Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Technisch ontwerp BRP Leveren

0.6

Datum 03-08-2017
Status Definitief

Inhoudsopgave

VERSIEHISTORIE			
Datum	**Versie**	**Omschrijving**	**Auteur**
20-10-2016	0.1	Eerste opzet TO Maak Bericht Module	Operatie BRP
11-01-2017	0.2	Toevoeging eerste opzet BLOB hoofdstuk	Operatie BRP
10-02-2017	0.3	Toevoeging Algemeen en Bevraging	Operatie BRP
19-03-2017	0.4	Invoeging TO Vrij Bericht	Operatie BRP
27-06-2017	0.5	Toevoegen diensten en bijwerken volgens richtlijnen	Operatie BRP
03-08-2017	0.6	Toevoegen Selectie	Operatie BRP

1 Inleiding

1.1 Beknopte omschrijving

Dit technisch ontwerp beschrijft de software van de diverse lever modules die verantwoordelijk zijn voor het leveren van personen aan afnemers en bijhouders.

1.1 Referenties

#	Document	Organisatie	Versie	Datum
[UCS]	UCS LV.1.MB – Maak BRP bericht	Agentschap BPR	2.18	18-10-2016

Nota: versienummers van 'interne' documenten worden niet opgenomen. Deze dienen consistent te zijn.

1.2 Leeswijzer

Dit document is bedoeld voor ontwikkelaars of software architecten die kennis nodig hebben over de interne structuur van de implementaties van de BRP Leveren diensten.

2 Algemeen

Enkele onderdelen binnen de BRP Leveren modules zijn generiek over de verschillende diensten. Zoals XML parsing en writing. Ook de applicatie-architectuur is hetzelfde binnen de verschillende implementaties. De volgende paragrafen gaan hier in detail op in.

2.1 Applicatie architectuur

Alle levering modules zijn gestructureerd in een schillenarchitectuur, welke ook wel Onion, Hexagonal of Clean Architecture wordt genoemd. Een belangrijk aspect van deze architectuur is dat afhankelijkheden (dependencies) van buitenste naar binnenste schil lopen. BRP Leveren heeft drie schillen.

De buitenste schil, welke Delivery is genoemd, bevat alle code die te maken heeft met het aan- en afvoeren van informatie naar of van de use-case implementaties. Hierbij moet gedacht worden aan JMS, DAL, Web Service en dergelijke code, welke de tweede schil onafhankelijk moet maken van deze specifieke technologieën.

Deze tweede schil hebben we Service genoemd. Hierin bevinden zich alle business services en use case implementaties. Ook kunnen hier de interfaces welke geïmplementeerd worden in de buitenste schil gevonden worden, zoals de DAO interfaces. De projecten in deze schil zijn in principe herbruikbaar, waardoor we makkelijker andere technologieën kunnen ondersteunen.

De domeinobjecten bevinden zich in de binnenste schil, welke we toepasselijk Domain hebben genoemd. Hier kunnen onder andere het levermodel, elementmodel en de interne communicatiemodellen gevonden worden. Ook zijn hier de entiteitmodellen te vinden.

2.2 Verzoek- en antwoordafhandeling

Verzoek- en antwoordafhandeling bevindt zich in de delivery-laag van onze architectuur. De gebruikte objecten zijn te vinden in de Service en Domain lagen.

Voor het doorgronden van de softwarestructuur verwijs ik naar de type hierarchy van VerzoekParser en AbstractGeneriekeBerichtParser voor het parsen en BrpBerichtWriter voor het schrijven van berichten. De volgende twee paragrafen beschrijven de aanpak in detail.

2.3 Binnenkomend verzoekafhandeling

Voor de verzoekafhandeling is gekozen voor DOM in combinatie met XPath. Deze worden gebruikt om de payload van de SOAP requests om te zetten naar verzoek objecten, welke begrepen worden door de service laag. De verzoek objecten zijn dus te vinden in de service-projecten, waarbij de Verzoek interface in brp-service-algemeen de basis biedt.

Deze implementatie is generiek opgezet, vanwege de grote overeenkomsten in de opbouw van de verschillende verzoeken van de BRP Leveren diensten. Deze generieke implementaties zijn te vinden in het brp-delivery-algemeen project, waarbij de AbstractGeneriekeBerichtParser en AbstractDienstVerzoekParser het meest interessant zijn.

De AbstractGeneriekeBerichtParser is de template voor de afhandeling van verzoekberichten voor alle BRP Leveren diensten. Deze klasse stuurt implementaties van VerzoekParser aan, welke het DOM object middels XPath bevragen en een Verzoek object vullen. De generieke parsing binnen alle diensten is te vinden in AbstractDienstVerzoekParser, wat op het moment van

schrijven de BRP stuurgegevens zijn. Parameter en andere dienst specifieke parsing is dan ook te vinden in specifieke delivery modules.

Aangezien het mappen van XML naar verzoekobjecten helemaal uitschrijven voor grote verzoeken niet het handigst is en we binnen deze opzet minder makkelijk een mapper kunnen gebruiken, hebben we voor de Geef Medebewoners dienst binnen brp-delivery-bevraging een aanpak met reflectie en annotaties geïmplementeerd.

2.4 Antwoordafhandeling

Voor het schrijven van antwoordberichten is gekozen voor StAX.

Ook hier geldt dat de implementatie generiek is opgezet. Vandaar ook weer een basis in brp-delivery-algemeen, met de specifieke elementen in de specifieke projecten. In het algemene project is AbstractBrpBerichtWriter de template voor het schrijven van alle BRP berichten binnen Leveren. Bij het bestuderen van de type hierarchy van de BrpBerichtWriter interface wordt duidelijk dat de synchrone antwoorden een aparte tak vormen. Deze berichten bevatten een resultaat, waar de uitgaande verzoeken dat niet doen.

De BerichtEntiteitWriter klasse implementeert een Visitor patroon, welke gebruikt wordt om de specifieke berichtelementen weg te schrijven. Voor de structuur van de berichtobjecten verwijs ik naar de klassen in brp-domain-berichtmodel en de beschrijving hiervan verderop in dit document.

2.5 Uitgaand verzoekafhandeling

Ook uitgaande verzoeken worden geschreven met StAX. De implementaties zijn te vinden in VerwerkPersoonBerichtWriter en VerwerkVrijBerichtWriter. Verder is de implementatie identiek aan de in paragraaf 2.4 beschreven afhandeling van synchrone antwoorden.

3 Dienst Implementaties

Dit hoofdstuk stelt als doel inzicht te geven in de relevante specifieke elementen binnen verschillende diensten. Waarom hebben we bijvoorbeeld voor een bepaalde aanpak gekozen en hoe uit zich dat in de praktijk.

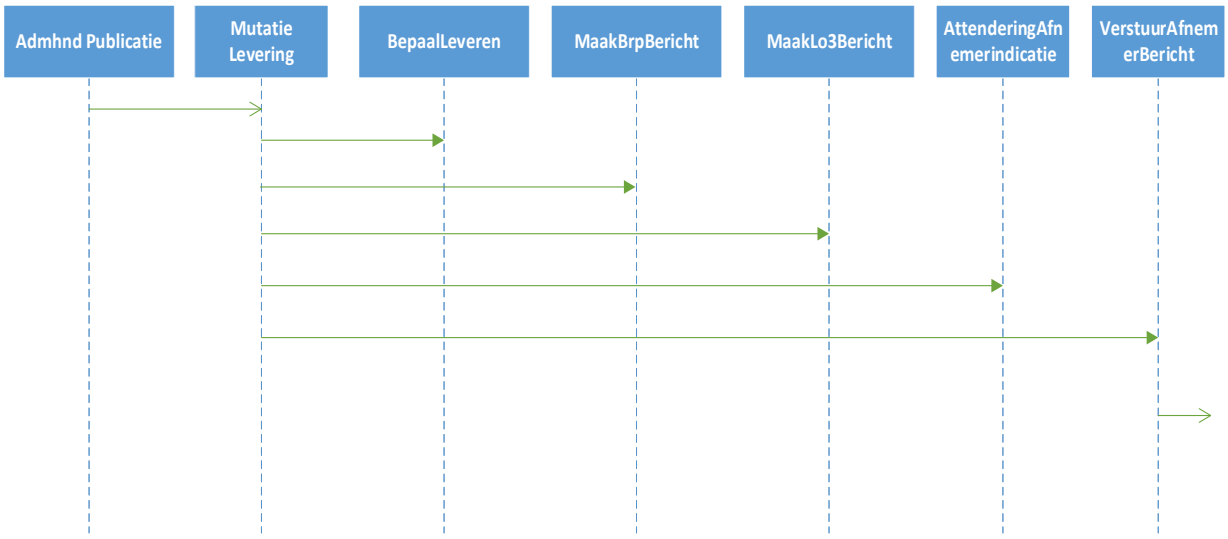
3.1 Mutatielevering

Mutatielevering heeft als startpunt de administratieve handelingen tabel. Hier zijn de bijhoudingen geregistreerd die moeten worden geleverd aan de verschillende afnemers. De handelingen die geleverd worden zijn de handelingen met de status te leveren. Andere handelingen worden niet geleverd, e.g. initiële vulling.

De administratieve handeling component is verantwoordelijk voor het opvragen van de te leveren handelingen. Deze component zet de te leveren handelingen op de Administratieve handelingen queue. De component zorgt er ook voor dat er niet gelijktijdig dezelfde personen worden geleverd waardoor de volgorde niet meer gegarandeerd zou zijn.

De mutatielevering component pakt berichten van queue en bepaalt voor welke autorisaties geleverd moeten worden en maakt berichten en zet deze uiteindelijk op de verzending queue. Als er een fout optreedt in het verwerken van de handeling wordt de status van de handeling op fout gezet.

Het maak BRP bericht gedeelte is generieke maak bericht functionaliteit.

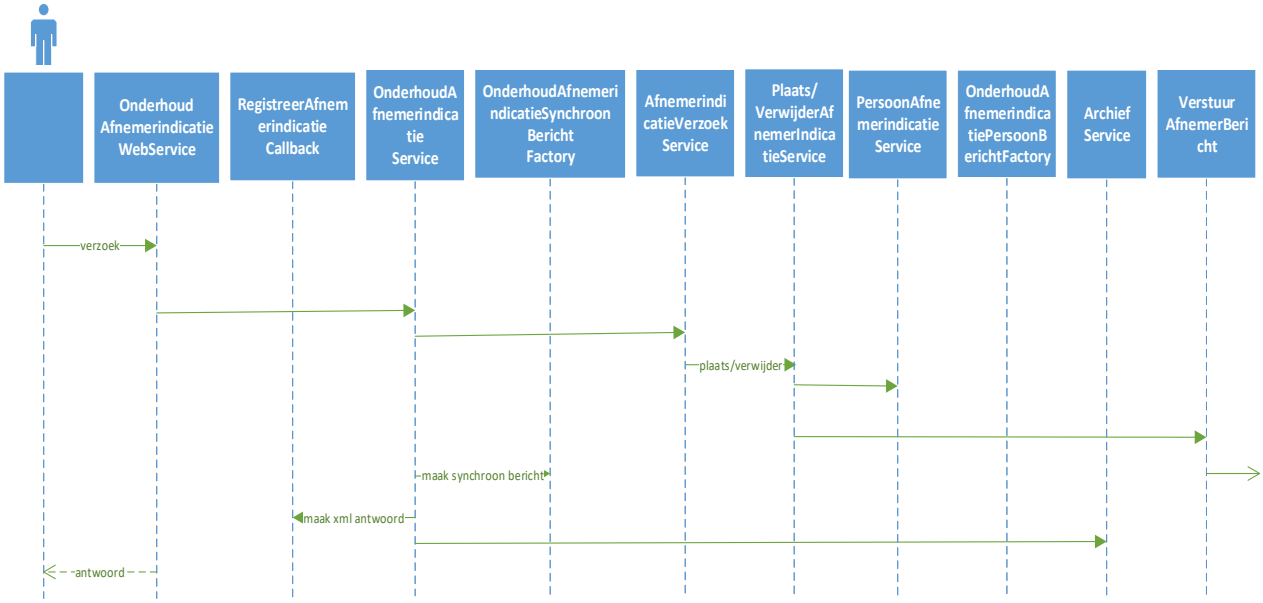


3.2 Afnemerindicatie

3.2.1 *Algemeen*

Afnemerindicatie kent 2 diensten plaatsen en verwijderen. Het plaatsen en verwijderen bestaat uit een aantal stappen. In de web laag vind de parsing van het verzoek plaats en wordt de onderhoud afnemerindicatie service aangeroepen. In de afnemerindicatie verzoek service wordt de algemene autorisatie controle uitgevoerd. Hierna wordt of een plaatsing of verwijdering uitgevoerd inclusief validatie. Hierna wordt het asynchrone bericht gemaakt dat naar de afnemer wordt verstuurd via de verzender component. Hierna wordt het synchrone antwoord bericht gemaakt en wordt de archivering uitgevoerd en synchrone bericht teruggestuurd.

3.2.2 *Flow*

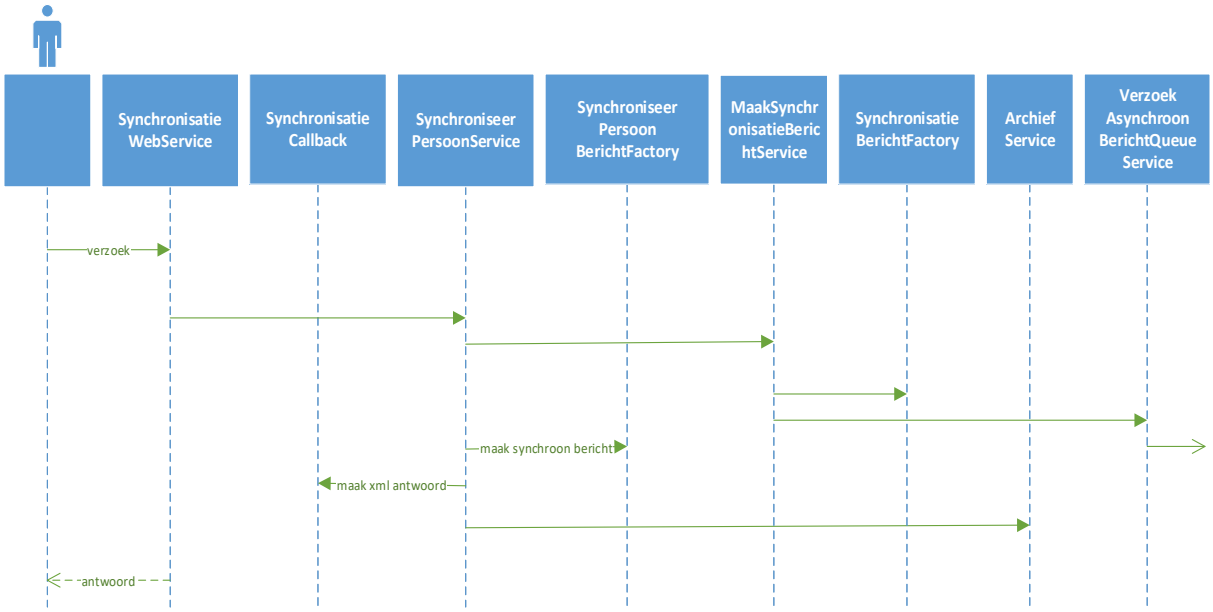


3.3 **Synchroniseer Persoon**

3.3.1 *Algemeen*

Synchroniseer Persoon is onderdeel van het synchronisatie koppelvlak. Het afhandelen van een synchroniseer persoon verzoek kent een aantal stappen. In de web laag vindt het parsen van het web verzoek plaats. Hierna wordt de synchroniseer persoon service aangeroepen. Deze roept de maak SynchronisatieBerichtService aan. Hier worden autorisatie en andere controles uitgevoerd en wordt het aynchrone bericht gemaakt en op de queue gezet naar de verzender component. Daarna wordt in de SynchroniseerPersoonService het synchrone bericht gemaakt.

3.3.2 *Flow*

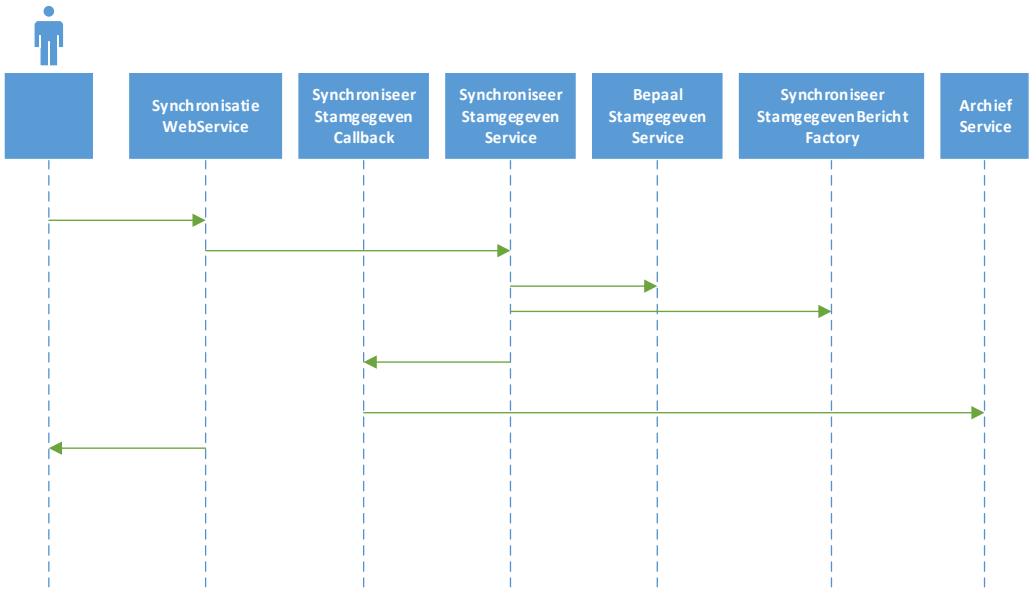


3.4 **Synchroniseer Stamgegeven**

3.4.1 *Algemeen*

Synchroniseer Stamgegeven is onderdeel van het synchronisatie koppelvlak.

3.4.2 *Flow*



3.5 **Bevraging**

De volgende bevragingsdiensten zijn op het moment van schrijven geïmplementeerd:

- Geef details persoon
- Zoek persoon
- Zoek persoon op adres
- Geef medebewoners

Bevraging heeft veel generieke elementen in zich, welke in de eerstvolgende paragraaf besproken zullen worden. De opvolgende paragrafen beschrijven waar nodig de dienstimplementatie in detail.

3.5.1 *Algemeen*

Alle implementaties binnen bevraging hanteren dezelfde flow:

1. Voer maak bericht stappen uit
2. Maak antwoordbericht
3. Archiveer bericht
4. Protocolleer bericht

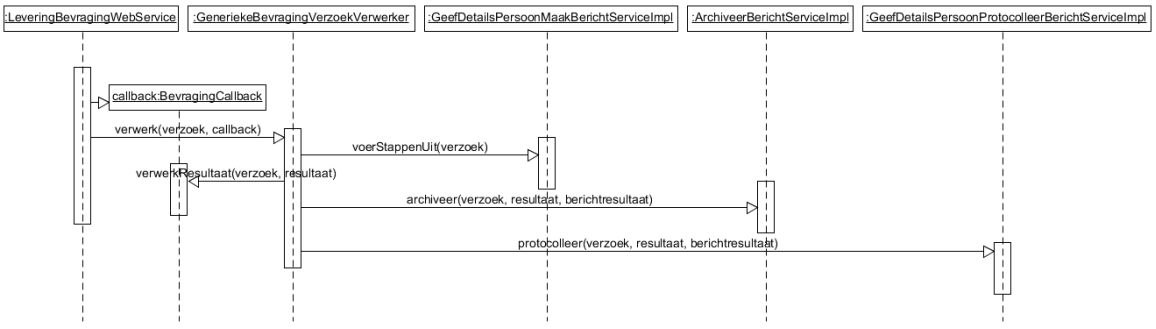
Het uitvoeren van de maak bericht stappen zijn vrij specifiek per dienst, wat betekent dat de implementaties te vinden zijn in de dienst specifieke functionele package. AbstractMaakBerichtServiceImpl voert het generieke deel uit, zoals autorisatie en foutafhandeling. De specialisaties van deze klasse voeren de dienst specifieke logica uit.

Voor het maken van het antwoordbericht wordt een callback naar de delivery laag gedaan, zodat we in onze use cases geen directe afhankelijkheid op XML hebben. Dit resultaat wordt, samen met nog andere attributen, samengevoegd in een object waar archivering en protocollering wat mee kunnen. Dit alles bevindt zich in AbstractMaakBerichtServiceImpl.

Archivering van berichten is ook volledig generiek over alle Bevraging diensten. En zijn daarom geen specialisaties te vinden in de functionele packages. Archivering is geïmplementeerd in ArchiveerBerichtServiceImpl.

Protocollering is ook generiek, op het bepalen van een paar velden na, welke verschillend zijn in de specifieke diensten. Daarom ook een generieke abstracte implementatie in algemeen, AbstractProtocolleerBerichtServiceImpl, en concrete implementaties in de functionele packages.

Het volgende sequence diagram illustreert deze algemene flow voor Geef details persoon.



3.5.2 *Geef details persoon*

Het produceren van een antwoordbericht voor Geef details persoon bestaat uit de volgende stappen:

1. Valideer verzoek
2. Converteer scope elementen
3. Ophalen persoon
4. Maak Geef details persoon bericht

Bij het valideren van het verzoek wordt gekeken of de invoer correct is. Voor Geef details persoon bestaat dit uit het valideren van het peilmoment in het verzoek.

Daarna worden de scope elementen geconverteerd. Hierbij wordt ook gekeken of de opgegeven elementen voldoen aan alle voorwaarden, dus die validatie is niet in stap 1 gebeurd.

Vervolgens wordt de persoonslijst opgehaald. In deze stap worden ook de parameters die nodig zijn voor het ophalen ervan gecontroleerd, zoals het valideren van de opgegeven BSN. Bij correcte invoer wordt een algemene service gebruikt voor het ophalen van de persoonslijst, genaamd SelecteerPersoonService, welke apart beschreven is.in hoofdstuk 5.

Na het succesvol ophalen van de persoonslijst worden parameters voor maakbericht opgesteld en aangeboden aan de module, met als resultaat een VerwerkPersoonBericht. Dit bericht wordt vervolgens gebruikt om in de delivery laag middels een callback een resultaatbericht te maken in XML. Voor een beschrijving van maakbericht wordt verwezen naar paragraaf 5.3.

3.5.3 *Zoek persoon*

Zoek persoon heeft een generiek deel dat wordt gedeeld met Zoek persoon op adres. Dit deel is te vinden in de zoekpersoongeneriek package. Deze basis definieert de flow voor zoeken als volgt:

1. Valideer verzoek
2. Ophalen personen
3. Valideer zoekresultaten
4. Maak Zoek persoon bericht

Verzoekvalidatie binnen Zoek persoon bestaat uit het valideren van de zoekcriteria. Dit betekent dat de zoekbereik parameters aan de verwachtingen moeten voldoen. Verder moet de lijst van zoekcriteria geldig zijn. Per zoekcriterium moeten dus de elementnaam, waarde en optie correct zijn. Het element behorende bij de elementnaam dient een attribuutelement te zijn en de autorisatie moet kloppen. De autorisatie moet het element bevatten waarop gezocht wordt en in geval van historisch zoeken moet het materieel peilmoment een correcte datum bevatten en de groep waar het te zoeken element zich bevindt geautoriseerd zijn op materiele historie. Het dienstspecifieke deel wat betreft invoervalidatie voor Zoek persoon is te vinden in ZoekPersoonMaakBerichtServiceImpl. Voor Zoek persoon wordt er gecheckt of er niet alleen op adres gezocht wordt.

Als het verzoek is gevalideerd kunnen de personen opgehaald worden die voldoen aan de zoekcriteria. De AbstractZoekPersoonOphalenPersoonServiceImpl klasse is hier voor het grootste deel verantwoordelijk voor. Er geldt standaard een maximum van 10 gelijktijdige verzoeken. Bij overschrijding van dit aantal wordt het verzoek niet uitgevoerd en een foutmelding terug gegeven. Op basis van de verzoekgegevens wordt een lijst van persoon ID’s samengesteld door

ZoekPersoonDataOphalerServiceImpl. Deze klasse maakt een SqlBepaler aan, welke op basis van een set parameters een SQL statement produceert. Dit statement wordt door de ZoekPersoonRepository gebruikt om te bepalen wat de query kost, middels een explain. Als deze te duur is (standaard groter dan 250), dan wordt er een QueryTeDuurException gegooid. Is het niet te duur wordt de SQL uitgevoerd, wederom in de ZoekPersoonRepository. Mocht er een timeout optreden bij het uitvoeren van het statement, zal een QueryCancelledException gegooid worden.

Gaat alles goed dan komt er een lijst personen terecht bij AbstractZoekPersoonOphalenPersoonServiceImpl, welke zal kijken of het aantal tussenresultaten niet te groot is. (standaard groter dan 250). Als het aantal resultaten niet te groot is worden de ID's gebruikt om Persoonslijsten op te halen. Deze worden vervolgens gefilterd, waarna er een laatste check op het aantal resultaten plaatsvindt. Dit gebeurt voor Zoek persoon in ZoekPersoonOphalenPersoonServiceImpl en checkt of het aantal resultaten niet het in de dienst gespecificeerde maximum aantal overschrijdt (standaard 10).

De ZoekPersoonBerichtFactoryImpl zal vervolgens de maakbericht parameters opstellen, deze aanbieden aan maakbericht en een VerwerkPersoonBericht terug krijgen.

3.5.4 *Zoek persoon op adres*

Deze dienst is in de meeste opzichten hetzelfde als Zoek persoon. Daarom zal deze paragraaf alleen de afwijkingen aanstippen.

ZoekPersoonOpAdresMaakBerichtServiceImpl heeft een referentie naar ValideerZoekPersoonOpAdresZoekCriteriaService, welke het ZoekPersoonOpAdresVerzoek valideert. Zoek persoon op adres kent enkele specifieke voorwaarden waaraan het verzoek moet voldoen wil het een zoekopdracht op adres zijn. De bedrijfsregels die dit bepalen zijn verwerkt in deze service.

3.5.5 *Geef medebewoners (van persoon)*

De specifieke logica voor Geef Medebewoners is te vinden in GeefMedebewonersMaakBerichtServiceImpl. De stappen binnen deze dienst zijn de volgende:

1. Valideer verzoek
2. Converteer naar Zoek persoon verzoek
3. Ophalen personen
4. Valideer zoekresultaten
5. Filter relaties
6. Maak Geef medebewoners bericht

De validatie bestaat uit het controleren van het peilmoment, via de PeilmomentValidatieService, en een check of er gezocht wordt op medebewoners op BAG-id óf per adres óf van een persoon.

Geef medebewoners heeft, in tegenstelling tot Zoek persoon, een specifieke set aan parameters. Daar waar je binnen Zoek persoon kunt zoeken op elementnaam, zijn de elementen binnen Geef medebewoners specifiek en beperkt. Dus worden, om van de Zoek persoon functionaliteit gebruik te kunnen maken, de identificatiecriteria uit het Geef medebewoners verzoek geconverteerd naar Zoek persoon zoekcriteria. Hiervoor is de ZoekCriteriaConverteerUtil verantwoordelijk. GeefMedebewonersVerzoek.Identificatiecriteria heeft attributen voorzien van

ZoekCriteriaElement annotaties. Deze mappen het attribuut naar een Element van een bepaald type. Deze types zijn PERSOON, ADRES en BAG, de drie elkaar uitsluitende zoekopties binnen Geef medebewoners. In geval van zoeken op BAG-id of adres, worden de zoekcriteria geconverteerd. In geval van zoeken op persoon wordt de BevragingSelecteerPersoonService gebruikt om een persoon op te halen. De prioriteit voor zoeken op persoon is BSN, a-nummer, persoon ID. In deze stap wordt ook de invoer gevalideerd. Als er een geldige persoon gevonden wordt, wordt GeefMedebewonersBepaalBAGSleutelService gebruikt om een BAG-id te bepalen van de gevonden persoon. In deze service worden nog extra checks gedaan, zoals bepalen dat de persoon niet overleden is en dat het adres een Nederlands is. Verder moet het het actuele adresrecord een BAG-id bevatten. In dit geval zal deze gebruikt worden om een ZoekPersoonGeneriekVerzoek op te stellen en te zoeken op het gevonden BAG-id.

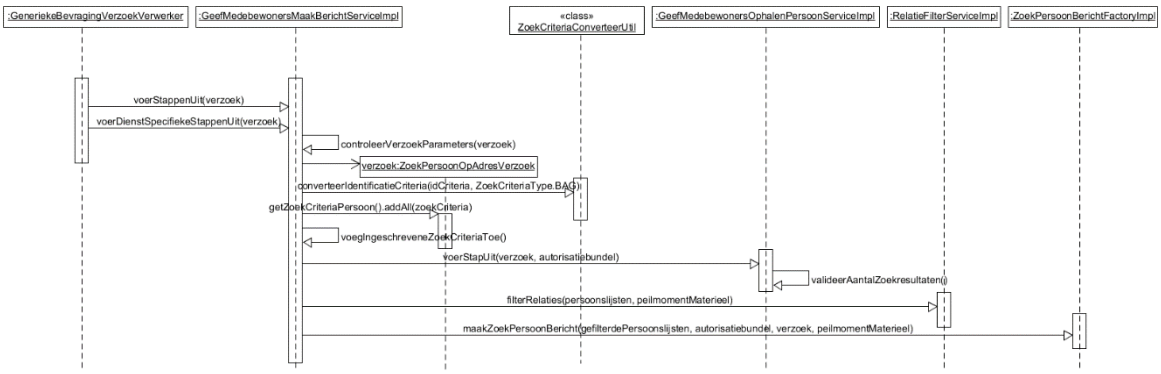
GeefMedebewonersOphalenPersoonServiceImpl is een specialisatie van de generieke Zoek persoon abstractie AbstractZoekPersoonOphalenPersoonServiceImpl, welke gebruikt wordt om met het opgestelde verzoek te zoeken naar personen.

Deze klasse bevat een methode om de zoekresultaten te valideren. Dit houdt voor Geef medebewoners in dat er gekeken wordt of alle gevonden personen dezelfde BAG-id hebben. Mochten er personen tussen zitten zonder BAG-id, die overleden zijn of überhaupt geen adresgegevens hebben, dan worden deze uit de lijst van resultaten verwijderd.

Vervolgens wordt de RelatieFilterService gebruikt om de gerelateerde personen uit de persoonslijsten te verwijderen die niet op het peilmoment woonachtig zijn op het adres.

Voor het maken van het VerwerkPersoonBericht wordt de ZoekPersoonBerichtFactory gebruikt binnen Geef medebewoners.

Het volgende sequence diagram illustreert Geef medebewoners op BAG-id.



3.6 **Vrij bericht**

3.6.1 *Context*

De Vrij Bericht web service implementeert de volledige functionaliteit voor het versturen en ontvangen van vrije berichten, zoals gedefinieerd in **[1]**. In dit TO wordt besproken uit welke componenten de service is opgebouwd en hoe de interactie daartussen verloopt.

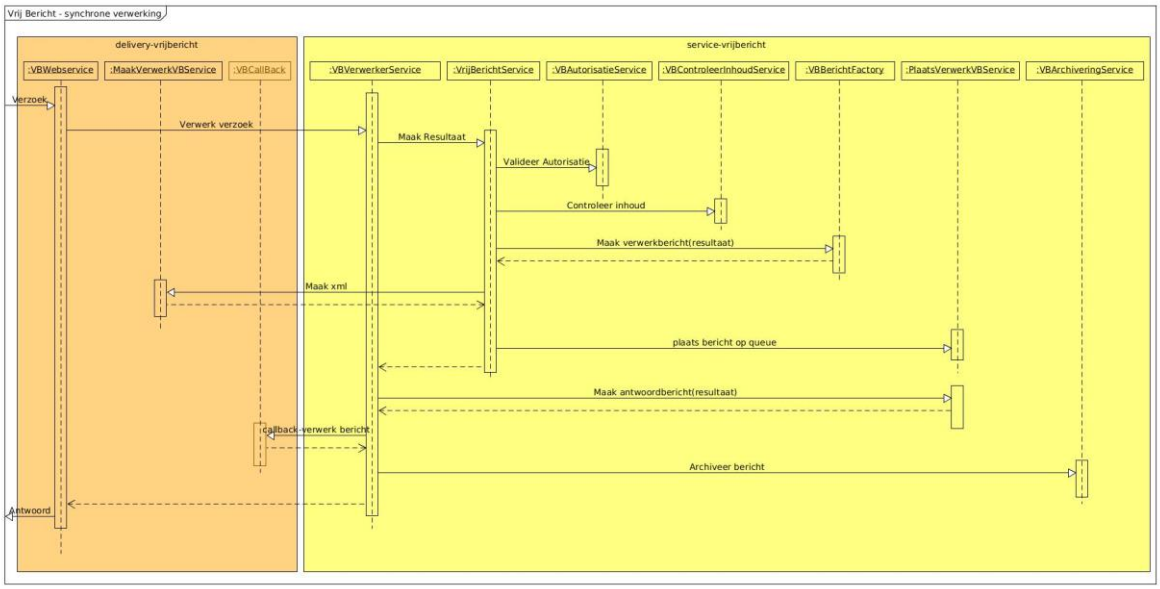
Er is een besloten tot een apart TO voor Vrij Bericht, omdat de Vrij Bericht service strikt genomen geen leveringsdienst is zoals bijvoorbeeld Geef Details Persoon, Synchroniseer Persoon etc. De functionaliteit onderscheidt zich inhoudelijk duidelijk van reguliere leveringsdiensten en dit is deels ook in de implementatie terug te zien. Hoewel waar mogelijk bestaande functionaliteit wordt hergebruikt, is de flow afwijkend van reguliere diensten : een aantal onderdelen (met

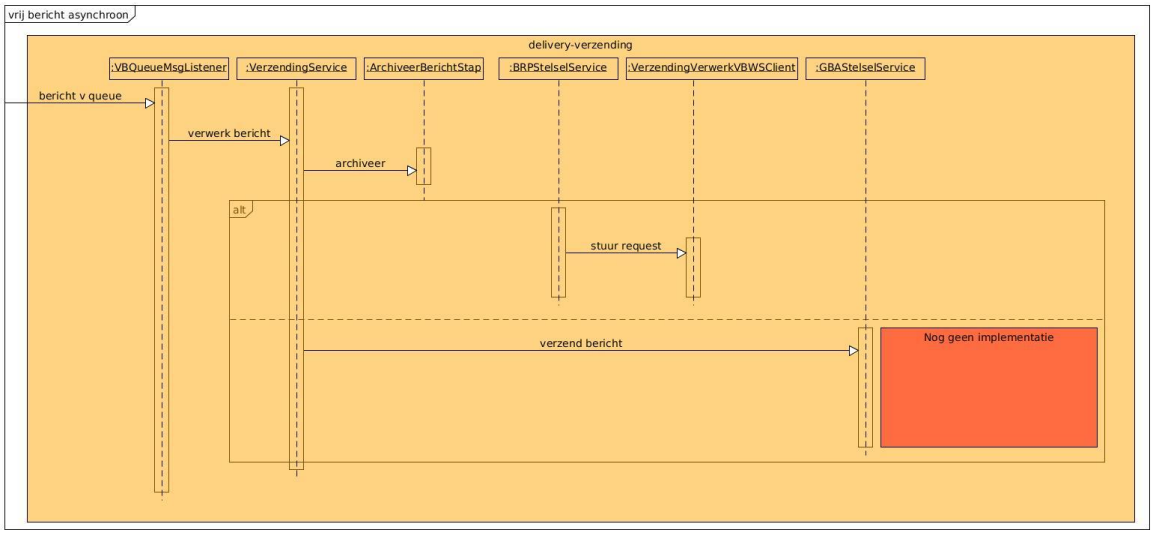
name autorisatie) is specifiek gemaakt voor vrij bericht en een reeks generieke onderdelen uit de leveringsdiensten zijn in het geheel niet van toepassing (bv. maak bericht, protocollering).

Een belangrijke aandachtspunt m.b.t. de implementatie was modulariteit : het is nog niet precies duidelijk hoe lang de Vrij Bericht service in huidige vorm beschikbaar zal moeten blijven. Door de huidige opzet kan de service op een later moment eenvoudig uitgefaseerd worden.

3.6.2 *Flow verzoek*

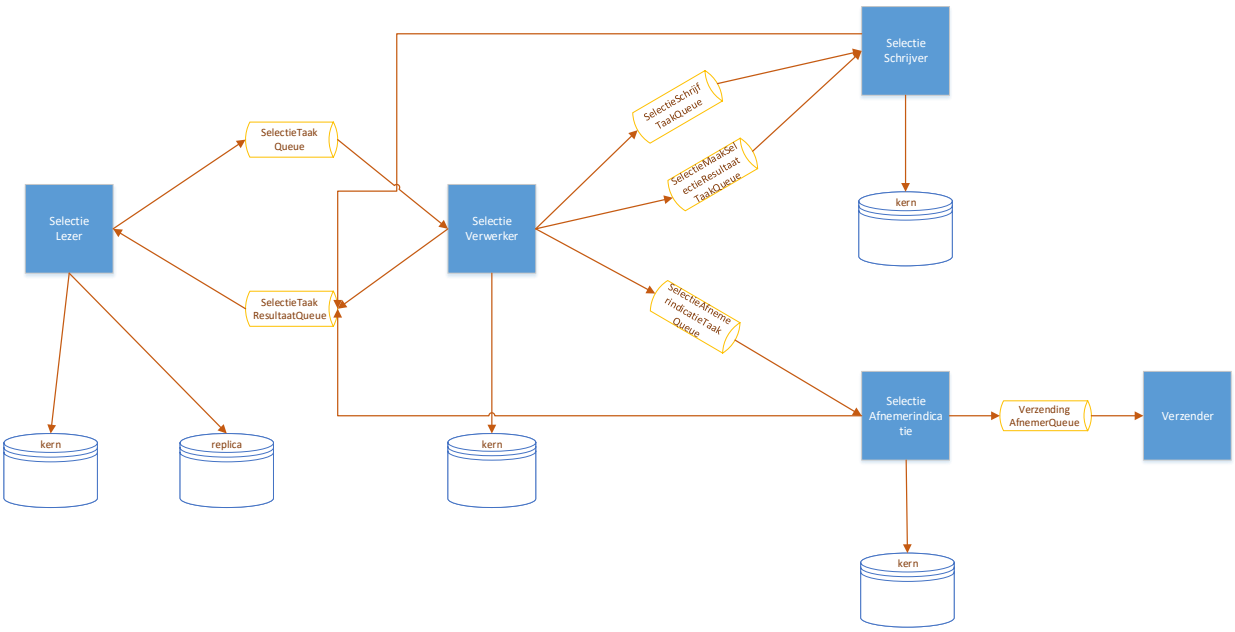
Hieronder staat een schematisch overzicht van de Vrij Bericht functionaliteit in de vorm van twee sequencediagrammen. Het eerste diagram geeft de flow weer met betrekking tot de synchrone verwerking. Er komt een verzoek tot het versturen van een vrij bericht binnen. Na archivering, autorisatiecontrole en inhoudelijke controle van het verzoek wordt het asynchroon XML-bericht gemaakt en op een queue gezet. Vervolgens wordt middels een callback het synchroon bericht resultaatbericht opgesteld, gearchiveerd en geretourneerd. Het tweede diagram geeft de asynchrone verwerking van het antwoordbericht weer, het bericht wordt van de queue gehaald, gearchiveerd en verzonden.





3.7 **Selectie**

3.7.1 *Overview*



3.7.2 *Beschrijving*

Selectie is onderverdeeld in de volgende componenten

Selectie Lezer: Verantwoordelijk voor het batchgewijs ophalen van perscache records (blobs) uit de selectie replica database. Voor een batch wordt een lijst van SelectieVerwerkTaakBerichten opdrachten gemaakt welke gestuurd worden naar de selectie verwerkers. Deze bevat de batch van persoonsgegevens en een x aantal autorisaties.

Configuratie:

Via de volgende config properties kunnen de groottes worden aangepast van de verwerkberichten:

- brp.selectie.lezer.selectietaak.autorisaties
- brp.selectie.verwerker.selectietaak.blob (kleiner of gelijk aan lezer batchsize)

De grootte van de thread pool voor het ophalen van batches van perscache records

- brp.selectie.lezer.poolsize

De grootte van de batch van perscache records:

- brp.selectie.lezer.batchsize

Selectie verwerker:

Verantwoordelijk voor het omzetten van de blobs naar persoonslijsten hier de relevante berichten voor maken voor de gegeven autorisaties. Voor selectie bestand worden er persoons fragmenten gemaakt waar later header en footer voor gemaakt worden. Voor afnemerindicatie met bericht worden er volledige berichten gemaakt.

Deze berichten worden op de schrijver verwerk queue gezet of op de afnemerindicatie queue. Voor de schrijver worden de berichten op de schrijver queue geplaatst met een selectie taak is als message group header. Dit zorgt ervoor dat er maar 1 proces tegelijkertijd in dezelfde selectie resultaat folder schrijft.

Deze component is schaalbaar.

Configuratie:

Het aantal listeners

- brp.selectie.verwerker.jms.concurrency

De root folder waar selectie bestanden (bsn/anr lijsten) zich bevinden:

- brp.selectie.verwerker.selectiebestandenfolder

Het aantal threads voor een verwerker (per listener)

- brp.selectie.verwerker.poolsize

Selectie schrijver:

Verantwoordelijk voor het wegschrijven van XML fragmenten naar filesysteem.

Als laatste stap in het proces ontvangt de schrijver per autorisatie nog de opdracht om van fragmenten selectie resultaat bestanden te maken, een totalenbestand en een steekproefbestand.

Deze component is schaalbaar.

Configuratie:

Het aantal listeners:

- brp.selectie.schrijver.jms.concurrency

Het aantal threads voor het verwerken van een schrijf opdracht (per listener)

- brp.selectie.schrijver.poolsize

De root resultaat folder

- brp.selectie.schrijver.resultaatfolder

Selectie Afnemerindicatie:

Verantwoordelijk voor het plaatsen/verwijderen van afnemerindicaties en het eventueel verzenden van een verwerk persoon bericht.

Deze component is schaalbaar.

Configuratie:

Het aantal listeners

- brp.selectie.afnemerindicatie.jms.concurrency

4 Maak Bericht

4.1 Context

De Maak Bericht functionaliteit is een verzameling van componenten die samen een aantal services en objecten aanbieden waarmee de verschillende diensten zoals bijvoorbeeld Geef Details Persoon, Synchroniseer Persoon en Mutatielevering op basis van doelbinding een bericht kunnen maken zoals dat verstuurd moet worden aan afnemers van de verschillende diensten.

Dit TO beschrijft het samenstellen van een persoon of meerdere personen zodat deze geleverd kunnen worden aan dienst afnemers in de vorm van een Resultaatbericht (e.g. bevraging, zoek persoon) of een NotificatieBericht (Volledige of Mutatie berichten).

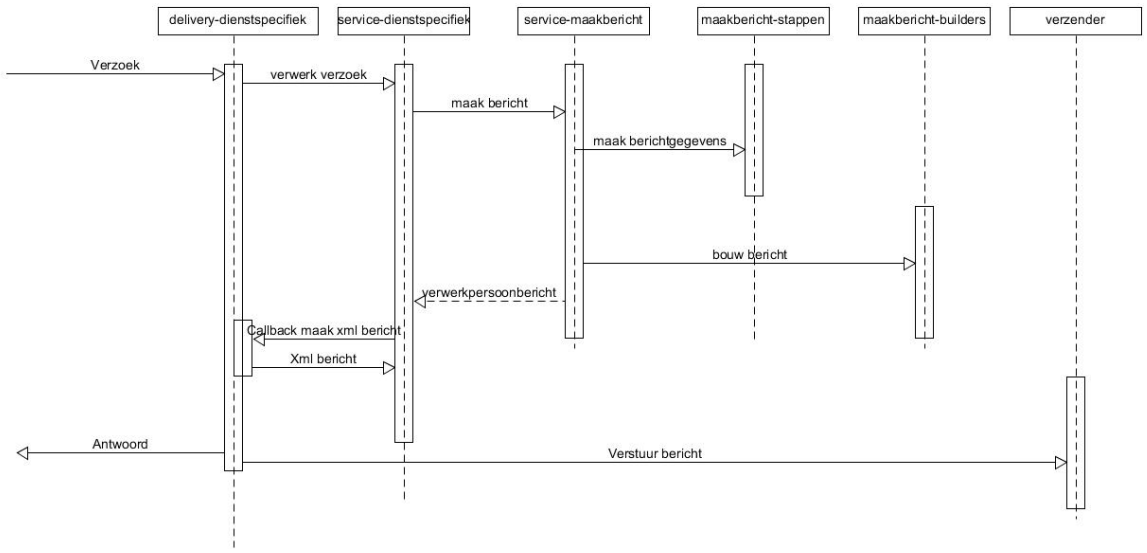
Op basis van de door de aanleverende use case meegegeven gegevens en het type bericht, wordt bepaald welke gegevens in het bericht opgenomen moeten worden. Dit is afhankelijk van de persoonslijst(en) zelf, de eventuele administratieve handeling, de autorisaties van de afnemer op de groepen en elementen, op de historie (formeel en materieel) en op de verantwoordingsinformatie.

Het samenstellen van een te leveren persoon uit een volledige PL bestaat uit een aantal stappen die een set van regels implementeert. Dit vormt de kern van een aantal diensten en is de kern van de leversoftware. Hierdoor is een TO op zijn plaats.

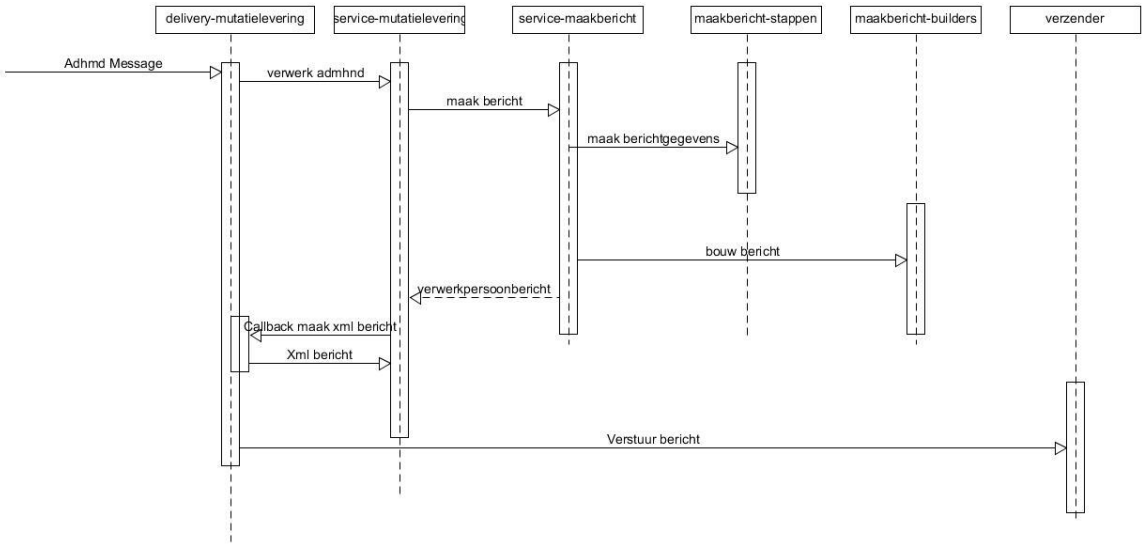
Hieronder staat een schematisch overzicht waar de maakbericht functionaliteit geplaatst kan worden. Er komt een verzoek binnen (e.g. geef details, een handeling, een zoek persoon) en op basis hiervan moet een antwoord bericht worden samengesteld met 1 of meerdere personen. In de volgende hoofdstukken wordt steeds een stuk van de functionaliteit toegelicht. Verdere technische documentatie bevindt zich in de javadoc.

Globaal gezien vallen de flows waarin verwerkpersoon berichten worden gemaakt in 2 delen uiteen. 1 deel zijn de verzoeken zoals Geef Details Persoon, Zoek Persoon, Synchroniseer Persoon etc. De andere flow is de flow vanuit mutatielevering.

4.2 Flow verzoek



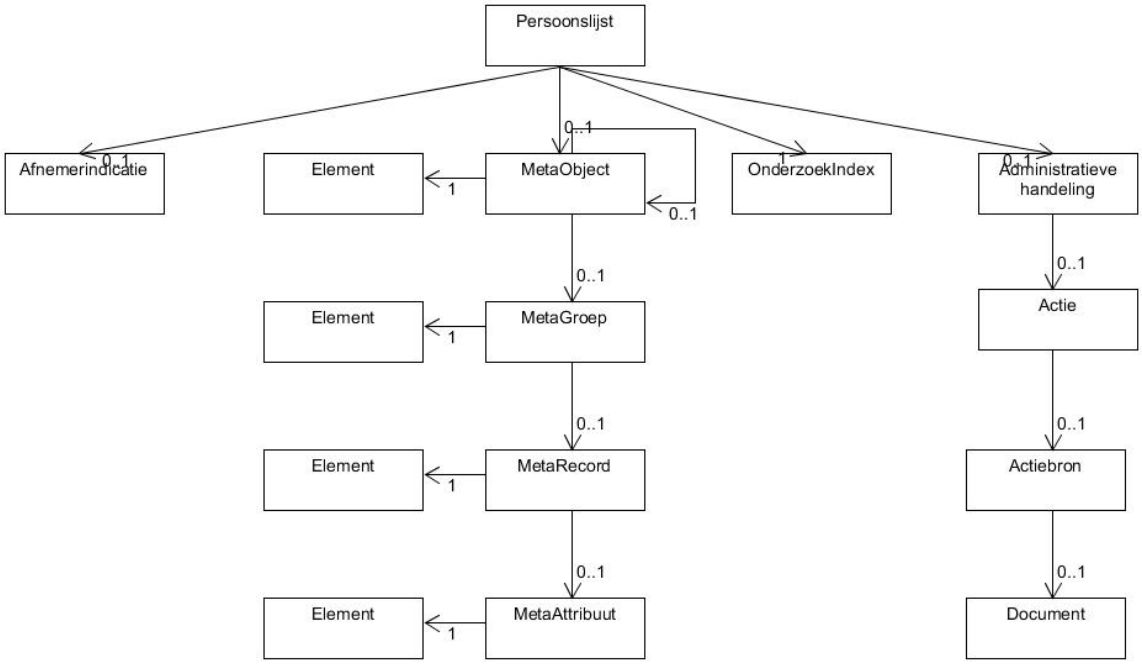
4.3 Flow mutatielevering



5 Componenten

5.1 **Persoonslijst in BRP leveren**

5.1.1 *Overzicht*



De persoonslijst zoals die gebruikt wordt in BRP leveren bestaat uit een generiek gedeelte en een specifiek gedeelte.

Het grootste deel van het onderliggende specifieke model is generiek gemaakt voor leveren. Leveren bestaat grotendeels het maken van persoonsberichten in de maakbericht module. De persoonsberichten worden gemaakt door een aantal filterstappen op de persoonsgegevens uit te voeren. Deze filterstappen zijn bijna allemaal generiek, d.w.z. ze hebben geen relatie met de inhoudelijke gegevens van de persoonslijst. Uitzondering hierop vormen de afnemerindicaties, onderzoek gegevens en de verantwoording gegevens. Deze zijn dan ook specifiek gemodelleerd. De verantwoordingegevens zijn ook specifiek op de MetaRecord's gemodelleerd. Deze informatie wordt veelvuldig specifiek gebruikt. Deze informatie is echter ook generiek beschikbaar als attribuut onder het record.

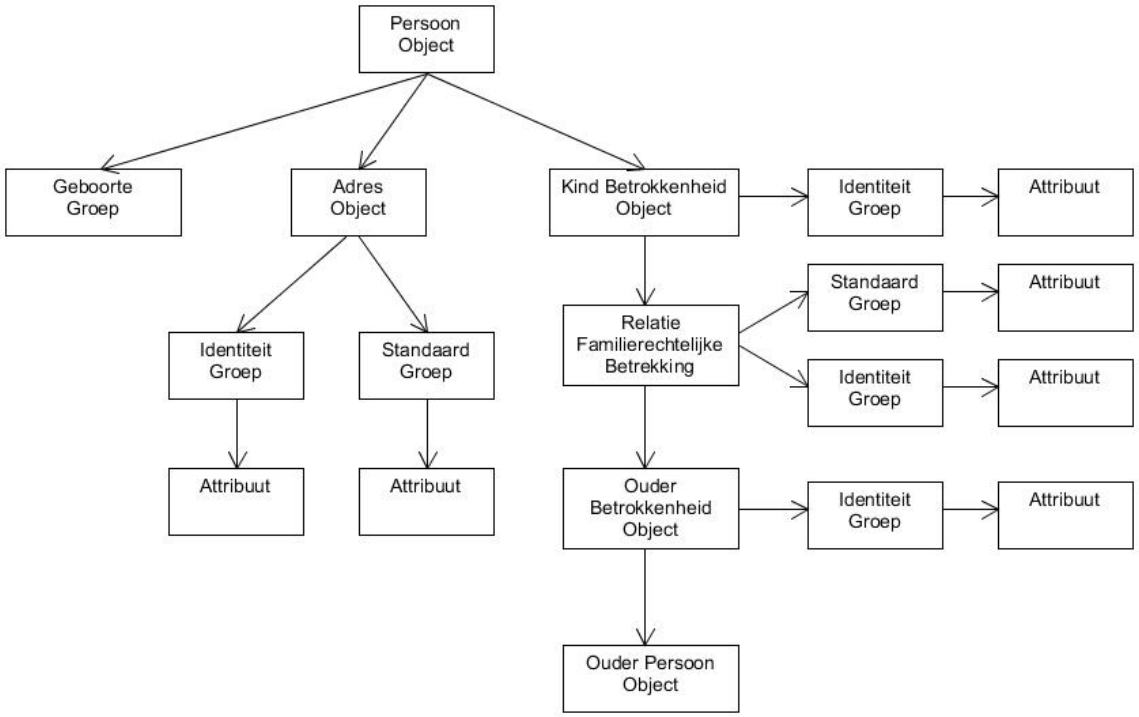
5.1.2 *Metamodel*

Het metamodel is een abstracte representatie van het specifieke onderliggende model. In de elementtabel is de structuur beschreven van het persoonsmodel.

Het metamodel is ook een hiërarchische representatie van de structuur zoals beschreven in de element tabel. Hierin wordt niet strikt de structuur gevolgd zoals deze vastligt in de element tabel. Betrokkenheden zijn in de element tabel zelfstandige entiteiten. Deze hebben hier geen relatie met een persoon. Deze betrokkenheden hebben in de daadwerkelijke persoonslijst wel een relatie met een persoon.

Een Metaobject bevat direct onderliggende groepen en zelf weer objecten. Voorbeelden van groepen die direct onder een metaobject vallen zijn bijvoorbeeld een geboortegroep direct onder persoon, adres standaard en identiteitsgroep onder adres object.

Hieronder is een fragment van een Persoonslijst weergegeven zoals het specifiek in het BMR uitgewerkt is. Dus met de specifieke entiteiten. Een voorbeeld van een persoon met een geboorte groep en een adres en een kind betrokkenheid met ouder.



Dit is een beschrijving van een onderdeel van de persoonslijst. Een verdere uitwerking volgt dit generieke patroon.

In het gegevensmodel kan een groep 1 of meerdere keren voorkomen. Dit noemen we de voorkomens van een groep. Een identiteitsgroep zal altijd maar 1 keer voorkomen per object, andere groepen kunnen meerdere keren voorkomen. Een voorkomen komt kort gezegd overeen met een rij in de database. (TODO verwijzing naar database model). In de voorkomens zien de we historiepatronen terugkomen zoals die in de BRP bestaan. (TODO verwijzing).

Het metamodel zoals gebruikt in BRP leveren is dus een combinatie van een abstractie van het BMR model en een structuur die de data bevat. Deze datatoevoeging zien we in de MetaRecord's die een Metagroep bevat.

5.1.3 *Element tabel*

In de elementtabel is de structuur van het gegevensmodel vastgelegd. Daarnaast bevat de elementabel autorisatiegegevens per element, database informatie waar de gegevens zijn opgeslagen en informatie die gebruikt wordt tijdens het maken van een XML bericht.

De elementtabel is runtime statisch beschikbaar, d.w.z. een statische representatie wordt tijdens de build van de software gemaakt. Een element kan dus niet runtime worden aangepast. Dit zal geen effect hebben op een eenmaal gebouwde module. Een aanpassing in de elementabel kan ook niet zonder uitvoerig testtraject worden doorgevoerd.

Hieronder is kort per gegevenselement de functie beschreven

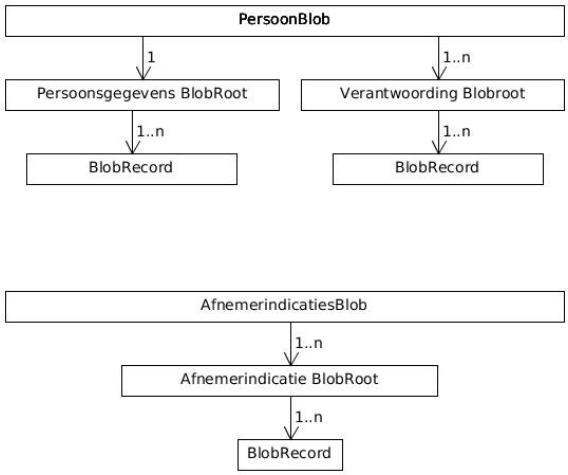
- Soort: De typering van een element. Object, groep of attribuut.
- Naam: De volledig naam waaronder dit element bekend is. Persoon.Adres.Tijdstipregistratie is de volledige naam voor tijdstipregistratie attribuut onder adres.
- Elementnaam: de naam voor de abstractie. Tijdstipregistratie voor bovengenoemd voorbeeld.
- Objecttype. De verwijzing naar het bovenliggende object van een element.
- Groep. De verwijzing naar de bovenliggende groep voor een element. Kan alleen gevuld zijn voor attributen.
- Volgnummer. De volgnummer van het element in de berichtenstructuur.
- Alias van. De alias verwijzing van een element. (TODO uit bmr overnemen)
- Type. De type aanduiding van een element. (TODO uit bmr overnemen)
- Autorisatie. De element autorisatie voor een element. Bijvoorbeeld verplicht, via groepsautorisatie, niet verstrekken etc.
- Tabel, identdbschema, identdb, histabel, hisidentdb, dbobject, hisdbobject. Informatie waar in de database het gegeven zich bevindt. TODO, verwijzing toevoegen
- TypeIdentDb, type zoals gedefinieerd in database
- Historiepatroon
- Verantwoordingcategorie
- TypeidentXSD: Niet gebruikt??
- Expressiebasistype: Typering van element. TODO hernoemen. Type is nodig om bijvoorbeeld datum als datum te kunnen herkennen.
- Srtinh: Gebruikt om te herkennen of een gegeven een stamgegeven is.
- Inverseassociatieidentcode. Rendering attribuut. Wordt gebruikt om te bepalen of een tussen element getoond moet worden. Zoals bij persoon, adressen, adres in de berichten structuur.
- Identxsd. De naam van het element in berichten
- Minumlengte. De minimumlengte van een element attribuut. Gebruikt in validatie zoekcriteria en voor rendering.
- Maximumlengte. De minimumlengte van een element attribuut. Gebruikt in validatie zoekcriteria en voor rendering
- Sorteervolgorde. Aanduiding of een attribuut gerbruikt wordt als sorteerelement. Gelijke objecten onderling worden bijvoorbeeld gesorteerd. Dit kan op volgnummer, maar ook bijvoorbeeld op nationaliteitcode bij adressen.
- Datumaanvanggeldigheid.
- Datumeindegeldigheid

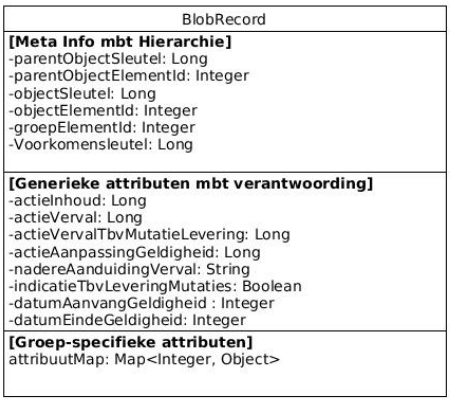
5.2 Laden Persoonsmodel: BLOB

Het persoonsmodel wordt ingeladen op basis van blobs. De rationale voor het gebruik van blobs is gebaseerd op het principe van gecontroleerde redundantie : data wordt redundant opgeslagen teneinde een optimalisatie m.b.t. bevraging te bewerkstelligen. In dit geval wordt de optimalisatie bereikt door eenmalig een blob uit te lezen, met daarin alle gegevens die nodig zijn om een complete persoonslijst op te bouwen. Het uitvoeren van complexe en vaak dure queries wordt zodoende vermeden en bovendien is de structuur van de blob dusdanig dat het opbouwen van het metamodel vereenvoudigd wordt. De consistentie van de redundant opgeslagen data wordt gewaarborgd, doordat blobs na elke mutatie (m.a.w. na een bijhouding of conversielevering) aangepast of nieuw aangemaakt worden.

5.2.1 *Structuur van de blob.*

Persoonsgegevens en afnemerindicaties op een persoon wordt in separate blobs opgeslagen. Hier zijn een aantal redenen voor. Ten eerste worden persoonsbeelden en afnemerindicaties apart gemuteerd door respectievelijk bijhouding en levering : door blobs te scheiden hoeft bij een mutatie op afnemerindicaties niet opnieuw het gehele persoonsbeeld verblobt te worden. Evenzo is het wat betreft levering niet altijd nodig om beschikking te hebben over afnemerindicaties op een persoon en kunnen deze dus bij het creeren van een persoonslijst genegeerd worden. Onderstaande figuur geeft de structuur weer van de persoon- en afnemerindicatieblobs.





Persoonblob

Een persoonblob bevat een platgeslagen representatie van de persoons- en verantwoordingsgegevens in de vorm van een verzameling BlobRecords. De blobrecords zijn gestructureerd door gebruik te maken van containers (BlobRoots). Elke persoonblob bevat 1 container met blobrecords met persoonsgegevens (inclusief onderzoeksgegevens) en een lijst containers voor records met verantwoordingsgegevens, waarbij elke container in de lijst een administratieve handeling representeert. Deze structuur wordt gehanteerd om de terugconversie van de blob te vereenvoudigen.

Elk blobrecord in de persoonblob bevat zowel de persoons- of verantwoordingsgegevens, als de informatie die nodig is om de hiërarchische structuur van het metamodel op te bouwen (de groep, het object en het parentobject waartoe het record behoort, voorkomensleutel etc.). Een reeks attributen is direct op het blobrecord gezet (actieinhoud, actieverval, actieaanpassinggeldigheid, actietbvmutatielevering, nadereaanduidingverval en datum aanvang/einde geldigheid). Deze zijn op dit niveau gemapt, omdat het gaat hier om attributen die in elke standaardgroep aanwezig zijn . De overige groep-specifieke attribuutwaarden zijn in het blobrecord in een aparte map geplaatst, met als key het attribuutelement-id.

Voor elke hoofdpersoon in een persoonblob worden alle gerelateerde betrokkenheden in de blob opgenomen. Vanuit performance overwegingen betreft het hier meer dan enkel een verwijzing naar de betrokken persoon, oftewel voor het ophalen van gegevens van betrokken personen hoeft geen aparte blob ingelezen worden. De omvang van de blob blijft tot het noodzakelijke beperkt door voor de betrokkenheden enkel de voor levering benodigde gegevens (de zgn. identificerende gegevens) op te nemen in de blob. Onder identificerende gegevens vallen voorkomens van de groepen Persoon.Identificatienummers, Persoon.Samengestelde naam, Persoon.Geboorte en (behalve voor de kind-betrokkenheid) Persoon.Geslachtsaanduiding.

Afnemerindicatiesblob

Een afnemerindicatieblob bevat een soortgelijke opzet als de persoonblob. De structuur is echter eenvoudiger : de blob bevat 1 lijst met blobroots, waarbij elke root een afnemerindicatie representeert. Elk blobrecord bevat afnemerindicatiegegevens en informatie mbt hiërarchische structuur (object en groep waartoe het record behoort).

Datatypering

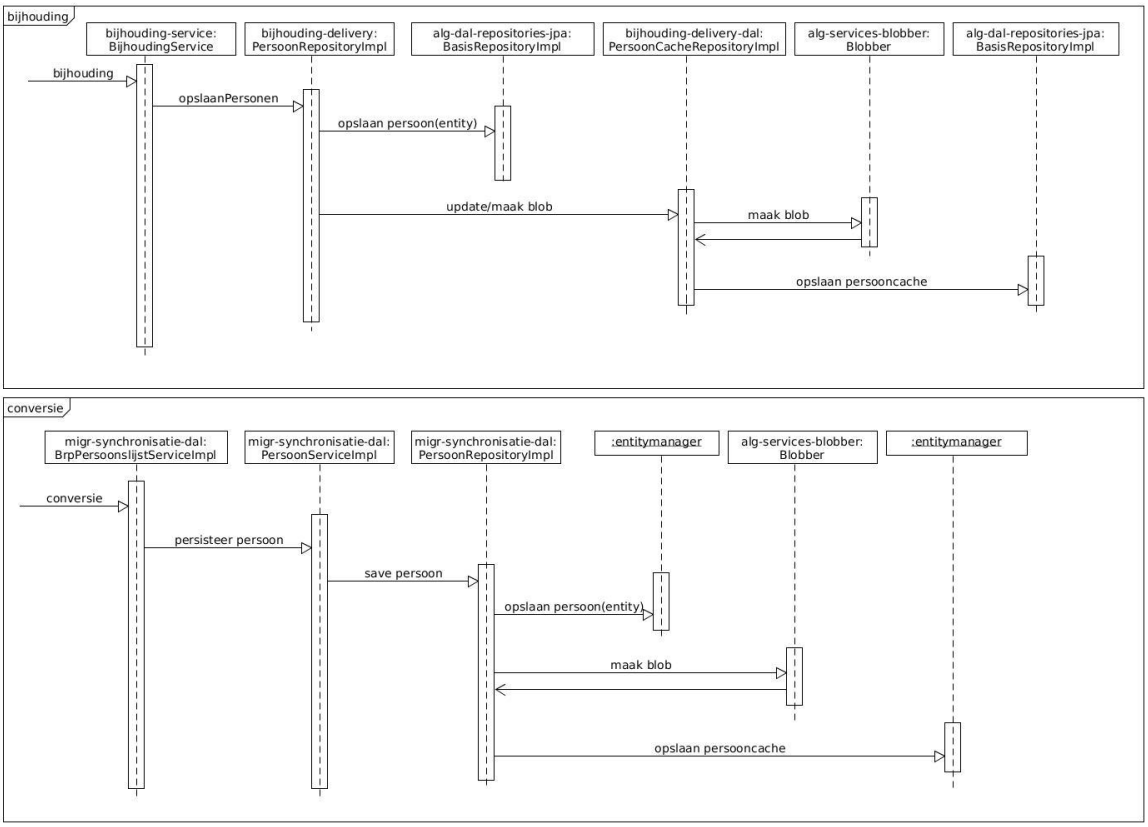
Het datatype van attribuutwaarden zoals gedefinieerd in het databasemodel wordt niet in de blobs opgeslagen. Informatie omtrent datatypering van een attribuut is bij het inladen van het persoonsmodel beschikbaar via het TypeIdentDb attribuut in de elementtabel. Door gebruik te maken van dit mechanisme blijven blobs robuust m.b.t. wijzigingen in datatypering in het databasemodel.

Mapping van stamgegevens

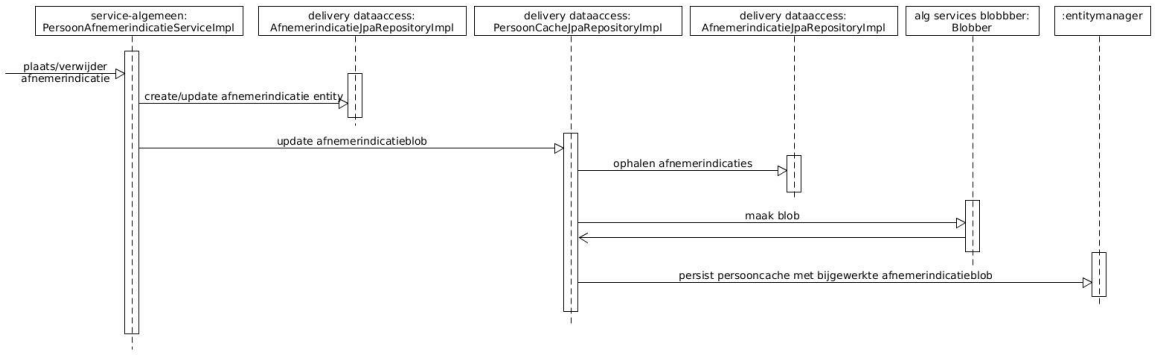
Bij het mappen van attribuutwaarden worden de waarden uit de entities direct ‘as-is’ gekopieerd naar de blob. Een uitzondering hierop vormen de stamgegevens, hier worden de codes i.p.v. de eigenlijke waarden in de database tabel in de blob gezet. Bijvoorbeeld voor het attribuut Persoon.Adres.SoortCode wordt i.p.v. het id direct de code gebruikt.

5.2.2 *Creatie, (de)serialisatie van blobs.*

Creatie, serialisatie en deserialisatie van blobs vindt plaats via de Blobber utility klasse. Deze klasse heeft eveneens een facade-functie, zodat alle overige internals m.b.t. de blob zijn afgeschermd van overige packages. Het bijwerken en opslaan van persoonblobs kan vanuit bijhouding en levering. Zie diagrammen hieronder voor de flows vanuit beide paden.



Afnemerindicatieblobs worden alleen bijgewerkt vanuit levering bij het plaatsen of verwijderen van een afnemerindicatie. De flow is dan alsvolgt:



Creatie

Om de flow van het creëren van blobs duidelijk te maken, bespreken we hier enkel hoe persoonblobs worden gemaakt. Dit illustreert dan ook direct hoe afnemerindicatieblobs worden gecreëerd, aangezien daarvoor dezelfde (maar sterk vereenvoudigde) opzet wordt gehanteerd.

Creatie van persoonblobs gaat a.h.v. een instantie van een Persoon entity. Binnen deze entity instantie zijn alle overige gerelateerde gegevens uit andere entities beschikbaar, die nodig zijn voor de creatie van het complete persoonsbeeld. Het creeren van de persoonblob in de Blobber util gebeurt door sequentueel de PersoonBlobber (t.b.v. constructie van de persoon BlobRoot) en VerantwoordingBlobber (t.b.v. constructie van de lijst met verantwoording BlobRoots) aan te roepen.

De PersoonBlobber maakt het mogelijk blobrecords met persoonsgegevens aan te maken, al dan niet met een specifiek historiepatroon. Afhankelijk van de historievorm worden hier de generieke attribuutwaarden gezet (m.a.w. actieinhoud, actieverval, actieaanpassinggeldigheid, actietbvmutatielevering, nadereaanduidingverval en datum aanvang/einde geldigheid). Tevens wordt hier een actiemap gevuld, die op een later moment gebruikt wordt om de verantwoording blobroots te creeren.

Het daadwerkelijke mappen van entities wordt vanuit de PersoonBlobber constructor gedelegeerd naar een PersoonBlobMapper. In de PersoonBlobMapper worden eerst alle groepen binnen het Persoon meta-object gemapt (Persoon.AfgeleidAdministratief, Persoon.Geboorte, Persoon.Verblijfsrecht etc.). Bijvoorbeeld voor de Persoon.AfgeleidAdministratief groep, wordt over alle PersoonAfgeleidAdministratiefHistorie records geloopt en worden corresponderende blobrecords gecreëerd met alle benodigde groep-specifieke attribuutwaarden, inclusief de benodigde meta-informatie t.b.v. creatie van het metamodel. Vervolgens worden alle objecten binnen het Persoon meta-object gemapt (Persoon.Adres, Persoon.Nationaliteit etc.), inclusief de aanwezige identiteit- en standaardgroepen. Bijvoorbeeld voor het Persoon.Adres object is de flow dan alsvolgt : (1) loop over alle PersoonAdres records, (2) creer voor elk record een blobrecord voor de Persoon.Adres identiteitgroep, (3) haal voor elk persoonadres-record de bijbehorende PersoonAdresHistorie records op en creer voor elk historierecord een blobrecord voor de Persoon.Adres standaard groep.

Na creatie van de Persoon BlobRoot wordt in de Blobber util de VerantwoordingBlobber gebruikt om een reeks verantwoording BlobRoots te creëren, waarbij elke root dezelfde object-hierarchie heeft (Adm. Handeling -> Actie -> ActieBron -> Document. Dit gebeurt a.h.v. de eerder opgebouwde actiemap in de PersoonBlobber.

Serialisatie

Persoonblobs en afnemerindicatieblobs worden in de de database opgeslagen in de perscache tabel. Het serialiseren en deserialiseren gebeurt a.h.v. de Jackson library. De blobs zijn vooralsnog geserialiseerd in JSON-formaat (bytes). Met het oog op performance is het mogelijk dat nog geëxperimenteerd gaat worden met andere binaire formaten zoals SMILE of CBOR.

Synchronisatie operationeel model en blob

Een belangrijk aandachtspunt is het in sync houden van het operationeel model met het metamodel. Het kwetsbare punt is hier de blob : indien wijzigingen in het databasemodel niet verwerkt worden in het mapping mechanisme van de blob, wordt er een incorrect aantal objecten/groepen/attributen gemapt. De BlobTerugConversieTest bewaakt het in sync blijven van de blob met het operationeel model. A.h.v. de actuele elementtabel wordt getest of alle verwachte objecten/groepen/attributen ook daadwerkelijk gemapt zijn in de blob. Omdat aanpassingen in de elementabel altijd i.c.m. een uitvoerig testtraject worden doorgevoerd, zullen synchronisatie-fouten vroegtijdig gesignaleerd worden.

Foutafhandeling

Het gooien van specifieke BlobExceptions is beperkt tot twee situaties :

- bij het deserialiseren van een blob ter ondervanging van IO-excepties die kunnen optreden bij het inlezen van de blob.
- bij serialisatie naar JSON formaat, ter ondervanging van JsonProcessingExceptions.

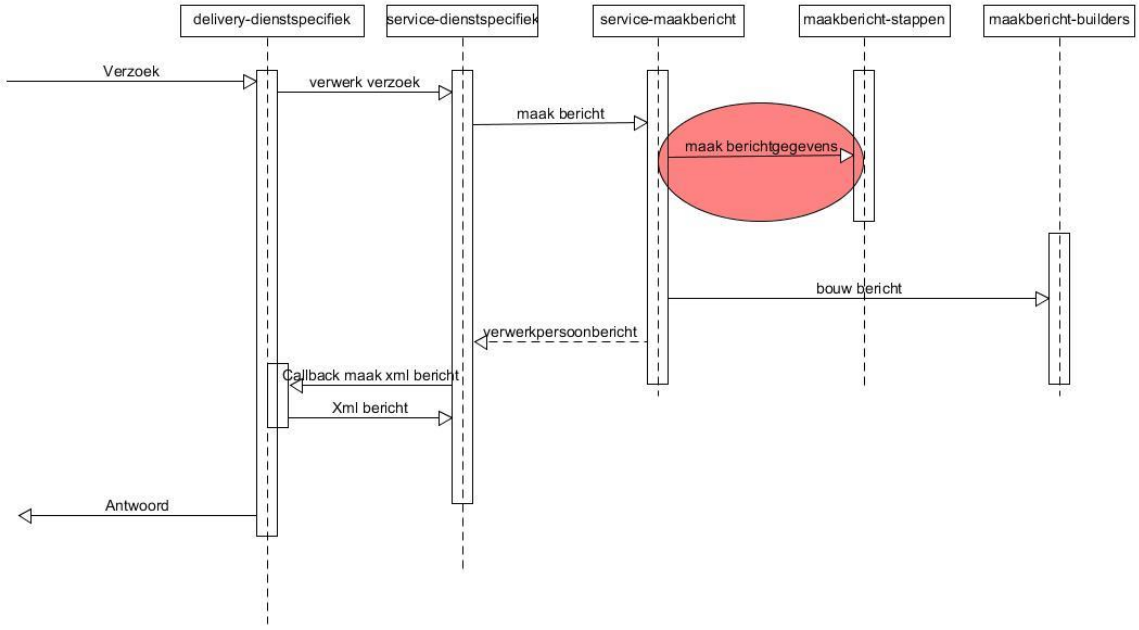
5.2.3 *Constructie persoonslijst o.b.v. blobs.*

Het construeren van een persoonslijst op basis van blobs is geimplementeerd in de PersoonslijstService in service-algemeen. Omdat het niet altijd nodig is om beschikking te hebben over afnemerindicaties, biedt de service mogelijkheid om (1) een persoonslijst inclusief afnemerindicaties te construeren of (2) een lijst persoonslijsten te construeren exclusief afnemerindicaties.

Als eerste stap worden via een persoon-id he t juiste PersoonCache record uit de database opgehaald met de persoon- en afnemerindicatiesblob. De blobs worden gedeserialiseerd en deze worden vervolgens gebruikt om via een factory een Persoonslijst te construeren. In het kader van robuustheid is er een fallback mechanisme ingebouwd, waarbij het persoonsmodel ook geladen kan worden a.h.v. van het genormaliseerde databasemodel. Indien er geen blob aanwezig is in de database, wordt via een query het volledige persoonsbeeld en/of een lijst met bijbehorende afnemerindicaties opgehaald, om deze vervolgens te serialiseren naar een persoon- en/of afnemerindicatieblob. Deze blobs worden vervolgens weer gebruikt in de persoonslijstfactory.

In de factory worden de persoon- en afnemerindicatiesblob via een generieke BlobTerugConverter naar specifieke builders converteerd, die gebruikt worden om een complete persoonslijst zoals gedefinieerd in 3.1.1 te construeren. In dit proces wordt een specifieke VerantwoordingConverter gebruikt om de lijst met verantwoording-blobroots naar een set administratieve handelingen met acties te converteren, zodat de actie-verantwoording (ActieInhoud, ActieVerval etc.) op record niveau gemapt kan worden. Het bepalen van het juiste datatype van elk MetaAttribuut gebeurt eveneens in de generieke BlobTerugConverter middels een AttribuutMapper : op basis van het TypeIdentDb van het attribuutelement wordt voor elke attribuut het juiste type bepaald.

5.3 Maak bericht: stel de te leveren persoonsgegevens samen



De module Maakbericht heeft als doel om het persoonsbericht samen te stellen. Het persoonsbericht wordt uiteindelijk opgenomen in een XML bericht (synchronisatiebericht / bevragingresponse / zoekvraag etc). Maakbericht wordt gekenmerkt door een groot aantal bedrijfsregels welke het uiteindelijke bericht inbeperken. Mag een attribuut/groep/object/verantwoording wel/niet getoond worden etc. De regels verschillen per dienst, per autorisatie en met welke tijdsbeeld er naar de persoonsgegevens gekeken wordt. Daarnaast worden twee typen berichten onderkend, het volledigbericht en het mutatiebericht. Ook hiervoor gelden weer specifieke uitzonderingen.

Maakbericht is in essentie een serie van stappen (chain-of-responsibility), waarbij elke vervolgstap voortborduurt op de bepalingen die eerder gedaan zijn. De volgorde van stappen is daarom belangrijk. Op een aantal detailstappen na is de volgorde vast, De stappen zijn implementaties van de MaakBerichtStap interface en zijn gedefinieerd in een stappenplan. Het startpunt voor de stappen en de interface voor de maakbericht module is de VerwerkPersoonBerichtFactory interface.

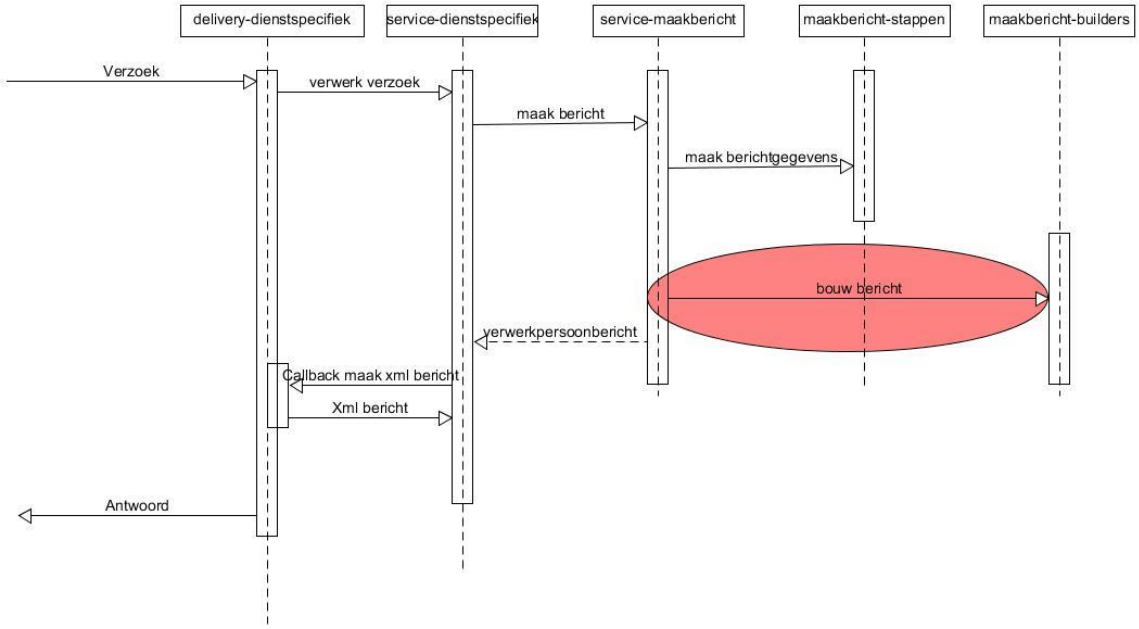
Voor een aantal diensten bestaat specifieke stappen. Deze stappen maken altijd deel uit van het stappenplan, ook als het niet relevant is voor een gegeven dienst. De stap implementatie is zelf verantwoordelijk voor het wel of niet uitvoeren. Veel stappen bevatten zogenoemde short-circuits om alleen uitgevoerd te worden indien relevant.

Maakbericht service wordt aangeroepen met een parameter object: MaakberichtParameters. Dit object bevat informatie die maar voor enige diensten van toepassing is en een verplicht deel wat voor elke dienst gelijk is. Elke dienst moet een persoon en autorisatie als input meegeven aan de maakbericht stappen. Hiernaast kan bijvoorbeeld het beeld worden ingeperkt door een delta berekening op basis van een administratieve handeling of op basis van een peilmoment.

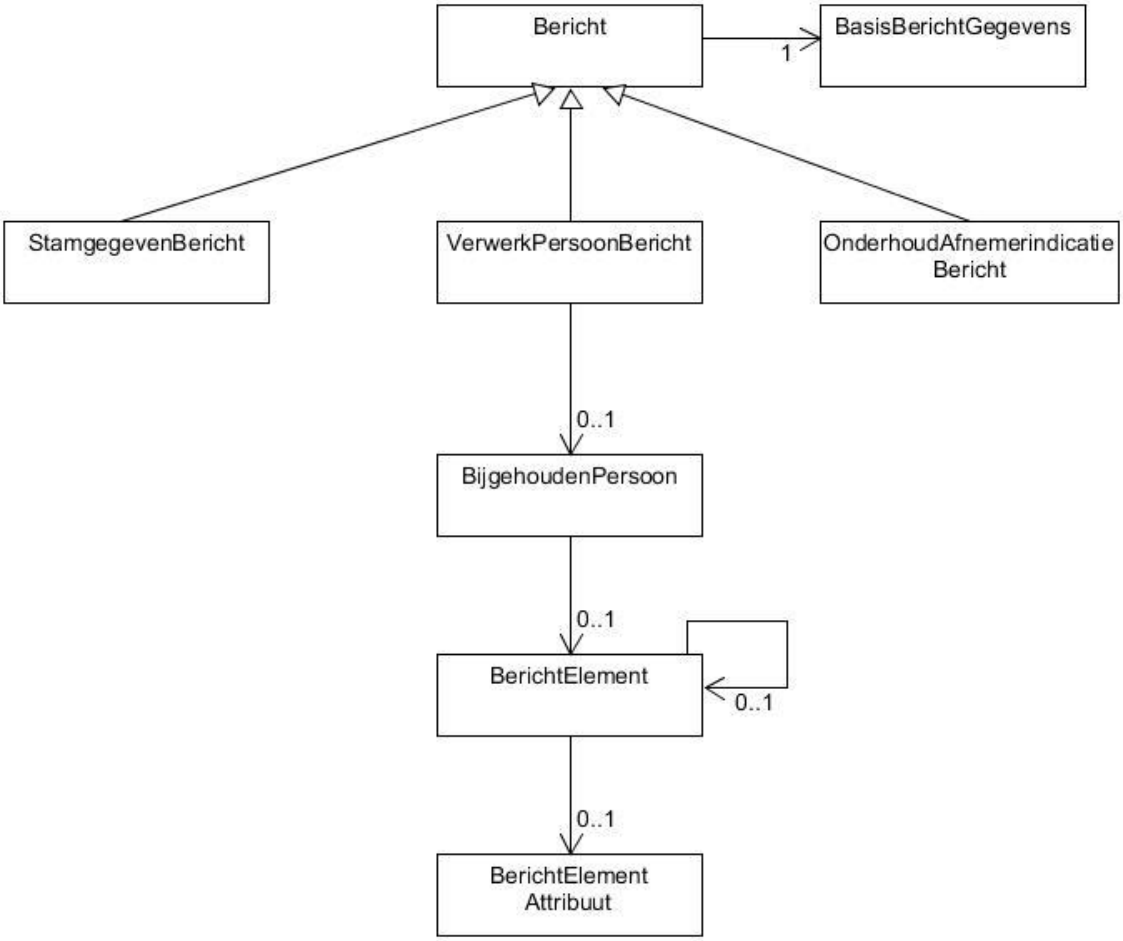
De verschillende stappen bouwen een set van metamodel objecten op waarmee in de maakberichtbuilders een berichten structuur gemaakt wordt die door de aanroepende dienst gebruikt kan worden om een xml bericht te maken. De output van de maakbericht stappen is een gevuld berichtgegevens object wat de geautoriseerde metobjecten bevat, met verwerkingsoort informatie en de geautoriseerde acties.

5.4 Maak bericht model

5.4.1 *Overzicht flow*



5.4.2 *Overzicht van persoon berichtmodel*


Bericht
BasisBerichtGegevens
1
StamgegevenBericht
VerwerkPersoonBericht
OnderhoudAfnemerindicatie Bericht
0..1
BijgehoudenPersoon
0..1
BerichtElement
0..1
0..1
BerichtElement Attribuut

6 Caches

Binnen leveren is zijn autorisaties, partijen en stamgegevens gecached. Deze worden automatisch gefreshed op een configureerbaar tijdstip via de volgende property: brp.levering.cache.cron. Ook kunnen de caches via JMX ververst worden. Uiteindelijk zal dit via een topic/subscribe mechanisme moeten gaan lopen.

Bij het inladen van autorisaties worden alleen autorisaties opgehaald die een datum einde hebben van maximaal 3 maanden in het verleden. Dienstbundels die niet volledig geconverteerd zijn worden ook niet ingeladen.

Ook worden autorisatie delen uitgesloten die ongeldige expressies bevatten.

Informatie uit de element tabel is ook gecached. Maar deze is runtime niet verversbaar. Deze is statisch beschikbaar en wordt gemaakt in het build proces.