel als gerelateerde worden de eventuele in de berichten voorkomende relatie-entiteiten niet getoond in de relatiegrafiek.
- Als de blokpijl en het gerelateerde blokje grijs zijn dan mag de relatie alleen voorkomen in vraag/antwoord berichten en in vrije berichten, maar niet in kennisgevingberichten.



- Boven de lijn voor de relatie is aangegeven hoe vaak een relatie maximaal in een StUF-entiteit mag voorkomen. Het symbool * wil zeggen dat een relatie een onbeperkt aantal keren kan voorkomen. Het minimum aantal is niet opgenomen, omdat binnen de basisschema's een relatie nooit verplicht is in een StUF-entiteit.

2.2 Van attribuutdomeinen naar simpleType's

Het informatiemodel definieert de domeinen voor de attribuutsoorten. In XML-Schema worden domeinen gedefinieerd als simpleTypes. Voor elk nog niet elders binnen de StUF-familie gedefinieerd domein uit een informatiemodel wordt in een sectormodel een simpleType gedefinieerd. Als voor een domein al ergens binnen de StUF-familie een simpleType is gedefinieerd, bijvoorbeeld in de StUF-standaard zelf of in een horizontaal sectormodel, dan wordt dit simpleType gebruikt.

Het is een best practice dat voor elk simpleType voor een domein ook als extension een complexType met simpleContent wordt gedefinieerd dat aan het simpleType de attributes StUF:noValue en StUF:exact toevoegt. Het attribute StUF:noValue is nodig voor de door StUF gedefinieerde bijzondere waarden geenWaarde, waardeOnbekend, waardeVastgesteldOnbekend, nietOndersteund en nietGeautoriseerd. Het attribute StUF:exact is nodig binnen vraagberichten. Deze complexTypes met simpleContent krijgen als naam de naam van het simpleType met daaraan toegevoegd "-e". Om het gebruik van StUF:noValue mogelijk te maken dient aan een element met een "-e"-complexType altijd het attribute xsi:nil="true" te worden toegevoegd. In een aantal gevallen, bijvoorbeeld in vrije berichten waar de gegevens in het bericht voor een bepaalde functie zo scherp mogelijk gedefinieerd worden, is het wenselijk om uit te sluiten dat xsi:nil="true" wordt gebruikt in een element. In dat geval dient in een restriction de waarde "true" voor het attribute nillable te worden verwijderd.

De StUF-standaard heeft, o.a. in het schema "stuf0310.xsd", simpleType's gedefinieerd voor de volgende domeinen:

- Sleutel
  Ten behoeve van identificerende sleutels heeft StUF het simpleType Sleutel gedefinieerd als string met een maximale lengte van 40. Dit simpleType wordt gebruikt voor de door StUF gedefinieerde sleutelOntvangend, sleutelGegevensbeheer en sleutelVerzendend. Naast Sleutel is ook Sleutel-e gedefinieerd.
- Datum en tijdstip
  StUF0301 definieert in het simpleType Datum een datum als JJJJMMDD en in het simpleType Tijdstip een tijdstip als JJJJMMDDhhmmssSSS. Dit is niet in overeenstemming met de ISO-normen voor datum en tijdstip. StUF definieert ook Datum-e en Tijdstip-e. StUF kent daarnaast zogenaamde onvolledige datums voor het geval in een datum de dag, de maand of het jaar onbekend is. DatumMetIndicator en TijdstipMetIndicator zijn een extension op Datum-e en Tijdstip-e waarbij het attribute StUF:indOnvolledigeDatum wordt toegevoegd.
  In een volgende StUF-versie zullen de voorschriften van het stelsel van basisregistraties voor datum en tijd worden overgenomen. Hierop vooruitlopend is het verstandig om in sectormodellen alvast de door het stelsel voorgeschreven datum en tijdstip formaten te gebruiken.
- Groepsnaam
  Ten behoeve van de attributes elementnaam en groepsnaam binnen de metagegevens inOnderzoek, brondocument en gebeurtenis definieert StUF het simpleType groepsnaam.
- StatusMetagegeven
  Ten behoeve van het metagegeven inOnderzoek definieert StUF het simpleType StatusMetagegeven met als enige toegestane waarde 'J'. In een entiteittype wordt niet dit simpleType gebruikt maar een complexType met simpleContent waarbij aan StUF:StatusMetagegeven in elk geval zijn toegevoegd de attributes StUF:metagegeven en StUF:noValue en mogelijk de attributes elementnaam en groepsnaam. In een inOnderzoek element dient altijd xsi:nil="true" opgenomen te worden. In een vraagbericht hoort een inOnderzoek element zonder attributes te worden opgenomen.
- Gebeurtenis
  Ten behoeve van het metagegeven gebeurtenis definieert StUF het simpleType Gebeurtenis als een string met een maximale lengte van 200. In een entiteittype wordt niet dit simpleType gebruikt maar een complexType met simpleContent waarbij aan StUF:Gebeurtenis in elk geval zijn toegevoegd het attribute StUF:metagegeven en het

attribute StUF:tijdstip en mogelijk de attributes elementnaam en groepsnaam. In een vraagbericht hoort een gebeurtenis-element zonder attributes te worden opgenomen.

De voor een sectormodel te definiëren simpleTypes en “-e” complexTypes worden opgenomen in een apart schema met als naam de sectormodelcode in kleine letters gevolgd door de sectormodelversie gevolgd door “_simpleTypes.xsd”. Voor het sectormodel bg0310 is de naam dus bg0310_simpleTypes.xsd.

Voor een sectormodel worden in een schema met als target namespace StUF zonodig restrictions gedefinieerd op de door StUF gedefinieerde complexTypes met simpleContent voor StatusMetagegeven en Gebeurtenis. In deze restrictions kan gedefinieerd worden welke waarden voor de attributes groepsnaam en elementnaam zijn toegestaan. Dit schema krijgt als naam de sectormodelcode in kleine letters gevolgd door de sectormodelversie en gevolgd door “_simpleTypes_stuf.xsd”. Voor het sectormodel bg0310 is de naam dus bg0310_simpleTypes_stuf.xsd.

Indien een sectormodel restrictions nodig heeft op simpleTypes uit een ander sectormodel dan worden deze gedefinieerd in een schema met als naam de sectormodelcode in kleine letters gevolgd door de sectormodelversie gevolgd door “_”, gevolgd door de sectormodelcode en -versie van het te restricten sectormodel in kleine letters gevolgd door “_simpleTypes.xsd”. Voor restrictions op simpleTypes uit het sectormodel bg0310 binnen het sectormodel zkn0310 is de naam dus zkn0310_bg0310_simpleTypes.xsd.

De in deze paragraaf genoemde bestanden worden opgenomen in een folder "entiteiten" binnen een folder met als naam sss om het sectormodel aan te duiden. Dit is de sectormodelcode in kleine letters gevolgd door de sectormodelversie. Bijvoorbeeld, sss is een placeholder voor namen zoals bg0204, bg0310, zkn0310, etc.

2.3 Van objecttypen naar complexType's

Idealiter worden de objecttypen uit het informatiemodel één-op-één vertaald naar entiteiten binnen de berichten. In de praktijk leidt dit niet altijd tot optimale berichten en services. Het is bijvoorbeeld niet verstandig om subtypen van Natuurlijk persoon als ingezetene, niet-ingezetene, ingeschreven persoon en ander natuurlijk persoon ook in de berichten te onderkennen, omdat niet alle systemen die berichten verzenden en ontvangen deze subtypen hoeven te kennen. Ander voorbeeld, het maakt berichten een stuk simpeler als relaties naar objecttypen als Land of Woonplaats of Openbare ruimte niet expliciet worden gelegd. Deze relaties uit het informatiemodel kunnen in de berichten worden geïmplementeerd door gegevens van het gerelateerde object op te nemen in het object vanwaaruit de relatie ligt. We noemen dit soort elementen platgeslagen elementen. Dit is in bg0310 onder meer gedaan bij de implementatie van adressen. De gemaakte keuzen bij het vertalen van het informatiemodel naar een berichtenmodel dienen verantwoord te worden in een zogenaamd keuzenVerStUFfing-document voor een sectormodel.

Binnen een sectormodel kan m.b.t. historie op verschillende manieren omgegaan worden met platgeslagen elementen. Ze kunnen behandeld worden als normale gegevens maar er kunnen ook specifieke eisen gesteld worden aan de manier waarop er mee omgegaan wordt. Ook deze keuze dient vastgelegd te worden in het keuzenVerStUFfing-document.

Het informatiemodel is een model waarin de richting van relaties tussen objecttypen veelal geen rol speelt. Vanuit het ene objecttype kunnen via relaties langs verschillende paden andere objecttypen

worden bereikt. Vanuit een persoon kan je bijvoorbeeld rechtstreeks naar het objecttype adres gaan of via de ouder van die persoon. In het eerste geval krijg je het adres van de persoon zelf en in het tweede geval het adres van één van zijn ouders. Een berichtenmodel bestaat per entiteit uit een hiërarchische structuur. Bovenaan in de hiërarchie staat de entiteit zelf met de elementen voor de attributen van het objecttype. Daarnaast bevat de entiteit elementen voor relaties naar andere objecttypen. Een bericht met zo'n basisentiteit kan willekeurig groot worden. Je kan voor een persoon via de heeftAlsOuders relatie bijvoorbeeld teruggaan tot Adam en Eva. Het is een StUF best practice om bij het maken van een sectormodel allereerst basis-entiteiten als complexTypes te maken, waarin een relatie en zijn gerelateerde altijd ook basis-entiteiten zijn.

De StUF-standaard kent voor de identificatie van objecten in berichten het begrip kerngegevens. De kerngegevens zijn de gegevens waarmee een object in verreweg de meeste gevallen uniek geïdentificeerd kan worden, zelfs als niet alle kerngegevens beschikbaar zijn. Voor een natuurlijk persoon zijn de kerngegevens bijvoorbeeld het burgerservicenummer, de naamsgegevens, het adres, de geboortedatum en het geslacht. Het is een StUF best practice om naast de complexTypes voor de basisentiteiten ook een complexType voor de kerngegevens te definiëren.

Het basis- en kerngegevens-complexType worden opgenomen in een schema met als naam sss_ent_basis.xsd met sss de code voor het sectormodel.

Het complexType voor de kerngegevens is een restriction van het basis-complexType. Voor relaties wordt alleen een kerngegevens-complexType gedefinieerd, als er meer kerngegevens zijn dan de entiteit vanwaaruit de relatie ligt en de gerelateerde entiteit. Bij een huwelijk is dit bijvoorbeeld het geval, omdat een persoon meerdere keren met dezelfde partner kan trouwen. De datumSluiting is ook een kerngegeven.StUF definieert functionaliteit voor twee veel voorkomende interactiepatronen, het doorgeven van mutaties en het opvragen van gegevens. Het is een StUF best-practice om de entiteiten die binnen de functionaliteit voor mutaties en voor vraag/antwoord worden gebruikt te definiëren als restric-tions op de entiteiten in het sss_ent_basis-schema in twee schema's met als namen sss_ent_mutatie. en sss_ent_vraagAntwoord.xsd. Hieronder worden de regels hiervoor nader toegelicht.
Een samenvatting van de besproken complexTypes staat in onderstaande tabel. M staat hier voor de mnemonic voor een objecttype. Het kan hierbij ook gaan om een relatie. De mnemonic voor een objecttype heeft veelal als vorm XXX en de mnemonic voor een relatie XXXYYY of XXXYYYZZZ, als er meerdere relaties zijn tussen de objecttypen XXX en YYY.

Naam	**Functie**	**Schema**
M-basis	ComplexType gebruikt als basis voor de complexTypes in de berichtdefinities	sss_ent_basis.xsd
M-kerngegevens	ComplexType met de kerngegevens voor een objecttype	sss_ent_basis.xsd
M-kennisgeving	ComplexType gebruikt binnen een kennisgevingbericht	sss_ent_mutatie.xsd
M-kerngegevensKennisgeving	ComplexType gebruikt als gerelateerde binnen een kennisgevingbericht	sss_ent_mutatie.xsd
M-antwoord	ComplexType gebruikt binnen een antwoordbericht en het start element in een vraag-bericht	sss_ent_vraagAntwoord.xsd
M-historieFormeel	ComplexType voor het historieFormeel element	sss_ent_vraagAntwoord.xsd

Naam	Functie	Schema
M-historieFormeelRelatie	ComplexType voor het historieFormeelRelatie element binnen een relatie	sss_ent_vraagAntwoord.xsd
M-historieMaterieel	ComplexType voor het historieMaterieel element	sss_ent_vraagAntwoord.xsd
M-gerelateerdeAntwoord	ComplexType voor een gerelateerde in een antwoordbericht	sss_ent_vraagAntwoord.xsd
M-vraag	ComplexType voor het gelijk, vanaf, totEnMet en scope element in een vraagbericht	sss_ent_vraagAntwoord.xsd
M-vraagScope	ComplexType voor het scope-element in een vraagbericht, als dit een andere inhoud heeft dan het complexType voor het gelijk, vanaf en totEnMet element	sss_ent_vraagAntwoord.xsd
M-vraagSelectie	ComplexType voor het gelijk, vanaf en totEnMet element in een vraagbericht, als dit een andere inhoud heeft dan het complexType voor het scope-element	sss_ent_vraagAntwoord.xsd
M-gerelateerdeVraag M-gerelateerdeVraagScope M-gerelateerdeVraagSelectie	ComplexType's voor een gerelateerde in een vraagbericht.	sss_ent_vraagAntwoord.xsd

Het is best practice om in het mutatie-schema gerelateerden met alleen de kerngegevens te gebruiken en in het vraag/antwoord schema gerelateerden met de inhoud van een kennisgeving uit het mutatie-schema. De achtergrond voor de keuze voor kerngegevens in een gerelateerde in een mutatie is dat het voor een mutatie voldoende is om de gerelateerde te kunnen identificeren c.q. te kunnen vastleggen met gegevens die de gerelateerde uniek identificeren. In een latere mutatie kunnen de gegevens van een gerelateerde altijd nog aangevuld worden. Bij het opvragen van de gegevens van een objecttype wil je regelmatig meer weten dan alleen de kerngegevens van een gerelateerde. Om die reden wordt bij vraag/antwoord een gerelateerde met de inhoud van een kennisgeving gebruikt.

In het informatiemodel hebben veel relaties geen richting c.q. is een relatie slechts in een uitzonderingssituatie zonder zijn inverse relatie gedefinieerd. Bij het definiëren van de basis-entiteiten, de vraag/antwoord entiteiten en de kennisgeving-entiteiten staat de ontwerper van een sectormodel steeds voor de vraag of de relatie in beide richtingen of slechts in één richting wordt opgenomen. Het is een StUF best practice om geen verschil te hebben in de relaties in de basisentiteiten en vraag/antwoord-entiteiten. Het is een StUF best-practice om relaties in kennisgeving-entiteiten slechts vanuit één kant te onderhouden en wel vanuit de entiteit die het minst vaak al bestaat bij het leggen van de relatie. Bij het leggen van een relatie tussen een document en zaak wordt de relatie gelegd vanuit document: de zaak bestaat vaak al voor het document, omdat de meeste documenten in het kader van een zaak worden aangemaakt. Het onderhouden van een relatie vanuit één kant vergt minder implementatie-inspanning dan vanuit twee kanten.

Het is een StUF best-practice om de schema's zo scherp mogelijk te definiëren. In de kennisgeving-complexTypes is daarom bijvoorbeeld altijd het attribute StUF:verwerkingssoort verplicht en mag een attribute als StUF:scope niet voorkomen. In sss_ent_mutatie.xsd wordt om deze reden ook een kerngegevensKennisgeving-complexType gedefinieerd met alleen de kerngegevens voor gebruik als gerelateerde in het kennisgeving-complexType voor een relatie. Als een gerelateerde in een relatie

meer dan alleen de kerngegevens mag bevatten dan wordt hiervoor in sss_ent_mutatie.xsd een apart complexType gedefinieerd.

In sss_ent_vraagAntwoord.xsd worden alle complexTypes gedefinieerd die worden gebruikt in vraag/antwoordberichten. Ook hier worden de complexTypes steeds zo scherp mogelijk gedefinieerd. Het scope-attribute mag bijvoorbeeld alleen voorkomen in M-vraag complexTypes. De historie-complexTypes dienen gedefinieerd te worden op basis van de specificatie voor materiële en formele historie in het sectormodel.

De schema's sss_ent_basis.xsd, sss_ent_mutatie.xsd en sss_ent_vraagAntwoord.xsd vormen de *kern* van een sectormodel. De kern moet altijd aanwezig zijn omdat de hier gedefinieerde complexTypes ook de basis vormen van de complexTypes gebruikt binnen de update en vraag of selectie elementen in vrije berichten. Het is namelijk een StUF best-practice dat een update- of een vraag- of selectie-element in een vrij bericht altijd een restriction bevat op het kennisgeving-complexType c.q. het vraag- of antwoord-complexType uit sss_ent_mutatie.xsd c.q. sss_ent_vraagAntwoord.xsd. Deze drie schema's bevatten de vertaling van het informatiemodel naar de entiteiten gebruikt binnen de berichten. Zij specificeren ook de maximale grootte van een entiteit die in een bericht kan voorkomen voor de standaard door StUF gedefinieerde functionaliteit voor kennisgevingen en vraag/antwoord. Het zijn niet de enige drie schema's nodig voor het maken van een sectormodel.

In de volgende secties gaan we dieper in op de complexType's uit de bovenstaande tabel.

2.3.1 Basis- en kerngegevens

Het complexType met het suffix “-basis” definieert de maximale omvang van een antwoordbericht. De exacte inhoud ervan volgt uit de hieronder gegeven richtlijnen voor de overige typen, omdat deze allemaal van het “basis”-complexType worden afgeleid door middel van het restriction-mechanisme. Het “basis”-complexType bevat het attribute StUF:entiteittype met als waarde de mnemonic voor het entiteittype, maar dit attribute is niet verplicht, omdat het niet mag voorkomen binnen de complexTypes voor de historie. Daarnaast bevat het de StUF-attributes voor een StUF-entiteit. Het is aan de ontwerper van het sectormodel of de attributes StUF:sleutelOntvangend, StUF:sleutelGegevensbeheer en StUF:sleutelVerzendend al dan niet worden opgenomen. Binnen een “basis”-complexType zijn de complexTypes voor de gerelateerden van relaties en voor de historie-elementen altijd “basis”-complexTypes.

Het complexType met het suffix “-kerngegevens” definieert de kerngegevens van een StUF-entiteit. Het wordt gedefinieerd als een restriction op het “-basis” complexType met als elementen de kerngegevens en de kernrelaties. Alle elementen in een “-kerngegevens” complexType krijgen minOccurs=”0”, tenzij het attribute StUF:sleutelOntvangend niet voorkomt binnen het “basis-” complexType. In het “kerngegevens” complexType is het attribute StUF:entiteit verplicht.

De “basis”- en “kerngegevens”-complexTypes worden gedefinieerd in een apart schema. Dit schema include het schema met de simpleTypes voor het sectormodel en importeert zonodig de schema's met StUF-simpleTypes of simpleTypes uit andere sectormodellen. De geïmporteerde schema's bevatten zonodig de benodigde restrictions op de te importeren simpleTypes. Het schema met de “basis”- en kerngegevens-complexTypes krijgt als naam de sectormodelcode in kleine letters gevolgd door de sectormodelversie en gevolgd door “ent_basis.xsd” en wordt opgenomen in de folder 'entiteiten'. Voor het sectormodel bg0310 is de naam dus bg0310_ent_basis.xsd.Bij het importeren van een ander sectormodel ten behoeve van het definiëren van restrictions wordt altijd

het “basis”-schema gebruikt. Met andere woorden deze restrictions worden altijd gedefinieerd op de “basis”-complexTypes. Op deze manier wordt de import vanuit het andere sectormodel zo klein mogelijk gehouden.

2.3.2 Antwoordgegevens

Het “antwoord”-complexType bevat zonodig de door StUF voorgeschreven elementen historieMaterieel, historieFormeel en in relaties ook historieFormeelRelatie met de hiervoor gedefinieerde complexTypes. Het historieMaterieel-element kan nul of meer keren voorkomen en het historieFormeel- en historieFormeelRelatie-element nul of één keer. Voor de gerelateerden worden veelal specifiek voor een bepaald antwoordbericht gedefinieerde complexTypes gebruikt. Relaties die volgens het informatiemodel maar een eindig aantal keren kunnen voorkomen, maar waarvoor materiële historie is gedefinieerd, mogen in een antwoordbericht een onbeperkt aantal keren voorkomen (maxOccurs unbounded). De relatie van een persoon naar een verblijfsobject heeft in het informatiemodel bijvoorbeeld een kardinaliteit één, maar komt in een antwoordbericht meerdere keren voor, omdat een persoon in de loop van de tijd in verschillende verblijfsobjecten heeft gewoond. In de topfundamenteel, een relatie of een gerelateerde binnen een antwoordbericht mogen de attributes StUF:verwerkingssoort en StUF:scope niet voorkomen. Het attribute StUF:noValue mag bovendien niet voorkomen in het element voor de topfundamenteel en het element voor een gerelateerde in een antwoordbericht.

Voor de in het informatiemodel meervoudig voorkomende relaties dient binnen het antwoordbericht nog de sortering te worden gedefinieerd. Dit wordt gedaan door binnen het “antwoord”-complexType een annotation met appinfo op te nemen. Binnen het appinfo element wordt de sorteringRelatie gespecificeerd door binnen het element sortering één of meer elementen element op te nemen met de Xpath expression voor het element waarop gesorteerd dient te worden. Als er in aflopende volgorde gesorteerd dient te worden, dan wordt binnen <element> het attribute order=”DESC” opgenomen. Hieronder staat een voorbeeld voor deze annotation[1] voor de huwelijksrelatie van een persoon.

```
<annotation>
    <appinfo>
        <StUF:sorteringRelatie>
            <StUF:element>gerelateerde/geslachtsnaam</StUF:element>
            <StUF:element order="DESC">datumSluiting</StUF:element>
        </StUF:sorteringRelatie>
    </appinfo>
</annotation>
```

De inhoud van deze appinfo is in het StUF-schema gedefinieerd als het element sorteringRelatie.

De “antwoord”-complexTypes worden gedefinieerd in een schema met als naam sss_ent_vraagAntwoord.xsd dat een include doet van het schema sss_ent_basis.xsd. Het bestand sss_ent_vraagAntwoord.xsd wordt opgenomen in de folder 'vraagAntwoord' voor de vraag/antwoordberichtcatalogus binnen de folder sss.

1 Mocht bij het genereren van software op basis van een wsdl deze annotation met appinfo problemen op leveren, dan kan de annotation uit het schema verwijderd worden.

2.3.3 Historische gegevens

Het “historieMaterieel”- en “historieFormeel”-complexType bevatten de elementen van een entiteittype waarvoor in het informatiemodel materiële respectievelijk formele historie is gedefinieerd. Als er voor een entiteittype geen historie is gedefinieerd dan komen deze complexTypes niet voor.

Het “historieMaterieel” complexType bevat altijd het element StUF:tijdvakGeldigheid met minOccurs=”1” (verplicht) en als ook formele historie is gedefinieerd het element StUF:tijdstipRegistratie met minOccurs=”0” (optioneel).

Het “historieFormeel” complexType bevat altijd de elementen StUF:tijdvakGeldigheid en StUF:tijdstipRegistratie met minOccurs=”1” (verplicht).

Het “historieMaterieel”- en “historieFormeel”-complexType voor een entiteittype waarvoor formele historie is gedefinieerd bevatten een optioneel historieFormeel-element. Het “historieFormeelRelatie”-complexType bevat altijd een optioneel “historieFormeel”- en “historieFormeelRelatie”-element. Het “historieMaterieel”- en “historieFormeel”-complexType voor een relatie bevatten geen gerelateerde. Het “historieFormeelRelatie”-complexType voor een relatie bevat dezelfde elementen als het “historieFormeel”-complexType plus een verplichte gerelateerde met het “kerngegevens”-complexType. Alle drie de “historie”-complexTypes bevatten geen StUF-attributes.

De “historie”-complexTypes worden gedefinieerd in een schema met als naam sss_ent_vraagAntwoord.xsd dat een include doet van het schema sss_ent_basis.xsd.

2.3.4 Gegevens in kennisgevingen

Het “kennisgeving”-complexType bevat geen historie-elementen. Als sleutelOntvangend wordt ondersteund, zijn alle elementen in een “kennisgeving”-complexType optioneel. Als sleutelOntvangend niet wordt ondersteund, dan zijn de kerngegevens verplicht in een kennisgeving. Relaties komen erin voor met de kardinaliteit voorgeschreven door het informatiemodel. Alleen de relaties die vanuit de topfundamenteel worden onderhouden worden opgenomen in het “kennisgeving”-complexType. In een “kennisgeving”-complexType waarvoor historieMaterieel is gedefinieerd moet het element StUF:tijdvakGeldigheid als niet verplicht element worden opgenomen. In een “kennisgeving”-complexType waarvoor historieFormeel is gedefinieerd moet StUF:tijdstipRegistratie als niet verplicht element worden opgenomen. In de topfundamenteel, de relaties en de gerelateerde zijn de attributes StUF:entiteittype en StUF:verwerkingssoort verplicht en mag het attribute StUF:scope niet voorkomen. Alleen binnen relaties mag het attribute StUF:noValue voorkomen. Voor gerelateerden wordt meestal het “kerngegevensKennisgeving”-complexType gebruikt, tenzij er van de gerelateerde meer gegevens dan alleen de kerngegevens mogen worden toegevoegd of gewijzigd. Er worden normaal gesproken geen aparte typen gedefinieerd voor een toevoegkennisgeving (mutatiesoort “T”), een wijzig- of correctiekennisgeving (mutatiesoort “C”, “F” of “W”) en een verwijderkennisgeving (mutatiesoort “V”).

Het “kerngegevensKennisgeving”-complexType is een restriction op het “basis”-complexType en bevat uitsluitend de kerngegevens. Als het attribute StUF:sleutelOntvangend niet wordt ondersteund, dan zijn alle elementen in het “kerngegevensKennisgeving”-complexType verplicht,

anders optioneel. Verder gelden dezelfde regels als voor het “kennisgeving”-complexType.

De “kennisgeving”- en “kerngegevensKennisgeving”-complexTypes worden gedefinieerd in een schema met als naam sss_ent_mutatie.xsd dat een include doet van het schema sss_ent_basis.xsd. Omdat deze complexTypes nodig zijn voor het definiëren van kennisgevingberichten wordt dit bestand opgenomen in de folder 'kennisgeving' voor de kennisgevingberichtcatalogus binnen de folder sss.

2.3.5 Gegevens in vraagberichten

Het “vraag”-complexType bevat geen historie-elementen en het attribute StUF:verwerkingssoort mag er niet in voorkomen. In een vraag-“complexType” mogen de elementen voor attributen en relaties maximaal één keer voorkomen. Alleen in de topfundamenteel van een “vraag”-complexType mag het attribute StUF:scope voorkomen. Voor een supertype zijn afzonderlijke “vraagScope”- en “vraagSelectie” complexTypes nodig. Het “vraagSelectie”-complexType bevat de elementen van het supertype en het “vraagScope”-complexType bevat alle subtypen van het supertype, omdat voor elk subtype gedefinieerd moet kunnen worden welke elementen je vraagt. Als een supertype als gerelateerde voorkomt in een entiteittype, dan zijn voor dat entiteittype ook afzonderlijke “vraagScope”- en “vraagSelectie”-complexTypes nodig. Hetzelfde geldt voor entiteittypen waarbinnen als gerelateerde een entiteittype voorkomt met afzonderlijke “vraagScope”- en “vraagSelectie”-complexTypes.

De “vraag”-, “vraagScope”- en “vraagSelectie”-complexTypes worden gedefinieerd in een schema met als naam sss_ent_vraagAntwoord.xsd dat een include doet van het schema sss_ent_basis.xsd. Dit bestand opgenomen in de folder 'vraagAntwoord' voor het vraag/antwoordberichtcatalogus binnen de folder sss.

2.3.6 Hergebruik in andere sectormodellen

In de praktijk komt het geregeld voor dat van een persoon in een ander sectormodel meer of minder gegevens moeten worden vastgelegd dan beschikbaar zijn in het sectormodel bg0310. Allereerst wordt dan in de namespace van het oorspronkelijke entiteittype als restriction een complexType aangemaakt dat precies de gewenste elementen bevat. Geef dit complexType als naam zijn mnemonic XXX gevolgd door een streepje en de code voor het sectormodel waarbinnen dit complexType gebruikt gaat worden in kleine letters en dat weer gevolgd door “-basis”. Zo'n complexType voor NPS binnen het sectormodel Zaken heet bijvoorbeeld NPS-zkn-basis.
Als in het andere sectormodel meer gegevens nodig zijn dan de restricted complexType biedt, dan wordt in het basisschema voor het sectormodel een complexType YYY-basis aangemaakt dat als eerste element een relatie isEen zonder elementen bevat met als gerelateerde het zojuist gedefinieerde XXX-sss-basis complexType. De mnemonic YYY mag gelijk zijn aan de mnemonic XXX als het niet gaat om een subtype, maar alleen om het toevoegen en verwijderen van elementen uit XXX. Voeg aan dit complexType YYY-basis alle elementen en relaties toe die extra benodigd zijn. Op YYY-basis kunnen vervolgens op de normale manier weer restrictions gedefinieerd worden.

Er is een RFC op de StUF-standaard ingediend om de isEen constructie in de standaard op te nemen, zodat er geen relatie meer gebruikt hoeft te worden, maar aan het isEen element zelf al het type XXX-sss-basis kan worden toegekend.

2.4 Beperking uitbreidbaarheid

In alle hierboven gedefinieerde complexTypes en in alle complexTypes die als restriction daarop gedefinieerd worden, dient te worden opgenomen final=”extension” om expliciet te specificeren dat er van deze complexTypes geen extension gedefinieerd mag worden. Deze specificatie is overigens niet waterdicht, want als je eerst een restriction definieert en vervolgens op die restriction weer een extension, dan wordt dat door XML Schema toegestaan.

Hiermee is de vertaling van het domeinmodel naar StUF-entiteittypen en complexTypes beschreven en hebben we de bouwstenen voor het definiëren van berichten en services.

3 Van entiteitschema naar berichten en services

De StUF-standaard beschrijft gedetailleerd de functionaliteit voor het doorgeven van mutaties met behulp van Lk-berichten, het synchroniseren van databases met de Sa- en Sh-berichten en het bevragen van gegevens met de Lv01-/La01- t/m Lv10-/La10-berichten. Vanuit het hierboven gedefinieerde sss_ent_basis.xsd kunnen de berichten en services voor deze functionaliteit gegenereerd worden voor de protocolbinding gebaseerd op wsdl, soap en http[2]. Ook voor andere protocolbindingen kunnen services worden gedefinieerd. In de praktijk is dit alleen nog gedaan binnen het sectormodel WOZ voor de Digikoppeling protocolbinding.

Vanuit het oogpunt van beheer is van belang dat de aldus gegenereerde schema's en wsdl's meer berichten en services bevatten dan systemen in de praktijk implementeren. De schema's met de gegenereerde berichten zijn dus catalogi waaruit berichten geselecteerd kunnen worden voor een bepaalde protocolbinding. Het is een best practice om in een sectormodel voor elk asynchroon bericht ook de synchrone variant op te nemen. Dit maakt het mogelijk om een centrale berichtenbuffer in te zetten zodat niet elke applicatie een eigen berichtenbuffer hoeft te implementeren voor het afhandelen van asynchrone berichten. Het omgekeerde geldt natuurlijk niet in alle gevallen: voor een synchroon bericht hoeft niet automatisch een asynchroon bericht te worden opgenomen. In een sectormodel moet je daarnaast aangeven welk type berichten de default zijn, asynchrone of synchrone berichten. In een koppelvlak mag daar overigens weer van afgeweken worden.
De gegenereerde wsdl's behorend bij de berichtcatalogi zijn voorbeeld-wsdl's.

Er worden voor het doorgeven van mutaties, het synchroniseren en het bevragen twee berichtcatalogi onderscheiden:

1. De mutatiecatalogus met de berichten voor het doorgeven van mutaties door middel van enkelvoudige kennisgevingen en het synchroniseren van databases.[3]
2. De vraagAntwoord-catalogus met de berichten voor het bevragen van databases.

Deze twee standaard berichtcatalogi behoren tot het vaste deel van het sectormodel en mogen niet worden aangepast zonder versiewijziging van het sectormodel; ze worden beschreven in secties 3.1 en 3.2. In sectie 3.3 worden de overige berichtcatalogi geintroduceerd. Deze kunnen worden toegevoegd zonder versiewijziging van het sectormodel. In sectie 3.4 wordt beschreven hoe voorbeeld-wsdl's kunnen worden toegevoegd aan een berichtcatalogus. In sectie 3.5 wordt geadviseerd om een hulpschema te gebruiken om alle berichtschema's binnen een sectormodel in

2 Zie het document stuf.bindingen.030200.pdf voor een beschrijving van deze protocolbinding.
3 De mutatiecatalogus bevat geen samengestelde kennisgevingen (berichtsoorten Lk03 en Lk04).

één keer te valideren.

3.1 Mutatieberichtcatalogus

Mutaties worden doorgegeven met behulp van kennisgeving- en synchronisatieberichten. Omdat samengestelde kennisgevingen veelal gebaseerd zijn op specifieke gebeurtenissen en doorgaans niet automatisch gegenereerd kunnen worden uit de kern schema's maken deze geen onderdeel uit van de mutatiecatalogus. Samengestelde kennisgevingen kunnen worden opgenomen in de overige berichtcatalogi naast de hierboven genoemde twee kerncatalogi. De volgende berichtsoorten uit de StUF-standaard zijn opgenomen in de mutatiecatalogus:

- enkelvoudige kennisgeving zonder toekomstmutatie (asynchroon: Lk01 en synchroon: Lk02) en desgewenst enkelvoudige kennisgeving met toekomstmutatie[4] (asynchroon: Lk05 en synchroon: Lk06).
- Synchronisatieberichten Sa01 t/m Sa04 en Sh01 t/m Sh04.

3.1.1 Enkelvoudige kennisgevingen

Enkelvoudige kennisgevingen zonder en met toekomstmutatie krijgen als elementnaam de mnemonic van het entiteittype waarover de kennisgeving gaat in kleine letters gevolgd door de berichtcode voor de kennisgeving. Een asynchrone enkelvoudige kennisgeving voor een natuurlijk persoon krijgt dus als elementnaam npsLk01 en een synchrone toekomstkennisgeving voor een adres aoaLk06.

Omdat de kennisgevingberichten ook hergebruikt kunnen worden binnen vrije berichten, wordt de kennisgeving eerst gedefinieerd als een complexType XXX-kennisgeving in sss_msg_mutatie.xsd en vervolgens worden de elementen gedefinieerd zoals bijvoorbeeld het hierboven beschreven element npsLk01.

Een enkelvoudige kennisgeving bevat drie elementen:

- stuurgegevens
- parameters
- object (één of twee keer)

Het complexType voor het stuurgegevens element wordt gedefinieerd als restriction op het StUF:Stuurgegevens-Lk0n complexType met als waarde voor entiteittype de mnemonic van het object waarover de mutatie gaat. Het complexType voor het parameters element wordt gebruikt uit het StUF-schema of als restriction gedefinieerd op ParametersKennisgeving uit het StUF-schema. Deze restrictions worden gedefinieerd in een schema met als naam sss_msg_stuf_mutatie.xsd dat een include doet van het schema sss_simpleTypes_stuf.xsd. Ook dit bestand wordt opgenomen in de folder 'mutatie' binnen de folder sss.

Voor het object element wordt het complexType gebruikt dat in sectie 2.3 als M-kennisgeving is beschreven. De kennisgevingberichten worden gedefinieerd in een schema met als naam sss_msg_mutatie.xsd. Dit schema doet een include van sss_ent_mutatie.xsd en een import van sss_msg_stuf_mutatie.xsd. Het schema sss_msg_stuf_mutatie.xsd doet een include van het schema sss_simpleTypes_stuf.xsd. De bestanden sss_msg_mutatie.xsd en sss_msg_stuf_mutatie.xsd vormen te samen de berichtdefinities voor de berichtcatalogus voor het doorgeven van mutaties voor het sectormodel sss en worden opgenomen in de folder 'mutatie' in de folder sss voor het sectormodel.

4 Mutatie waarbij de ingangsdatum van de mutatie in de toekomst ligt.

3.1.2 Synchronisatieberichten

Synchronisatieberichten worden tevens gedefinieerd in de schema's sss_msg_stuf_mutatie.xsd en sss_msg_mutatie.xsd in de folder 'mutatie'. Het bestand sss_stuf_mutatie.xsd bevat de restrictions op de StUF:Stuurgegevens complexTypes voor synchronisatieberichten. Het element entiteittype krijgt als waarde de mnemonic van het entiteitttype dat wordt gesynchroniseeerd.

Een synchronisatiebericht krijgt als naam de mnemonic voor het entiteittype in kleine letters gevolgd door de berichtcode voor het bericht, bijvoorbeeld npsSh01 voor het asynchrone synchronisatie-historisch bericht voor een natuurlijk persoon.

Een vraag-om-synchronisatie bericht (Sa03, Sa04, Sh03 en Sh04) bevat na het stuurgegevens element de kerngegevens van het object waarom gevraagd wordt. Het hiervoor benodigde complexType XXX-kerngegevens is gedefinieerd in het schema sss_ent_basis.xsd.

Een synchronisatie-actueel bericht (Sa01 en Sa02) bevat na het stuurgegevens element een toevoegkennisgeving. Voor deze toevoegkennisgeving wordt in het schema sss_msg_mutatie.xsd een complexType XXX-Lk0nT gedefinieerd met XXX de mnemonic voor het objecttype. Daarnaast wordt ook voor het synchronisatie-actueel bericht zelf een complexType XXX-Sa0n gedefinieerd, omdat dit bericht ook voorkomt in een synchronisatie-historisch bericht. Het berichtelement xxxSa0n wordt gedefinieerd met als complexType XXX-Sa0n.

Een synchronisatie-historisch bericht (Sh01 en Sh02) wordt gedefinieerd voor alle objecttypen waarvoor historie is gedefinieerd. Het bevat na de stuurgegevens een element actueel met een synchronisatie-actueel bericht gevolgd door een historie element met daarin eerst een element oudste met een toevoegkennisgeving voor de oudste situatie gevolgd door 0 of meer wijzigkennisgevingen voor de opbouw van de historie tot en met de laatste situatie. Voor deze wijzigingkennisgevingen wordt in het schema sss_msg_mutatie.xsd een complexType XXX-Lk0nW gedefinieerd.

3.2 Vraag/antwoordberichtcatalogus

De vraag/antwoordberichtcatalogus wordt gedefinieerd in de schema's sss_msg_vraagAntwoord.xsd en sss_msg_stuf_vraagAntwoord.xsd. sss_msg_stuf_vraagAntwoord.xsd bevat de restrictions op de StUF:Stuurgegevens complexTypes voor vraag- en antwoordberichten en de restrictions op de StUF:ParametersVraag complexTypes. In de restrictions op de StUF:Stuurgegevens complexType krijgt het element entiteittype weer als waarde de mnemonic van het entiteittype dat wordt gesynchroniseeerd.

In de restrictions op de StUF:ParametersVraag complexTypes worden de sorteringen voor een entiteittype gespecificeerd. Voor elk fundamenteel entiteitttype wordt een simpleType XXX-sortering gedefinieerd als restriction op StUF:Sortering met XXX de mnemonic voor het entiteittype. Hieronder worden nog nadere voorschriften gegeven voor de definitie van het simpleType XXX-sortering.
Er worden maximaal zes restrictions op StUF:ParametersVraag gedefinieerd:

1. ParameterVraagSynchroon waarin de elementen peiltijdstipMaterieel en peiltijdstipFormeel worden weggelaten en de elementen sortering en vervolgvraag verplicht zijn. Het element sortering krijgt als type XXX-sortering. XXX-ParametersVraagSynchroon wordt gebruikt in de Lv01, Lv07, Lv09, Lv11 en Lv13-vraagberichten.

2. ParametersVraagAsynchroon waarin de elementen indicatorAantal, peiltijdstipMaterieel en peiltijdstipFormeel worden weggelaten en de elementen sortering en vervolgvraag verplicht zijn. Het element sortering krijgt als type XXX-sortering. XXX-ParametersVraagAsynchroon wordt gebruikt in de Lv02, Lv08, Lv10, Lv12 en Lv14-vraagberichten.
3. ParametersVraagSynchroonMaterieel waarin het element peiltijdstipFormeel worden weggelaten en de elementen sortering en vervolgvraag verplicht zijn. Het element sortering krijgt als type XXX-sortering. XXX-ParametersVraagSynchroonMaterieel wordt gebruikt in het Lv03-vraagbericht.
4. ParametersVraagAsynchroonMaterieel waarin de elementen indicatorAantal en peiltijdstipFormeel worden weggelaten en de elementen sortering en vervolgvraag verplicht zijn. Het element sortering krijgt als type XXX-sortering. XXX-ParametersVraagAsynchroonMaterieel wordt gebruikt in het Lv04-vraagbericht.
5. ParametersVraagSynchroonFormeel waarin de elementen sortering en vervolgvraag verplicht zijn. Het element sortering krijgt als type XXX-sortering. XXX-ParametersVraagSynchroonFormeel wordt gebruikt in het Lv05-vraagbericht.
6. ParametersVraagAsynchroonFormeel waarin het element indicatorAantal wordt weggelaten en de elementen sortering en vervolgvraag verplicht zijn. Het element sortering krijgt als type XXX-sortering. XXX-ParametersVraagAsynchroonFormeel wordt gebruikt in het Lv06-vraagbericht.

De restrictions ParametersVraagSynchroonMaterieel en ParametersVraagAsynchroonMaterieel worden alleen gedefinieerd als er materiële historie is gedefinieerd voor het entiteittype. De restrictions ParametersVraagSynchroonFormeel en ParametersVraagAsynchroonFormeel worden alleen gedefinieerd als er formele historie is gedefinieerd voor het entiteittype.

In de restriction XXX-Sortering op StUF:Sortering wordt via het facet `<maxInclusive value="nn"/>` het aantal sorteringen voor het entiteittype XXX gezet. Daarnaast worden in een `<annotation>` van dit simpleType als `<appinfo>` de sorteringen gespecificeerd[5]. `<appinfo>` bevat één of meer elementen `<sorteringObject>`. Het element `<sorteringObject>` bevat allereerst het element `<nummer>` met het nummer van de sortering gevolgd door één of meer `<element>` elementen met de elementen uit de sortering. Een sorteringselement wordt gespecificeerd als een Xpath expression. Wanneer op een element Descending gesorteerd wordt, dan wordt op het `<element>` element een attribute order="DESC" toegevoegd. De sorteringen voor NPS worden bijvoorbeeld als volgt geannoteerd:

```
<annotation>
   <appinfo>
      <StUF:sorteringObject>
         <StUF:nummer>1</StUF:nummer>
         <StUF:element>geslachtsnaam</StUF:element>
         <StUF:element>voorletters</StUF:element>
         <StUF:element>voorvoegselGeslachtsnaam</StUF:element>
      </StUF:sorteringObject>
      <StUF:sorteringObject>
         <StUF:nummer>2</StUF:nummer>
         <StUF:element>verblijfsadres/aoa.identificatie</StUF:element>
      </StUF:sorteringObject>
      <StUF:sorteringObject>
```

5 Mocht bij het genereren van software op basis van een wsdl deze annotation met appinfo problemen op leveren, dan kan de annotation uit het schema verwijderd worden.

```
    <StUF:nummer>3</StUF:nummer>
    <StUF:element>verblijfsadres/aoa.postcode</StUF:element>
    <StUF:element>verblijfsadres/aoa.huisnummer</StUF:element>
    <StUF:element>verblijfsadres/aoa.huisletter</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>4</StUF:nummer>
    <StUF:element>verblijfsadres/gor.straatnaam</StUF:element>
    <StUF:element>verblijfsadres/aoa.huisnummer</StUF:element>
    <StUF:element>verblijfsadres/aoa.huisletter</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>5</StUF:nummer>
    <StUF:element>sub.verblijfBuitenland/lnd.landcode</StUF:element>
    <StUF:element>sub.verblijfBuitenland/sub.adresBuitenland3</StUF:element>
    <StUF:element>sub.verblijfBuitenland/sub.adresBuitenland2</StUF:element>
    <StUF:element>sub.verblijfBuitenland/sub.adresBuitenland1</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>6</StUF:nummer>
    <StUF:element>sub.verblijfBuitenland/lnd.landnaam</StUF:element>
    <StUF:element>sub.verblijfBuitenland/sub.adresBuitenland3</StUF:element>
    <StUF:element>sub.verblijfBuitenland/sub.adresBuitenland2</StUF:element>
    <StUF:element>sub.verblijfBuitenland/sub.adresBuitenland1</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>7</StUF:nummer>
    <StUF:element>geboortedatum</StUF:element>
    <StUF:element>geslachtsnaam</StUF:element>
    <StUF:element>voorletters</StUF:element>
    <StUF:element>voorvoegselGeslachtsnaam</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>8</StUF:nummer>
    <StUF:element>inp.bsn</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>9</StUF:nummer>
    <StUF:element>inp.a-nummer</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>10</StUF:nummer>
    <StUF:element>sub.rekeningnummerBankGiro</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>11</StUF:nummer>
    <StUF:element>sub.telefoonnummer</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>12</StUF:nummer>
    <StUF:element>sub.emailadres</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>13</StUF:nummer>
    <StUF:element>rps.isEigenaarVan/gerelateerde/kvkNummer</StUF:element>
</StUF:sorteringObject>
<StUF:sorteringObject>
    <StUF:nummer>14</StUF:nummer>
```

```
            <StUF:element>anp.identificatie</StUF:element>
        </StUF:sorteringObject>
    </appinfo>
</annotation>
```

De inhoud van deze appinfo is in het StUF-schema gedefinieerd als het element sorteringObject.

Omdat voor een entiteittype meerdere vraag-berichten worden gedefinieerd met allemaal dezelfde inhoud na het parameters-element wordt deze inhoud gedefinieerd als een group met als naam xxxVraagBody met xxx de mnemonic in kleine letters voor het entiteittype. xxxVraagBody bevat een sequence met de elementen gelijk, vanaf, totEnMet, scope en start. De elementen gelijk, vanaf en totEnMet krijgen als complexType XXX-vraagSelectie of als die niet gedefinieerd hoeft te worden XXX-vraag. Het element object binnen scope krijgt als type XXX-vraagScope of als die niet gedefinieerd hoeft te worden XXX-vraag. Het element object binnen start krijgt als type XXX-antwoord. De xxxVraagBody groups worden gedefinieerd in het sss_msg_vraagAntwoord.xsd schema. In dit schema worden ook de berichtelementen voor de vraagberichten gedefinieerd.

Een vraagbericht krijgt als naam de mnemonic voor het entiteittype in kleine letters gevolgd door de berichtcode voor het bericht, wanneer het “vraag”- of het “vraagScope”- en het “vraagSelectie”-complexType gebruikt wordt. Het synchrone vraagbericht zonder historie voor een natuurlijk persoon krijgt bijvoorbeeld als naam npsLv01. Een antwoordbericht krijgt als naam de mnemonic voor het entiteittype in kleine letters gevolgd door de berichtcode voor het bericht, wanneer het “antwoord”-complexType gebruikt wordt. Het synchrone antwoordbericht zonder historie voor een natuurlijk persoon krijgt bijvoorbeeld als naam npsLa01. De vraagbericht-elementen worden gedefinieerd in het schema sss_msg_vraagAntwoord.xsd met behulp van de in sss_ent_stuf_vraagAntwoord.xsd gedefinieerde complexTypes en de in sss_msg_vraagAntwoord.xsd zelf gedefinieerd groups. De antwoordbericht-elementen worden ook gedefinieerd in het schema sss_msg_vraagAntwoord.xsd met behulp van de in sss_ent_stuf_vraagAntwoord.xsd gedefinieerde complexTypes voor de stuurgegevens. Het object-element binnen het element antwoord krijgt als type XXX-antwoord.

Het schema sss_msg_stuf_vraagAntwoord.xsd doet een include van het schema sss_simpleTypes_stuf.xsd of als dat niet is gedefineerd van stuf0301.xsd. Het schema sss_msg_vraagAntwoord.xsd doet een include van het schema sss_ent_vraagAntwoord.xsd en een import van het schema sss_msg_stuf_vraagAntwoord.xsd.

Het sss_msg_vraagAntwoord.xsd schema definieert de berichtcatalogus voor het opvragen van objecten voor het sectormodel sss. De bestanden die deze berichtcatalogus definiëren, worden opgenomen in de folder 'vraagAntwoord' in de folder sss voor de namespace.

3.3 Overige berichtcatalogi

In de vorige secties werden de twee kerncatalogi (mutatie en vraagAntwoord) van een sectormodel beschreven. Een sectormodel zal veelal deze twee standaard berichtcatalogi bevatten maar een sectormodel kan nog andere berichtcatalogi bevatten als het gebruik wil maken van andere mogelijkheden die de StUF-standaard te bieden heeft. Met de mutatie- en vraagAntwoord-catalogus benut een sectormodel slechts een klein deel van de mogelijkheden die de StUF-standaard biedt. Er kunnen ook samengestelde kennisgevingen worden gedefinieerd of vrije berichten.

In een samengestelde kennisgeving worden twee of meer kennisgevingen samengevoegd die

als één transactie verwerkt moeten worden. Een samengestelde kennisgeving krijgt als elementnaam de berichtcode in kleine letters gevolgd door een "-" en de functie uit de stuurgegevens in camel case (de eerste letter een kleine letter, alle spaties verwijderd en elke letter na een spatie een hoofdletter).

De exacte inhoud van samengestelde kennisgevingen kan scherper worden gedefinieerd door voor de enkelvoudige kennisgevingen eigen complexTypes te definiëren. Het is een StUF best-practice om alleen samengestelde kennisgevingen te definiëren, als de eventuele foutafhandeling bij verwerking in de database de som is van de foutafhandeling voor de enkelvoudige kennisgevingen erbinnen. Als er meer foutafhandeling vereist is, dan dient een vrij bericht gedefinieerd te worden met daarbinnen twee of meer elementen met de update-functie.

In een vrij bericht kan de functionaliteit van een enkelvoudige kennisgeving in een element met als functie update worden opgenomen of de functionaliteit van een vraagbericht in een element met als functie vraag. Voor een vrij bericht is altijd een beschrijving van de gewenste functionaliteit nodig.

Nieuwe berichten kunnen aan een bestaand sectormodel sss zonder consequenties worden toegevoegd als ze worden ondergebracht in een bestaande berichtcatalogus die niet een standaard catalogus is of in een nieuwe berichtcatalogus. In het laatste geval wordt er een nieuwe folder bbb (de naam van de nieuwe berichtcatalogus) aangemaakt binnen de folder sss (code van het bestaande sectormodel). Wanneer de nieuwe berichten aan de kerncatalogi worden toegevoegd dan heeft dat wel gevolgen, namelijk een wijziging van de versie en de namespace van het sectormodel. Het is mogelijk dat een berichtencatalogus slechts één bericht bevat.

Er zijn twee redenen om extra berichten en services te definiëren naast de berichten uit de mutatie-, vraagAntwoord- of een andere catalogus:

1. Restrictie: een systeem wil expliciet aangeven dat het strengere eisen stelt aan binnenkomende berichten dan het bericht uit de berichtcatalogus.
2. Extra functionaliteit: een systeem wil berichten met nieuwe functionaliteit definiëren.

Van restrictie is sprake als een bericht gedefinieerd kan worden als een restriction op het complexType voor een bericht in de mutatie-, vraagAntwoord- of een andere berichtcatalogus. Bij restrictie mag geen extra foutafhandeling worden gedefinieerd bovenop de standaard verwerking. Als het binnenkomende bericht valide is, dan dient het net zoals elke ander bericht van dat type verwerkt te kunnen worden. Als er toch extra eisen gesteld worden, dan dient een vrij bericht gedefinieerd te worden. Deze eis wordt gesteld om te waarborgen dat elke enkelvoudige kennisgeving en elk vraag/antwoordbericht met dezelfde functionaliteit verwerkt kan worden. In alle andere gevallen is er sprake van extra functionaliteit.
Een restrictie kan op twee manieren worden gedefiniëerd:

1. De restrictie wordt in een separaat document beschreven, maar niet afgedwongen via een schema. Voor deze aanpak is gekozen bij het definiëren van het BAG/GBA koppelvlak binnen het sectormodel bg0204.
2. De restrictie wordt via een schema afgedwongen.

Als de restrictie via een schema wordt afgedwongen, wordt een nieuw berichtelement gedefinieerd gebaseerd op een complexType dat een restriction is van een bericht-complexType uit een andere berichtcatalogus. Restrictions op entiteiten uit andere berichtcatalogi worden ondergebracht in een schema met als naam sss_ent_bbb.xsd en de berichtelementen en de eventuele bericht-complexTypes van een berichtcatalogus worden ondergebracht in een schema met als naam sss_msg_bbb.xsd met bbb een aanduiding van de berichtcatalogus. Eventuele wsdl's volgen ook

deze naamgevingssystematiek. Het is een StUF best-practice om restricties via een schema af te dwingen.

Indien nodig wordt het schema sss_ent_bbb.xsd gecreëerd waarin restrictions op complexTypes uit andere berichtcatalogus zijn opgenomen. Voor entiteit-complexTypes binnen een samengestelde kennisgeving of een element met als functie update in een vrij bericht geldt als extra eis dat deze altijd een restriction moeten zijn van het kennisgeving-complexType voor die entiteit in de mutatie-berichtcatalogus. Voor entiteit-complexTypes binnen een element met als functie vraag in een vrij bericht geldt als extra eis dat deze altijd een restriction moeten zijn van de vraag-complexTypes voor die entiteit (gelijk, vanaf, totEnMet en scope-element) of het antwoord-complexType (start-element). Ook deze complexTypes worden ondergebracht in het schema sss_ent_bbb.xsd.

Deze voorschriften waarborgen dat een complexType voor een entiteit altijd kleiner is dan een basis-complexType en dat in samengestelde kennisgevingen en in het update element van vrije berichten nooit een entiteit zit die niet via de standaard verwerking voor een kennisgeving kan worden verwerkt. Een element met functie vraag in een vrij bericht kan ook altijd met de standaard verwerking voor een vraagbericht verwerkt worden. Deze waarborgen maken hergebruik van functionaliteit mogelijk in software die StUF-berichten verwerkt.

Naast het entiteitschema schema sss_ent_bbb.xsd kan de overige berichtcatalogus bbb analoog aan de standaard berichtcatalogi de berichtschema's sss_msg_stuf_bbb.xsd en sss_msg_bbb.xsd bevatten met als enige verschil dat naast vraagAntwoord, enkelvoudige kennisgeving- en synchronisatieberichten hierin ook definities voor samengestelde kennisgevingberichten en vrije berichten kunnen voorkomen.

3.4 Services

Zoals al eerder gezegd dienen in een sectormodel bij elke berichtcatalogus een of meer voorbeeld wsdl's te worden geleverd. Het definiëren van services is beschreven in document stuf.bindingen.030200.pdf. Daar worden binnen een wsdl de volgende `<portType>` elementen voorgeschreven voor het definiëren van services op basis van de diverse berichttypen van StUF :

- OntvangAsynchroon
  Deze service dient alle ondersteunde asynchrone berichten te kunnen ontvangen.
- BeantwoordVraag
  Deze service dient alle ondersteunde synchrone vraagberichten te kunnen verwerken
- VerwerkSynchroneKennisgeving
  Deze service dient alle ondersteunde synchrone kennisgeving- en synchronisatieberichten te kunnen verwerken.
- VerstrekSynchronisatieBericht
  Deze service dient alle ondersteunde synchrone vraag-om-synchronisatieberichten te kunnen verwerken.
- VerwerkSynchroonVrijBericht
  Deze service dient alle ondersteunde synchrone vrije berichten te kunnen verwerken.
- VerwerkTriggerbericht
  Deze service dient het triggerbericht te kunnen verwerken, indien dit wordt ondersteund.

Het staat een systeem vrij om naast deze services ook nog andere services voor StUF-berichten te ondersteunen. Hiervoor dient het systeem dan zijn eigen wsdl te definiëren.

De wsdl's voor de bovengenoemde services worden per berichtcatalogus gemaakt. De wsdl's voor ontvangAsynchroon krijgen als naam sss_ontvangAsynchroon_bbb.xsd met bbb de naam van de berichtcatalogus ('vraagAntwoord' en 'mutatie' voor de twee hierboven gedefinieerde berichtcatalogi). De wsdl's met het `<portType>` verwerkSynchroneKennisgeving krijgen als naam sss_verwerkSynchroneKennisgeving_bbb.wsdl met bbb de naam van de berichtcatalogus. De wsdl's voor verstrekSynchronisatieBericht krijgt als naam sss_verstrekSynchronisatieBericht.wsdl. Er wordt vooralsnog niet voorzien dat een andere berichtcatalogus ook synchronisatieberichten gaat definiëren, zo ja, dan krijgt deze wsdl in zijn naam de berichtcatalogus. De wsdl's voor beantwoordVraag krijgen als naam sss_beantwoordVraag.wsdl. Hier geldt hetzelfde als voor de sss_verstrekSynchronisatiebericht.wsdl. De wsdl voor verwerkSynchroonVrijBericht krijgt weer als naam sss_verwerkSynchroonVrijBericht_bbb.wsdl, omdat in verschillende berichtcatalogi vrije berichten kunnen worden gedefinieerd. De laatst genoemde wsdl kan alleen voorkomen in overige berichtcatalogi. De service VerwerkTriggerbericht is sectomodelonafhankelijk en komt in geen enkele berichtcatalogus voor. Deze generieke service is gespecificeerd in het bestand stuf0310.services.wsdl van de StUF-onderlaag.

3.5 Overkoepelend schema voor het valideren van berichten

De berichtdefinities liggen voor een sectormodel nu vast in verschillende berichtcatalogi. Voor het valideren van berichten is dit niet handig. Dan is er behoefte aan één schema dat alle berichtdefinities voor een sectormodel bevat. Zo'n schema is eenvoudig te maken door het msg-schema van elke berichtencatalogus via include op te nemen in een overkoepelend schema. Dit schema wordt opgenomen in de folder sss en heeft als naam msg_totaal.xsd. Bij elke toevoeging van een berichtcatalogus zal ook dit schema wijzigen door het toevoegen van een include voor die berichtcatalogus.

4 Koppelvlakken

In het vorige hoofdstuk is uitgebreid gesproken over sectormodellen, berichtcatalogi en voorbeeld wsdl's. Dit zijn specificaties die nodig zijn om uiteindelijk concrete koppelvlakken te specificeren en te implementeren. In feite specificeert een *berichtcatalogus* samen met zijn voorbeeld-wsdl's een verzameling services waarvan een systeem moet aangeven in hoeverre het deze implementeert in voor dat systeem specifieke wsdl's. In de systeem-wsdl's worden de voorbeeld-wsdl's teruggesneden tot de operations van dat systeem. Wanneer het systeem fysiek een andere technologie gebruikt dan WSDL (bijvoorbeeld CPA's in de context van ebMS) dan moet dat systeem de services die het ondersteunt toch ook publiceren als een WSDL. Dit omdat WSDL een breed geadopteerde standaard is die door nagenoeg alle specialisten in het vakgebied wordt begrepen.

Een *koppelvlak* definieert een verzameling services op basis van berichten uit één of meer berichtcatalogi uit één of meer sectormodellen die een systeem in zijn geheel moet implementeren of daadwerkelijk implementeert.

Er is sprake van 'moet implementeren' oftewel een *normatief* koppelvlak, als het koppelvlak een normatieve beschrijving is van functionaliteit die van systemen wordt verwacht. Er is sprake van 'daadwerkelijk implementeert' oftewel een *beschrijvend* koppelvlak, als het koppelvlak een feitelijke beschrijving is van de functionaliteit die een systeem biedt of verlangt van gebruikers van zijn services. Normatieve koppelvlakken hebben veelal geen webadres en beschrijvende koppelvlakken wel. De koppelvlakken van de Landelijke Voorziening BAG en WOZ zijn voorbeelden van een beschrijvend koppelvlak. Het koppelvlak dat de Landelijke Voorziening WOZ verwacht bij een afnemer of bij de gemeente zijn voorbeelden van normatieve koppelvlakken. In geval van een normatief koppelvlak moet een systeem in de eigen wsdl('s) de message-, portType-

en binding-elementen uit de koppelvlak-wsdl integraal overnemen.

Een koppelvlak is precies de verzameling services gebaseerd op één of meer onderliggende berichtcatalogi. De onderliggende berichtcatalogi bestaan onafhankelijk van het koppelvlak en kunnen onafhankelijk van het koppelvlak beheerd worden, mits er nooit berichten verwijderd of gewijzigd worden uit een berichtencatalogus. In het geval van de BAG wordt er dus onderscheid gemaakt tussen een BAG-berichtencatalogus en de daarop gebaseerde BAG/WOZ en BAG/GBA koppelvlakken. Wie weet komen er in de toekomst nog koppelvlakken bij tussen de BAG en andere gemeentelijke systemen. Ook deze koppelvlakken kunnen worden gebaseerd op de BAG-berichtencatalogus.

Vooralsnog mogen koppelvlakspecificaties niet worden opgenomen binnen de folder van een sectormodel of een berichtcatalogus. Deze dienen op de website van de beheerder van het betreffende koppelvlak te worden gepubliceerd of op een speciaal gereserveerde plek op de StUF Community.
In hoofdstuk 3 is gesteld dat in een sectormodel aangegeven moet worden welk type berichten de default zijn, asynchrone of synchrone berichten. Een koppelvlak mag hier echter van afwijken en een andere default definiëren.

5 StUF XML-Schema conventies

5.1 Elementnamen

Het is een best practice om bij het toekennen van namen aan elementen camelcase te hanteren. Het is daarbij van belang dat elk op zichzelf staand woord binnen de naam begint met een hoofdletter behalve de eerste letter (lowerCamelCase genoemd).
Het is van belang hier goed over na te denken. Het volgende voorbeeld illustreert dat.

Het betreft de naamgeving van het element 'statusNotificatie' en de vraag is of het niet juist 'statusnotificatie' moet zijn. Op het eerste gezicht lijkt dat niet zo van belang maar bij nader inzien zit er een subtiel semantisch verschil tussen 'statusnotificatie' en 'statusNotificatie'. Bij de eerste betreft het een 'notificatiebericht waarin de status wordt gemeld' terwijl het bij de tweede gaat om een bericht waarin 'de status van een notificatie' wordt gemeld'.

Elementen die uit een andere entiteit in een entiteit zijn platgeslagen krijgen een prefix die gelijk is aan de mnemonic van de entiteit waaruit ze afkomstig zijn gevolgd door een '.' (punt). Een goed voorbeeld is 'inp.geboorteplaats'.

5.2 Namen van complexTypes en simpleTypes

Ook bij namen van complexTypes en simpleTypes hanteren we camelcase. Daarbij begint echter elk op zichzelf staand woord binnen de naam met een hoofdletter, dus ook het eerste woord (UpperCamelCase of PascalCase genoemd). Het moge duidelijk zijn dat het ook hier van belang is goed na te denken over de naam.

5.3 Documentatie

5.3.1 Patch informatie

Elk XML-schema en wsdl bestand dient te worden voorzien van een 'appinfo' element met de volgende structuur:

```
<appinfo>
   <StUF:onderdeel>xxxxxxxxxxxxxxxxxxxxxxx</StUF:onderdeel>
   <StUF:patch>nn</StUF:patch>
   <StUF:patchdatum>jjjjmmdd</StUF:patchdatum>
   <StUF:schemaversie>nn</StUF:schemaversie>
</appinfo>
```

In een XML-Schema bestand wordt deze structuur geplaatst in het 'annotation' element dat het eerste element is in het 'schema' element. In een wsdl bestand wordt deze structuur geplaatst in het 'documentation' element in het 'definitions' element.

Het element 'onderdeel' wordt daarbij gevuld met de namespace identifier van het onderdeel waar het betreffende XML-Schema of wsdl bestand deel van uitmaakt. Een onderdeel kan een koppelvlak, een sectormodel maar ook een onderlaag zijn. Het schema 'stuf0301.xsd' behoort samen met de andere stuf0301 schema's en wsdl's dus tot het onderdeel 'StUF 3.01 onderlaag' dat geïdentificeerd wordt met de namespace identifier 'http://www.egem.nl/StUF/StUF0301'. **Let op!** Een XML-schema of wsdl bestand dat als targetNamespace identifier 'http://www.egem.nl/StUF/StUF0301' heeft kan deel uitmaken van een onderdeel dat als namespace identifier 'http://www.egem.nl/StUF/sector/bg/0310' heeft.

Het element 'patch' bevat het nummer van de patch waar het XML-schema of wsdl bestand deel van uitmaakt. Indien voor een onderdeel een patch wordt uitgebracht wordt in alle XML-schema en wsdl bestanden die deel uitmaken van dat onderdeel de waarde van dit element aangepast.
In de XML-schema en wsdl bestanden van de eerste versie van een onderdeel heeft dit element de waarde '00'.

Het element 'patchdatum' bevat de datum waarop de patch is uitgebracht. Indien voor een onderdeel een patch wordt uitgebracht wordt in alle XML-schema en wsdl bestanden die deel uitmaken van dat onderdeel de waarde van dit element aangepast.
Voor de XML-schema en wsdl bestanden die voor het eerst geïntroduceerd worden is de waarde van dit element gelijk aan de datum waarop het onderdeel wordt gepubliceerd. Het formaat van dit element heeft het formaat 'jjjjmmdd'.

Het element 'schemaversie' kan voor elk XML-schema en wsdl bestand binnen een onderdeel afwijken. Het wordt alleen met 1 opgehoogd als er daadwerkelijk wijzigingen zijn aangebracht in het betreffende bestand. Initieel heeft dit element de waarde 0.

Elk onderdeel kent een eigen 'patch' en 'patchdatum' waarde. Een wijziging in de bg0310 schema's betekent dus niet automatisch dat de 'appinfo' structuur voor de XML-Schema en wsdl bestanden in de StUF onderlaag wordt aangepast. Echter een wijziging in een van de StUF 3.01 schema's

betekent niet alleen dat de 'appinfo' structuur van alle StUF 3.01 schema's en wsdl's moet worden aangepast maar ook de 'appinfo' structuur van alle daarboven liggende onderdelen (sectormodellen en koppelvlakken).

6 Illustratie

Om het bovenstaande verhaal te illustreren passen we de best practices toe op het sectormodel bg0310. De opdeling van de schema's van bg0310 is weergegeven in Afbeelding 1 op pagina 28. Het plaatje bestaat uit vier lagen waarvan de onderste drie lagen bij het sectormodel bg0310 horen.

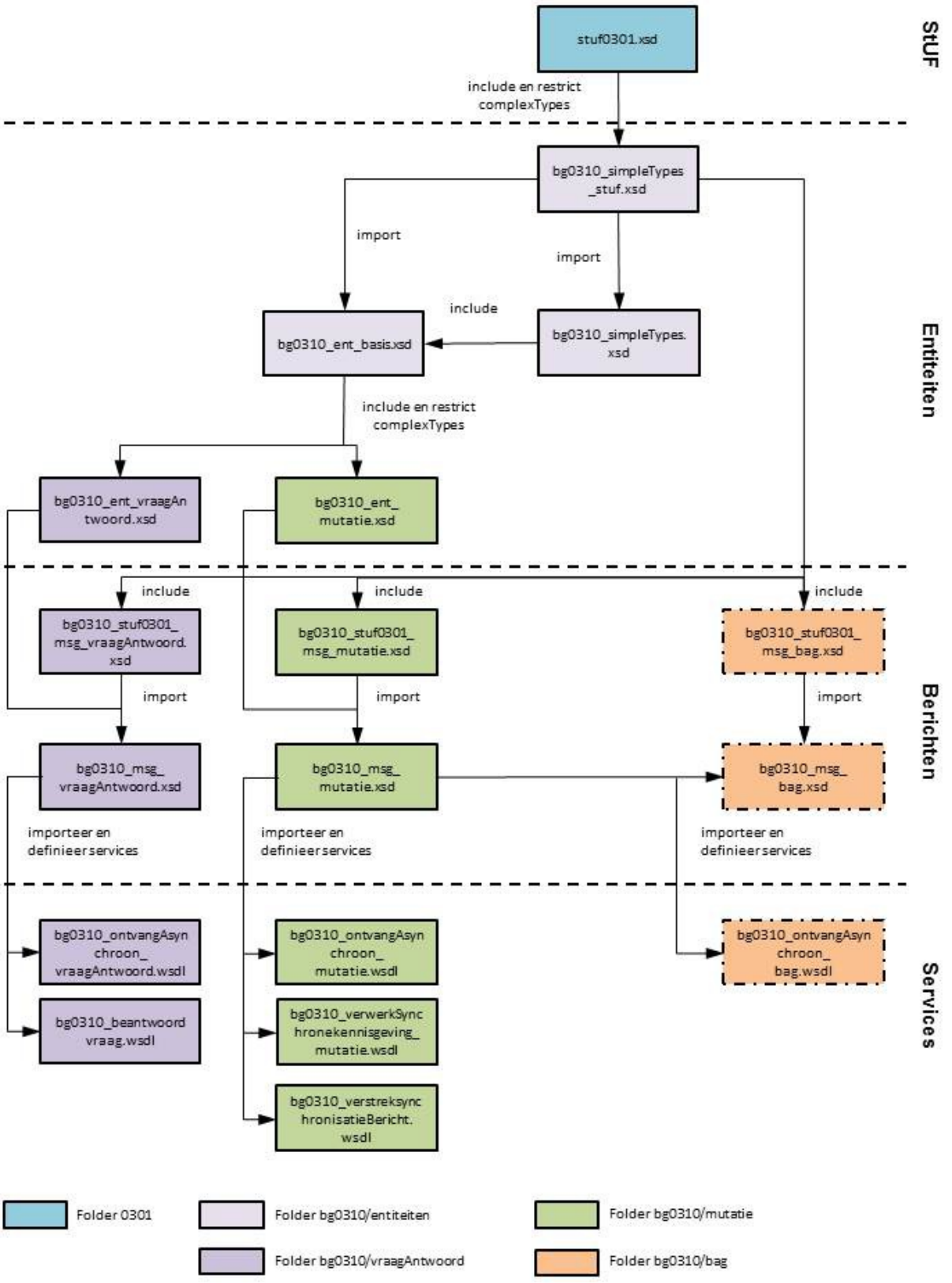
Voor de volledigheid beginnen we in de bovenste laag met het stuf0301.xsd schema dat onderdeel is van de generieke StUF-standaard. Dit schema is de basis waarop alle sectormodellen worden gebaseerd.

In de tweede laag staan de schema's voor het definiëren van de verschillende complexTypes voor de entiteittypen van het sectormodel bg0310. De belangrijkste schema's zijn bg0310_ent_basis.xsd met de basis- en kerngegevens-complexTypes en de schema's met de daarvan afgeleide complexTypes gebruikt binnen de berichtcatalogi voor vraag/antwoordberichten en mutatieberichten. De simpleTypes en de daarvan afgeleide complexTypes met de attributes StUF:noValue en StUF:exact worden gedefinieerd in een apart schema bg0310_simpleTypes.xsd. Daarnaast zijn er nog wat restrictions nodig op complexTypes gedefinieerd in StUF. Deze worden gedefinieerd in het schema bg0310_simpleTypes_stuf.xsd.

In de derde laag staan de schema's voor het definiëren van de berichtelementen van de twee standaard berichtcatalogi die de kern vormen van het sectormodel bg0310, namelijk de vraagAntwoord- en mutatiecatalogus. Voor de berichtelementen zijn daarbij restrictions nodig op StUF-complexTypes voor stuurgegevens en parametersVraag. De berichtcatalogi schema's, herkenbaar aan “_msg” in hun naam doen een include van de schema's met de complexType definities voor vraag/antwoord en mutaties of een include van de berichtcatalogus met mutaties.

In de onderste laag staan de wsdl's voor de in StUF gedefinieerde services voor een berichtcatalogus.

Als wordt besloten om bijvoorbeeld de BAG-berichtcatalogus toe te voegen aan bg0310 dan kan simpelweg een folder BAG in de folder bg0310 toegevoegd worden met de schema's bg0310_msg_stuf_BAG.xsd en bg0310_msg_BAG.xsd. Het is in dit geval niet nodig om nog andere schema's toe te voegen omdat er geen nadere restrictions worden gedefinieerd op de complexTypes in bg0310_ent_mutatie.xsd. De folder BAG bevat ook een voorbeeld-wsdl voor de ontvangAsynchroon service. Daarnaast wordt in het schema bg0310_msg_totaal in de folder bg0310 tevens een include opgenomen van bg0310_msg_BAG.xsd. Bij het toevoegen van een nieuwe berichtcatalogus wijzigen de bestaande wsdl's en schema's niet met uitzondering van het schema bg0310_msg_totaal.xsd ten behoeve van de berichtvalidatie. Voor de leesbaarheid hebben we het schema bg0310_msg_totaal.xsd en de bijbehorende include-pijlen weggelaten in de onderstaande figuur.



Afbeelding 1