vektis streams

Standaardisatie

EI-standaarden

Architectuurprincipes

Versie document	2.1
Versie datum	23-2-2022
Uitgave document	1
Uitgave datum:	23-2-2022
Kenmerk:	Architectuur EI-standaarden v2.1_u1.docx

Contact

Vektis
Postbus 703
3700 AS ZEIST

Bezoekadres
Vektis
Sparrenheuvel 18
3708 JE ZEIST

Telefoon: 030 – 8008 300

Helpdesk: standaardisatie@vektis.nl
Website: www.vektis.nl
Informatie over standaarden: www.vektis.nl/standaardisatie

De inhoud van de Vektis standaardisatie documentatie is met uiterste zorgvuldigheid tot stand gebracht. De inhoud wordt regelmatig gecontroleerd en geactualiseerd. Vektis kan echter niet aansprakelijk worden gesteld voor de juistheid, volledigheid en actualiteit van de website. Vektis is niet aansprakelijk voor eventuele schade of consequenties ontstaan door direct of indirect gebruik van de inhoud van de documentatie.

Informatie uit deze documentatie mag je overnemen mits je daarbij de bron vermeldt.

Inhoudsopgave

1. Inleiding

Dit document beschrijft de architectuurprincipes die Vektis hanteert bij het opstellen van EI-standaarden. Dit document is een levend document, dat wil zeggen dat het toepassen van de principes kan leiden tot nieuwe inzichten. Deze nieuwe inzichten kunnen leiden tot een nieuwe versie van dit architectuur document. Wijzigingen ten opzichte van voorgaande versies zijn opgenomen in de bijlagen (5.1). Aangepaste architectuurprincipes worden van kracht in standaarden die na het doorvoeren van de wijziging worden opgesteld. In elke EI-standaard is een verwijzing opgenomen naar de versie van de architectuurprincipes die is gebruikt bij het opstellen van de standaard.

1.1. Reikwijdte EI-standaarden

Bij het opstellen van de architectuurprincipes is rekening gehouden met de reikwijdte van de EI-standaarden.

Op het moment van opstellen van dit document, ondersteunen de EI-standaarden elektronische informatie-uitwisseling die volgt uit de volgende wetgeving:

- Zorgverzekeringswet (Zvw);
- Wet langdurige zorg (Wlz);
- Wet forensische zorg (FZ);
- Wet maatschappelijke ondersteuning (Wmo);

Binnen de informatie-uitwisseling die uit deze wetten volgt, onderscheiden we de volgende hoofdgroepen (domeinen):

- declaraties;
- schade-informatie;
- EESSI-berichten (Electronic Exchange of Social Security Information, betreft grensoverschrijdende berichten)
- overige berichten (bijvoorbeeld fraudesignalen, controles op verzekeringsrecht en rapportages).

Voor elk van deze domeinen geldt dat de berichten van toepassing kunnen zijn op meerdere zorgsoorten (huisartsenzorg, medisch specialistische zorg, geestelijke gezondheidszorg, farmacie, hulpmiddelenzorg, etc.).

1.2. Doelen architectuur

Met de principes die in dit document staan beschreven, willen we de volgende doelen bereiken:

- Uniforme oplossingen over alle domeinen en zorgsoorten heen
  Binnen verschillende domeinen en zorgsoorten, zien we vaak dezelfde vraagstukken terugkomen. Voor deze vraagstukken willen we graag een uniforme oplossing die over alle EI-standaarden heen geldt. Dit zorgt ervoor dat we bij het opstellen van een nieuwe standaard niet steeds opnieuw een oplossing hoeven te bedenken. Daarnaast hoeven partijen die meerderen EI-standaarden ondersteunen ook maar één oplossing te implementeren die dan voor meerdere standaarden kan worden gebruikt. Om dit doel te bereiken, gaan de architectuurprincipes steeds uit van de herbruikbaarheid van gegevens.
- Privacy by design
  Er is steeds meer aandacht voor het belang van de privacy van de patiënt, in het bijzonder met betrekking tot de digitale uitwisseling van de privacygevoelige gegevens. Vanuit de EI-standaarden willen we bijdragen aan dit belang door ervoor te zorgen dat alleen de gegevens die echt nodig zijn, zijn opgenomen in de berichten. Om dit doel te bereiken, gaan de architectuurprincipes steeds uit van dataminimalisatie. Dat wil zeggen, berichten bevatten alleen de minimaal benodigde informatie en er wordt zoveel mogelijk gebruik gemaakt van verwijzingen naar bronnen. Bijvoorbeeld door enkel het opnemen van de AGB-code van een zorgaanbieder, waarbij aanvullende informatie uit het AGB-register moet worden opgehaald.
- Aansluiting bij andere informatiestandaarden in de medische IT
  Binnen de medische IT komen steeds meer informatiestandaarden met als doel eenduidige vastlegging en uitwisseling van medische gegevens. Omdat de EI-standaarden medische informatie bevatten en berichten vaak vanuit medische informatiesystemen worden verstuurd, willen we vanuit de EI-standaarden waar mogelijk aansluiten op de beschikbare informatiestandaarden. De architectuurprincipes gaan daarom uit van het gebruik van zorginformatiebouwstenen (ZIB's) (zoals beschreven in paragraaf 2.5).

1.3. Leeswijzer

Hoofdstuk 2 beschrijft alle architectuur principes die we hanteren bij het inhoudelijk opstellen van berichten. De architectuur principes in dit hoofdstuk zijn zoveel mogelijk techniek onafhankelijk beschreven.

Hoofdstuk 3 beschrijft hoe deze algemene architectuur principes worden gebruikt bij het opstellen van EI-standaarden in XML-formaat. Daarnaast beschrijft dit hoofdstuk nog een aantal XML-specifieke principes.

1.4. Documentstatus

Versie	Datum	Status	Omschrijving
2.1	23-2-2022	definitief	Versiebeheer beschreven.
2.0	15-3-2021	definitief	Tweede versie, vervangt XML Schema Definition Architectuurprincipes (v1.3).

1.5. Openstaande punten

De volgende punten moeten nog worden opgenomen in toekomstige versies van dit document:

- Architectuurprincipes met betrekking tot het opstellen en beheer van codelijsten.
- Architectuurprincipes met betrekking tot het uitvoeren van controles.
- Architectuurprincipes die gelden bij het opstellen van XSLT's.
- Eventueel de principes achter de standaard processen binnen de EI-standaarden zoals de berichtroutering, retoursystematiek en crediteren (in hoofdlijnen, gedetailleerde beschrijvingen blijven in de Standaardbeschrijving en Invulinstructie staan).

2. Architectuur bericht (algemeen)

Dit hoofdstuk beschrijft de algemene architectuur principes die Vektis hanteert voor het opstellen van berichten voor de EI-standaarden.

Nieuwe berichten worden zoveel mogelijk opgezet in XML. Bij het opstellen van de algemene architectuur van het bericht is dan ook rekening gehouden met de mogelijkheden die XML biedt. De principes zijn echter zo opgezet dat ze ook toepasbaar zijn bij andere technische invullingen van het bericht (mits de techniek de mogelijkheden heeft om het principe toe te passen), denk hierbij bijvoorbeeld aan ASCII berichten.

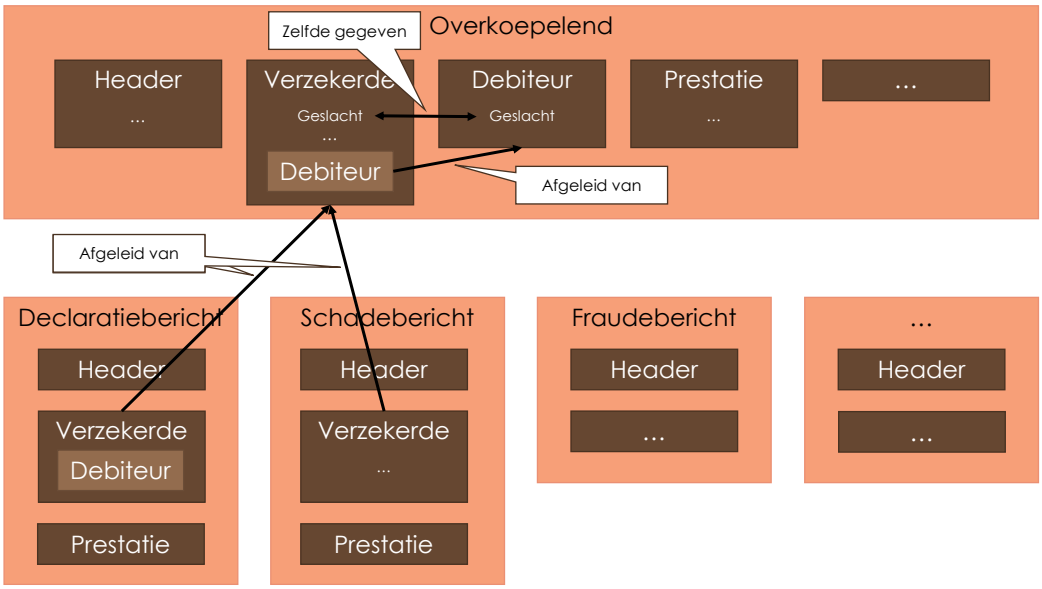
2.1. Generiek opstellen berichten

Vektis wil over alle berichten heen consequent dezelfde oplossingen en structuren gebruiken. Om dit te bereiken, worden berichten samengesteld uit generieke herbruikbare gegevens en gegevensblokken. Deze paragraaf beschrijft de principes die Vektis hanteert voor het opstellen van de generieke gegevens(blokken) en hoe uit deze blokken een bericht kan worden opgesteld.

In hoofdlijnen houden deze principes in dat álle gegevens op een overkoepelend niveau (de bouwdoos) worden gedefinieerd (de generieke gegevens). Deze gegevens worden op dit overkoepelende niveau gegroepeerd in gegevensblokken (de generieke gegevensblokken). Hierbij is het mogelijk dat een gegeven in meerdere blokken terugkomt en ook dat gegevensblokken andere gegevensblokken kunnen bevatten.

Bij het opstellen van berichten worden uit dit overkoepelende niveau de gegevensblokken gepakt die nodig zijn voor het bericht. Een bericht kan dus nooit een gegeven bevatten dat niet in het overkoepelende niveau is opgenomen.
Omdat het gebruik (en vooral het verplicht zijn) van een gegeven nooit over alle EI-standaarden hetzelfde is, is het niet verplicht om het gehele gegevensblok te gebruiken maar kunnen onderdelen van het blok worden gebruikt. Een gegevensblok in een bericht is daarmee altijd een afgeleide van het generieke gegevensblok.

Onderstaande afbeelding toont een samengesteld voorbeeld van deze principes. Hierin is het resultaat van de principes weergegeven voor de verzekerde van een bericht. De header, prestatie, etc. kunnen op eenzelfde manier worden opgebouwd.



NB: Deze paragraaf beschrijft alleen de principes over het generiek opstellen van gegevens(blokken) en berichten. Principes die gaan over het inhoudelijk opstellen van berichten, gegevensblokken, etc. staan beschreven in paragraaf 2.2 en verder.

2.1.1. Definiëren generieke gegevens(blokken)

Voor het opstellen van de generiek herbruikbare gegevens, hanteert Vektis onderstaande principes.

1.	Gegevens(blokken) worden altijd overkoepelend gedefinieerd. Dit wil zeggen dat er één overkoepelende 'bouwdoos' is waarin alle gegevens en gegevensblokken zijn gedefinieerd die in de berichten voorkomen (dus ook als er maar één bericht is waarvoor ze relevant zijn). Toelichting: Door één plek te hebben waar alle gegevens zijn gedefinieerd, kan overzicht worden behouden over de gegevens die we al kennen. Bij het opstellen van standaarden voor nieuwe berichten bevordert dit het hergebruik van bestaande gegevens.
2.	In de bouwdoos kunnen losse gegevens worden gedefinieerd.
3.	In de bouwdoos kunnen gegevensblokken worden gedefinieerd. Deze gegevensblokken kunnen bestaan uit verschillende losse gegevens en/of andere gegevensblokken uit de bouwdoos.
4.	Een gegevensblok in de bouwdoos beschrijft alleen uit welke gegevens(blokken) het mogelijk kan bestaan. We beschrijven in de bouwdoos dus **niet** hoe gegevens(blokken) kunnen/moeten voorkomen binnen het gegevensblok. Toelichting: Het komt zeer zelden voor dat een gegeven binnen alle berichten moet worden gebruikt. Per bericht zal daarom toch gekeken moeten worden of een gegeven(sblok) verplicht of optioneel is en hoe vaak het kan voorkomen. Het vastleggen hiervan binnen de bouwdoos heeft daarom weinig waarde.
5.	Gegevens en gegevensblokken zijn herbruikbaar. Dit wil zeggen dat losse gegevens en gegevensblokken kunnen worden gebruikt in meerdere gegevensblokken. Wanneer een gegeven(sblok) in meerdere gegevensblokken voorkomt, wordt in elk blok dezelfde definitie van het gebruikte gegeven(sblok) gehanteerd. Voorbeeld: De gegevensblokken AanvullendeVerzekerdengegevens en Debiteur bevatten allebei een gegevensblok Naamgegevens. In de AanvullendeVerzekerdengegevens en de Debiteur wordt hiervoor hetzelfde gegevensblok gebruikt.

	Toelichting: Het doel van dit principe is om over alle standaarden heen met dezelfde definities te werken. We willen voorkomen dat er binnen berichten en/of gegevensblokken het zelfde gegeven(sblok) wordt genoemd, terwijl er iets anders wordt bedoeld.
6.	Wanneer een gegevensblok (A) een ander gegevensblok (B) bevat, dan hoeven in gegevensblok A niet alle onderdelen uit gegevensblok B te zijn opgenomen maar kan ook een subset worden gebruikt. (Gegevensblok B kan binnen gegevensblok A niet solitair worden uitgebreid. Extra gegevens moeten eerst generiek in gegevensblok B worden toegevoegd.) Voorbeeld: De gegevensblokken DeclaratieContext en DebetPrestatie bevatten allebei een gegevensblok Zorgaanbieder. Binnen de DeclaratieContext bevat dit blok alleen de gegevens ZorgaanbiederscodeenZorgaanbiederSoort. Binnen de DebetPrestatie bevat dit blok ook de gegevens(blokken) NaamZorgverlener, ZorgaanbiederSpecificatie en ZorgaanbiederRol. Toelichting: We willen voorkomen dat er een wildgroei ontstaat aan varianten op hetzelfde gegevensblok. Dit zou namelijk betekenen dat er niet meer een generiek concept is gedefinieerd, maar dat er alsnog situatie specifieke gegevensblokken nodig zijn. Dit gaat in tegen het principe van herbruikbaarheid.
7.	De gegevens(blokken) zijn zo gedefinieerd dat ze domein en/of zorgsoort overstijgend zijn. Dit wil zeggen dat de definitie het concept of proces uitdraagt dat ten grondslag ligt aan het gegeven. Voorbeeld: In het zorgprestatiemodel van de GGZ wordt gesproken over een zorglabel. Dit zorglabel geeft aanvullende informatie over een prestatie. In het declaratiebericht is dit zorglabel opgenomen als AanvullendPrestatieKenmerk. Toelichting: Met name voor de verwerkers die meerdere soorten berichten kunnen ontvangen, is het belangrijk dat gegevens die hetzelfde doel dienen steeds op dezelfde manier worden aangeleverd. Hierdoor wordt het eenvoudiger voor hen om een nieuwe standaard te implementeren.

	Verder helpt dit principe bij het voorkomen van een wildgroei aan verschillende oplossingen voor hetzelfde probleem, ondersteunt het consistentie over verschillende domeinen heen en houdt het berichtdefinities waarin verschillende zorgsoorten kunnen worden opgenomen compact (in plaats van voor elke zorgsoort een apart veld opnemen, volstaat één veld).
8.	Zorgsoort specifieke informatie wordt vastgelegd in de gegevensblokken die de zorg beschrijven. Voorbeeld: Met betrekking tot declaraties wordt de zorgsoort specifieke informatie vastgelegd bij de prestatie. Toelichting: Door de zorgsoort specifieke informatie vast te leggen bij het gegevensblok dat de zorg beschrijft, kunnen andere gegevensblokken eenvoudiger zorgsoort en domein overschrijdend worden gebruikt.

2.1.2. Opstellen bericht uit generieke gegevensblokken

Voor het opstellen van een bericht vanuit de generieke gegevensblokken, hanteert Vektis onderstaande principes.

9.	Berichten worden opgebouwd uit gegevensblokken die in de bouwdoos zijn gedefinieerd. Toelichting: Vanuit het oogpunt van herbruikbaarheid leggen we alle gegevensblokken generiek vast, ook als deze mogelijk maar in één standaard worden gebruikt. Hiermee zorgen we dat gegevensblokken altijd op dezelfde manier terug komen.
10.	Bij het opnemen van een gegevensblok in een bericht hoeven niet alle losse gegevens en sub-gegevensblokken van dat gegevensblok te worden opgenomen. Het is dus mogelijk om een subset van een gegevensblok op te nemen in een bericht. Voorbeeld: Zowel in het declaratiebericht als het bericht voor verantwoording forensische zorg is het gegevensblok Plaatsingsbesluit opgenomen. Voor de verantwoording

	forensische zorg zijn meer gegevens van het plaatsingsbesluit relevant dan voor de declaratie. In het declaratiebericht zijn deze extra gegevens niet opgenomen in het plaatsingsbesluit. Toelichting: Vanuit het oogpunt van dataminimalisatie willen we dat berichten alleen de gegevens bevatten die daadwerkelijk nodig zijn voor de verwerking van het bericht.
11.	Net als binnen gegevensblokken (zie principe 6) is het binnen een bericht **niet** mogelijk om een gegevensblok uit te breiden. Wanneer voor een bericht extra gegevens moeten worden toegevoegd, moet dit altijd in het gegevensblok in de bouwdoos worden gedaan.
12.	Bij gebruik van een gegevensblok in een bericht, beschrijven we **wel** hoe gegevens(blokken) kunnen/moeten voorkomen binnen het gegevensblok in dat bericht. Voorbeeld: Het is niet in alle berichten verplicht een gegevensblok voor een prestatie op te nemen. Het is in de bouwdoos daarom niet mogelijk om voor het gegevensblok Prestatie vast te stellen of het verplicht is. Binnen declaratieberichten is het echter altijd verplicht om een prestatie op te nemen. Binnen declaratieberichten kan het gegevensblok Prestatie daarom wel als verplicht worden gesteld. Toelichting: Binnen een bericht kunnen gegevens(blokken) aanwezig zijn die verplicht moeten worden opgenomen. Dit is een essentieel verschil met de generieke gegevensblokken in de bouwdoos.
13.	Als uitgangspunt moet bij het opstellen van een bericht worden onderzocht of er ook andere berichten zijn die hetzelfde doel dienen en of het mogelijk is deze verschillende berichten onder te brengen in één berichtdefinitie. Voorbeeld: Er kunnen vanuit verschillende zorgsoorten declaratieberichten worden verstuurd. Al deze berichten hebben als doel de geleverde zorg bij de debiteur te declareren. In plaats van een declaratiebericht per zorgsoort op te stellen, kan daarom beter één generiek declaratiebericht worden opgesteld.

	Toelichting: Het gebruik van één berichtdefinitie garandeert uniformiteit vanuit de verschillende invalshoeken. Voor indieners waarvoor meerdere zorgsoorten van toepassing zijn, geeft dit ook de mogelijkheid om hun gegevens in te dienen via één bericht (in plaats van een bericht per berichtsoort).
14.	Wanneer meerdere berichten zijn ondergebracht in één berichtdefinitie, zijn gegevens(blokken) alleen verplicht als dit geldt voor alle berichten die worden afgedekt in de berichtdefinitie. Als dit niet het geval is, zijn de gegevens(blokken) optioneel opgenomen.

2.2. Opbouw van het bericht

Voor de inhoudelijke opbouw van een bericht, hanteert Vektis onderstaande principes.

1.	Elk bericht begint met een header. De header dient de volgende doelen: • aangeven om welk bericht het gaat; • informatie geven over verzender en ontvanger van het bericht.
2.	Naast de header kan nog een gegevensblok worden opgenomen met berichtdefinitie specifieke informatie die voor het hele bericht geldt.
3.	Berichten zijn opgebouwd volgens een boomstructuur. Dit wil zeggen, na de header en bericht specifieke informatie volgt een (repeterend) gegevensblok. Binnen dit repeterende gegevensblok kunnen weer andere repeterende gegevensblokken voorkomen (die betrekking hebben op het gegevensblok waar ze in voorkomen), enzovoorts.
4.	Elementen die over de tijd heen vaker kunnen voorkomen, worden in het bericht op de juiste plaats in de boomstructuur gezet. Voorbeeld: Bij declaraties in de forensische zorg moet een plaatsingsbesluit worden opgegeven. Het plaatsingsbesluit wordt afgegeven aan een verzekerde. Het is echter mogelijk dat een verzekerde over de tijd heen meerdere plaatsingsbesluiten heeft. In één declaratie kunnen prestaties worden gedeclareerd die vallen onder verschillende plaatsingsbesluiten. In het declaratiebericht is het plaatsingsbesluit daarom opgenomen bij de prestatie en niet bij de verzekerde.
5.	Wanneer in een bericht binnen een gegevensblok een subgegevensblok is opgenomen, dan wordt enkel de verwijzing naar het subgegevensblok

	opgenomen met als type het gegevensblok dat wordt gebruikt. Dit gegevensblok staat verderop in de berichtspecificatie uitgewerkt. Aandachtspunt hierbij is wel dat er geen cirkeldefinities ontstaan. Toelichting: Omdat de berichtspecificatie een statisch document is, willen we voor het overzicht niet de gehele boomstructuur uitwerken. Daarnaast hoeven de gegevens maar één keer te worden beschreven als het gegevensblok vaker voorkomt als subgegevensblok.

2.3. Opbouw gegevensblokken

Voor de inhoudelijke opbouw van gegevensblokken, hanteert Vektis onderstaande principes.

1.	Van elk gegevensblok worden generiek de volgende kenmerken vastgelegd: • naam; • indien het gegevensblok ZIB's gebruikt, het ZIB-ID en de versie van de gebruikte ZIB's; • definitie; • uit welke losse gegevens en gegevensblokken het blok bestaat.
2.	Van elk gegevensblok worden op berichtniveau de volgende extra kenmerken vastgelegd (naast de generieke kenmerken): • voorkomen (cardinaliteit, deze staat bij de aanroep van het blok of in het klassendiagram); • toelichting met daarin een korte invulinstructie met condities en constraints (beperkingen).
3.	Losse gegevens worden alleen gegroepeerd in een gegevensblok wanneer dit een doel dient. We groeperen losse gegevens niet enkel om het groeperen. Doelen om gegevens te groeperen kunnen zijn: • Hetzelfde groepje gegevens moet vaker voor kunnen komen (binnen een bericht of gegevensblok, maar ook daarbuiten). • Er zijn controles die gelden voor alle gegevens in het groepje. • De gegevens behoren functioneel bij elkaar en groeperen bevordert de leesbaarheid van een bericht of gegevensblok.
4.	Een gegevensblok begint altijd met de gegevens(blokken) die in alle toepassingen van het gegevensblok van belang zijn (ook al zijn ze eventueel optioneel). Daarna

	volgt een subgegevensblok waarin gegevens(blokken) zijn opgenomen die niet voor alle toepassingen van het gegevensblok een betekenis hebben. Voorbeeld: Voor declaraties zijn binnen de prestatie gegevens zoals de prestatiecodelijst en prestatiedatum altijd van belang. Deze gegevens staan dan ook aan het begin van de prestatie opgenomen. De diagnose is niet voor alle soorten prestaties van belang. De diagnose is daarom opgenomen in het subgegevensblok 'AanvullendeGegevens'. Toelichting: Wanneer een gegevensblok meerdere toepassingen heeft, wordt zo eenvoudig duidelijk gemaakt welke gegevens voor alle toepassingen relevant zijn en bij welke gegevens moet worden gekeken of ze relevantie hebben voor de specifieke toepassing.

2.4. Opbouw losse gegevens

Voor de opbouw van losse gegevens, hanteert Vektis onderstaande principes.

1.	Generiek worden van elk los gegeven de volgende kenmerken vastgelegd: • naam; • indien het een gegeven uit een ZIB betreft, het ZIB-ID en de versie; • type, zie ook principe 4; • definitie; • toelichting met daarin een korte invulinstructie met condities en beperkingen en een bronverwijzing bij gebruik van een codelijst (voor zover dit allemaal vanuit generiek oogpunt mogelijk is).
2.	Op berichtniveau worden van elk los gegeven de volgende extra kenmerken vastgelegd (naast de generieke kenmerken): • voorkomen (cardinaliteit); • aanvullende toelichting op de generieke toelichting met daarin een korte invulinstructie met condities en beperkingen en een bronverwijzing bij gebruik van een codelijst zoals deze voor het bericht gelden.
3.	Uitgangspunt is dat gegevens geen lengte restrictie krijgen (tenzij het voor de gebruikte techniek noodzakelijk is een lengte te definiëren). Dit geldt ook voor codes.

	Dit betekent dat de ontvanger lange waarden af moet kappen op de beschikbare lengte in zijn systeem. De zender dient hier rekening mee te houden (informatie die aan het einde van een lange waarde is opgenomen, is mogelijk niet beschikbaar voor de ontvanger). Toelichting: Ook voor codes die een vaste lengte hebben, leggen we geen lengte vast. Dit kan bijvoorbeeld zijn als vanuit de gebruikte codelijst een vaste lengte hebben. Zo zorgen we ervoor dat, mocht de lengte van de code in de toekomst wijzigen, dit geen invloed heeft op de standaard. In de invulinstructie geven we wel aan dat de lengte zoals gedefinieerd in de codelijst gehanteerd dient te worden, dus inclusief het aantal voorloopnullen dat de codelijst voorschrijft.
4.	De volgende typen worden gebruikt: • reeks karakters (string) • alfanumeriek (string beperkt tot [a-z][A-Z][0-9]); • alfanumeriek in hoofdletters (string beperkt tot [A-Z][0-9]) • geheel getal (integer); • numeriek (string beperkt tot [0-9], wordt gebruikt bij codelijsten die voorloopnullen hanteren); • decimaal getal (decimal); • ja/nee (boolean); • datum (date); • tijd (time, wordt opgenomen inclusief timezone); • datum en tijd (datetime, wordt opgenomen inclusief tijdzone); • object (base64binary, kan bijvoorbeeld worden gebruikt voor afbeeldingen).
5.	Gegevens die een bedrag representeren zijn altijd van het type decimaal getal.
6.	Gegevens die een bedrag representeren bevatten alleen positieve waarden (>=0). Het ‘teken’ van het bedrag kan worden aangegeven via een los gegeven dat aangeeft of het debet of credit betreft.
7.	Gegevens die een aantal representeren zijn van het type geheel getal. Via een gegeven dat de eenheid van het aantal representeert, kan het aantal worden geschaald.
8.	Gegevens die een aantal representeren bevatten alleen positieve waarden (>=0). Het ‘teken’ van het aantal kan worden aangegeven via een los gegeven dat aangeeft of het debet of credit betreft. In gevallen waar het niet logisch is om negatieve aantallen via een debet/credit indicatie aan te geven, kan hiervan worden afgeweken.

	<u>Toelichting:</u> Om verwarrende situaties te voorkomen, is het belangrijk dat de debet-credit indicatie eenduidig is opgenomen en er niet meerdere indicaties zijn die elkaar tegen kunnen spreken.
9.	Gegevens die als waarde altijd ja/nee (of waar/onwaar) bevatten, zijn van het type ja/nee (boolean). Hierbij moet wel worden opgelet of het gegeven niet 'ja', 'nee', 'onbekend' moet bevatten, in dat geval geldt het type ja/nee niet.
10.	Het type 'datum en tijd' wordt alleen gebruikt voor gegevens die **altijd** een datum en tijd bevatten. Als dit niet altijd het geval is, wordt een los datum gegeven en/of een los tijd gegeven opgenomen. <u>Voorbeeld:</u> Voor sommige zorgsoorten is bij een prestatie alleen de datum van de prestatie relevant, terwijl voor andere zorgsoorten zowel de datum als de tijd van belang zijn. Voor prestatie worden daarom een losse prestatiedatum en prestatie tijd opgenomen. <u>Toelichting:</u> Wanneer in een bepaalde situatie enkel de datum relevant is, moet in een 'datum en tijd' gegeven een dummywaarde voor de tijd worden gebruikt. Het gebruik van dummywaarden is niet gewenst. Wanneer afhankelijk van de situatie een datum, en/of tijd van belang is bij een gegeven, moet dit gegeven daarom worden opgesplitst in een datum gegeven en een tijd gegeven. Afhankelijk van de situatie wordt dan het datum en/of tijdgegeven gevuld.
11.	Bij het toevoegen van een nieuw gegeven geldt als uitgangspunt dat het type gelijk is aan het type van functioneel vergelijkbare gegevens.
12.	Het type van een gegeven past altijd zo zuiver mogelijk bij de waarden die het gegeven kan bevatten. (Gegevens die alleen gehele getallen kunnen bevatten, krijgen het type geheel getal en niet alfanumeriek, etc.)

2.5. Gebruik maken van ZIB's

Binnen medische informatiesystemen wordt steeds meer gebruik gemaakt van zorginformatiebouwstenen (ZIB's). Zorginformatiebouwstenen zijn informatiemodellen die worden gebruikt om inhoudelijke (niet technische) afspraken vast te leggen ten behoeve van het standaardiseren van informatie, die gebruikt wordt in het zorgproces.

Om het eenvoudiger te maken voor de medische informatiesystemen om aan te sluiten op de EI-standaarden van Vektis, wordt bij het opstellen van generieke gegevensblokken waar mogelijk gebruik gemaakt van de ZIB's. Deze paragraaf legt uit hoe Vektis dat doet.

Meer informatie over ZIB's is beschikbaar op de volgende pagina:
https://zibs.nl/wiki/ZIB_Hoofdpagina

Opmerkingen:

- ZIB's worden gebruikt bij het opstellen van generieke gegevensblokken. Voor het gebruik van deze gegevensblokken in berichten gelden de principes zoals beschreven in paragraaf 2.1.2. Dat er een ZIB is gebruikt voor het blok heeft hier dus geen invloed op.
- ZIB's kunnen worden gezien als generieke gegevensblokken die door een externe partij zijn gedefinieerd. In de principes die worden gehanteerd voor het gebruik van ZIB's, wordt op sommige punten afgeweken van hoe er met gegevensblokken wordt omgaan die Vektis zelf heeft opgesteld. De reden hiervoor is dat de ZIB's zijn bedoeld om medische informatie uit te wisselen. Omdat de EI-standaarden alleen de noodzakelijke medische informatie moeten bevatten, wordt niet altijd vastgehouden aan de structuur uit de ZIB's.
- Mogelijk gaat in de toekomst ook gebruik worden gemaakt van andere extern gedefinieerde gegevensblokken. Het uitgangspunt is dat voor deze gegevensblokken dan dezelfde principes worden gehanteerd als voor de ZIB's.

2.5.1. Wanneer gebruik maken van een ZIB

Onderstaande principes beschrijven wanneer een ZIB moet worden toegepast bij het opstellen van een generiek gegevensblok.

13.	Uitgangspunt is om bij het opstellen van generieke gegevensblokken zo veel mogelijk gebruik te maken van de ZIB's.
14.	Gebruik een ZIB in een gegevensblok als het concept van de ZIB toepasbaar is op een deel van de gegevens die zijn opgenomen in het gegevensblok.

	Voorbeeld: Bij de ZIB Patiënt staat het volgende concept: “Een persoon die medische, psychische, paramedische of verpleegkundige zorg ontvangt. In sommige zorgsectoren wordt in plaats van de term patiënt de term cliënt of deelnemer gebruikt.” Binnen de declaratiestandaard is dit concept van toepassing op een verzekerde. Daarom worden alle relevante patiëntgegevens binnen de verzekerde opgenomen zoals deze in de ZIB Patiënt zijn gedefinieerd.
15.	Het is niet nodig de volledige ZIB te integreren in een gegevensblok. Er kan dus ook gebruik worden gemaakt van losse onderdelen van een ZIB. Toelichting: Vanuit het oogpunt van dataminimalisatie willen we alleen gegevens versturen die daadwerkelijk van belang zijn voor de EI-standaard.
16.	Gebruik losse onderdelen uit een ZIB als de definitie van het onderdeel overeenkomt met het gegeven dat in het gegevensblok moet worden opgenomen. Toelichting: Als afwijken op definitie nodig is, dan bevat de ZIB eigenlijk niet het gegeven dat nodig is voor de standaard. Wanneer in dit geval toch gebruik gemaakt zou worden van het onderdeel van de ZIB, zou dit verwarring kunnen veroorzaken met als gevolg dat berichten niet de juiste gegevens bevatten.

2.5.2. Hoe een ZIB te gebruiken

Onderstaande principes beschrijven hoe Vektis ZIB's gebruikt bij het opstellen van generieke gegevensblokken.

17.	Gegevensblokken hoeven niet één op één met een ZIB overeen te komen (zie ook principe 15). Wanneer gebruik wordt gemaakt van een ZIB, worden in het generieke gegevensblok alleen de gegevens uit de ZIB opgenomen die daadwerkelijk van belang zijn voor de EI-standaarden.
18.	Het is mogelijk een gegeven uit de ZIB uit te breiden als deze niet specifiek genoeg gedefinieerd is vanuit de ZIB.

	Voorbeeld: In de ZIB Patiënt is het gegeven Identificatienummer opgenomen dat meerdere keren kan voorkomen. Er is echter geen mechanisme beschreven waarmee kan worden aangegeven welk soort identificatienummer het om gaat (BSN, patiëntnummer, etc.). In de standaard is het identificatienummer daarom uitgesplitst in: • BSN • Verzekerdennummer • Patiënt identificatienummer
19.	Wanneer een gegevensblok is gebaseerd op een ZIB, is het mogelijk extra gegevens op te nemen in het gegevensblok (die dus geen onderdeel zijn van de ZIB). Dit kunnen zowel losse gegevens als gegevensblokken zijn. We doen dit alleen als de gegevens noodzakelijk zijn voor het bericht en niet al op een andere manier zijn opgenomen in de ZIB waarop het gegevensblok is gebaseerd. Deze extra gegevens plaatsen we op de functioneel gewenste positie. Toelichting: Om een bericht goed leesbaar te houden (voor mensen), is het belangrijk dat gegevens die functioneel een relatie met elkaar hebben bij elkaar staan.
20.	Waar mogelijk hanteren we in een generiek gegevensblok dezelfde volgorde als van de gegevens in de ZIB. Reden hiervoor is dat de ZIB op deze manier herkenbaar terugkomt in het gegevensblok. Wanneer vanuit de EI-standaard de volgorde van gegevens in een ZIB niet wenselijk is, kan van deze volgorde worden afgeweken. Voorbeeld: Bij een declaratie hoeven van een verzekerde alleen de NAW-gegevens te worden opgenomen in het bericht, als dit bericht wordt opgestuurd naar een servicebureau. Als het bericht direct naar een zorgverzekeraar wordt gestuurd, zijn alleen het identificatienummer en de geboortedatum van belang. Het identificatienummer, de geboortedatum en NAW-gegevens zijn echter allemaal onderdeel van de ZIB Patiënt. Binnen deze ZIB zijn deze gegevens op hetzelfde niveau gedefinieerd. Omdat binnen de EI-standaarden het versturen van declaraties via een servicebureau een andere stroom is dan het direct versturen naar de verzekeraar, is ervoor gekozen niet de volgorde van gegevens uit de ZIB te gebruiken. In plaats daarvan zijn extra niveaus aangebracht, waardoor de Verzekerde als volgt is opgenomen:

	• Verzekerde
	  • Identificatienummer
	  • Geboortedatum
	  • Aanvullende verzekerdengegevens
	  • Naam
	  • Adres
	  • …

21.	Het uitgangspunt is dat de naamgeving van de ZIB en de gegevens en gegevensblokken uit de ZIB worden overgenomen. Daar waar dit niet logisch is, kan een andere naamgeving worden gehanteerd. <u>Toelichting:</u> We willen dat de ZIB herkenbaar in een gegevensblok is opgenomen, afwijken in naamgeving zorgt ervoor dat de ZIB minder herkenbaar wordt. Omdat we de ZIB's niet één op één overnemen, maar ook losse onderdelen uit de ZIB's kunnen opnemen in gegevensblokken, is het echter niet altijd logisch om de naam van de ZIB te gebruiken voor het gegevensblok.
22.	Wanneer voor de vulling van een gegeven gebruikt wordt gemaakt van een codelijst, zijn zorgverzekeraars verplicht de codelijsten volgens de NEN-normen te volgen. Als deze codelijst afwijkt van de codelijst uit de ZIB, wordt de NEN-lijst gebruikt. In de documentatie nemen we dan wel een mapping van de ZIB-codelijst naar de NEN-codelijst op. Als er geen NEN-lijst gebruikt hoeft te worden, is het uitgangspunt dat de codelijst zoals gedefinieerd in de ZIB wordt gebruikt, mits dit de principes zoals beschreven in 2.6 Gebruik codelijsten niet tegenspreekt. Deze codelijst wordt dan ook opgenomen in het bericht.
23.	Wanneer een gegevensblok onderdelen van een of meerdere ZIB's bevat, leggen we bij het gegevensblok vast op welke ZIB's het gegevensblok is gebaseerd. Daarnaast leggen we bij de gegevens zelf ook een verwijzing naar het gegeven uit de ZIB vast. <u>Toelichting:</u> Door vast te leggen welke ZIB's zijn gebruikt, is het voor implementatie partijen (van medische informatiesystemen) eenvoudiger om het gegevensblok op de interne structuur de plotten.

24.	Wanneer bij een gegeven afkomstig uit een ZIB, wordt afgeweken van de ZIB (bijvoorbeeld in naam of gebruikte codelijst) dan wordt dit expliciet aangegeven in de Berichtspecificatie samen met de reden van de afwijking.
25.	Wanneer een generiek blok dat onderdelen van een ZIB bevat wordt gebruikt in een bericht, dan wordt op dat moment naar de cardinaliteit van de gegevens uit de ZIB gekeken. Wanneer de initiële berichtenstroom naar een partij gaat die in zijn systemen geen gebruik maakt van de ZIB's, hoeft hierbij niet te worden gelet op het verplicht zijn van het onderdeel volgens de ZIB. Wanneer de initiële berichtenstroom naar een partij gaat die in zijn systemen wel gebruik maakt van de ZIB's, moeten in elk geval de verplichte onderdelen van de ZIB zijn opgenomen. <u>Toelichting:</u> De systemen van bijvoorbeeld zorgverzekeraars en gemeenten zijn geen medische informatiesystemen. Omdat de ZIB's gericht zijn op de medische informatiesystemen, is niet te verwachten dat zorgverzekeraars en gemeenten hun systemen (in de nabije toekomst) gaan inrichten volgens de ZIB's. Voor berichten waarvoor een zorgverzekeraar of gemeente de initiële verwerker is, is het daarom niet van belang of verplichte onderdelen van een ZIB aanwezig zijn. De systemen van bijvoorbeeld zorgverleners zijn wel medische informatiesystemen. Omdat deze systemen steeds vaker zijn ingericht volgens de ZIB's, moeten bij het gebruik van een ZIB in elk geval de verplichte onderdelen zijn opgenomen omdat deze door het systeem ook als aanwezig worden geacht.
26.	Bij het toevoegen van (onderdelen van) een nieuwe ZIB, gebruiken we altijd de definitie van de ZIB uit de laatste publicatie die bij de zorgverleners in de systemen in gebruik is. Hiermee bedoelen we het concept dat de ZIB beschrijft, de gegevens die zijn opgenomen in de ZIB en de codelijsten die worden gehanteerd. We gebruiken niet de definities uit prepublicaties. In de berichtspecificatie wordt voor elk gebruikt onderdeel aangegeven welke ZIB is gebruikt en uit welke publicatie de definitie is overgenomen.
27.	Wanneer de definitie van een ZIB wijzigt bij het uitkomen van een nieuwe publicatie van de ZIB's, gaan we niet automatisch over op deze nieuwe definitie. Het kan dus ook voorkomen dat over verschillende gegevensblokken heen, verschillende versies van de ZIB's worden gehanteerd. Bij de eerstvolgende gelegenheid wordt gekeken of de wijzigingen alsnog kunnen worden doorgevoerd.

	Toelichting: Het overgaan op de nieuwe definitie zou kunnen betekenen dat er een nieuwe implementatie nodig is van de EI-standaarden die gebruik maken van het gegevensblok. We willen voorkomen dat gebruikers van de standaard wijzigingen moeten doorvoeren, enkel en alleen omdat er een nieuwe versie van de ZIB beschikbaar is.

2.6. Gebruik codelijsten

1.	Voor gegevens waar over alle zorgsoorten heen dezelfde waarden geldig zijn, worden generiek gedefinieerde codelijsten gebruikt. Toelichting: In het gebruik en controle van de codelijsten is het wenselijk dat codes dezelfde betekenis hebben. Bij gebruik van lijsten per zorgsoort zou het voor kunnen komen dat codes met gelijke betekenis in verschillende lijsten verschillende waarden hebben. Dit is niet wenselijk. Voorbeeld: Voor het geslacht van de verzekerde wordt altijd codelijst COD046-NEN gebruikt.
2.	Codes worden zoveel mogelijk generiek gedefinieerd zodat generieke codelijsten mogelijk zijn. Toelichting: Het is niet wenselijk dat er verschillende lijsten ontstaan met daarin verschillende waarden die allemaal ongeveer hetzelfde zeggen, maar net niet helemaal.
3.	Wanneer voor verschillende zorgsoorten verschillende codes gebruikt kunnen worden, worden verschillende codelijsten per zorgsoort gebruikt. Toelichting: Het is in de controles die worden uitgevoerd slecht onderhoudbaar om codelijsten te gebruiken waar de codes die gebruikt mogen worden afhankelijk zijn van de zorgsoort waarvoor een gegeven is ingevuld. Voorbeeld: De diagnose is afhankelijk van de zorgsoort. Voor de diagnose bestaan daarom codelijsten per zorgsoort.

4.	Codes hebben altijd een begindatum. Wanneer een code niet meer geldig is, heeft de code een expiratiedatum. Toelichting:
5.	***TH: Zijn er nu al andere zaken die we willen vastleggen mbt codelijsten? Wat zijn de openstaande vragen over codelijsten?***

[Nog uit te werken, hangt sterk samen met techniek en de controles die moeten worden uitgevoerd. In de principes moeten de volgende wensen terugkomen:

- ***Codes en codelijsten moeten uniform gebruikt worden over alle standaarden heen.***
- ***Het toevoegen/verwijderen van codes aan een bestaande lijst moet niet noodzakelijk leiden tot het aanpassen van alle gegevensblokken die gebruik maken van de codelijst.***

2.7. Naamgevingsconventies

Voor de naamgeving van losse gegevens, gegevensblokken en codelijsten, hanteert Vektis onderstaande principes.

1.	De naam is intuïtief te begrijpen.
2.	De naam is betekenisvol en beschrijvend.
3.	De naam is generiek herbruikbaar, een gegeven(sblok) staat op zichzelf en kan verschillend toegepast worden. Dit betekent onder andere dat de naam niet omschrijvend mag zijn. Dat wil zeggen, de naam mag geen business rules en/of waardenbereik bevatten.
4.	We hanteren Pascal casing. Dit wil zeggen dat: • de naam bestaat uit een of meer woorden, aan elkaar en zonder spaties of koppelstreepjes; • elk woord begint met een hoofdletter en wordt gevolgd door kleine letters; • bij afkortingen die alleen uit hoofdletters bestaan, is alleen de eerste letter als hoofdletter geschreven, de overige letters zijn kleine letters. Met 'elk woord' bedoelen we elk woord zoals dit in de Nederlandse taal wordt geschreven. Is een woord een samenstelling van twee woorden, maar is dit volgens de Nederlandse taal één woord, dan is er dus maar één hoofdletter.

	Voor meer informatie over Pascal casing, zie bijvoorbeeld https://techterms.com/definition/pascalcase. Voorbeelden: • Postcode blijft Postcode; • Datum overlijden wordt DatumOverlijden; • AGB-code wordt AgbCode.

5.	De naam bevat weinig tot geen afkortingen. Toelichting: Afkortingen kunnen dubbelzinnig worden opgevat of onbekend zijn. Hierdoor kan de naam verkeerd worden geïnterpreteerd.
6.	De naam bevat geen diakritische tekens. Waar deze tekens niet te vermijden zijn, wordt de letter zonder diakrieten gebruikt. Voorbeeld: • Patiënt wordt Patient. Toelichting: Voor technieken waarbij het bericht ook de naam van de gegevens(blokken) bevat, kan het voorkomen dat het uitlezen van het bericht niet goed gaat als de naam een diakriet bevat.
7.	In de naamgeving hanteren we het enkelvoud van woorden. Enige uitzondering hierop is verzekerde, hiervan hanteren we het meervoud. Voorbeeld: • AanvullendePrestatieGegevens • AanvullendeVerzekerdenGegevens Toelichting: Wanneer voor verzekerde het enkelvoud wordt gebruikt, kan het woord worden gelezen als bijvoeglijknaamwoord in plaats van het zelfstandig naamwoord dat wordt bedoeld.
8.	In een eenvoudige constructie ziet de naam er als volgt uit: prefix onderwerp, suffix datatype. Voorbeeld: AgbCode
9.	In een complexe constructie ziet de naam er als volgt uit: onderwerp, status/bewerking, datatype. Voorbeeld: ZorgaanbiederOntvangenTotaalbedrag

10.	Voor een aantal typen gegevens, bevat de naam een standaard suffix. We hanteren de volgende suffixen: • Code: voor gegevens die een code bevatten; • Nummer: voor gegevens die een nummer bevatten; • Datum: voor gegevens die een datum bevatten.

3. Architectuur bericht (XML)

Dit hoofdstuk beschrijft de architectuur principes die Vektis hanteert voor het opstellen van XML-berichten voor de EI-standaarden. Deze principes beschrijven de toepassing van de principes beschreven in hoofdstuk 2 op XML-berichten. Daarnaast staan nog een aantal principes beschreven die alleen binnen XML van belang zijn.

3.1. Opstellen XML Bericht

Onderstaande principes beschrijven hoe in het algemeen de XML van EI-berichten moet worden opgesteld. Omdat dit ook principes zijn die van belang zijn voor partijen die de berichten opstellen, zijn deze principes bij elke XML-standaard opgenomen in de berichtspecificatie of invulinstructie.

1.	De berichten zijn opgesteld in XML versie 1.0 (xml version="1.0"). Toelichting: In maart 2020 is een uitvraag gedaan welke versie van XML de systemen van diverse stakeholders ondersteunen. Hieruit bleek dat alleen versie 1.0 wordt ondersteund.
2.	Encoding van XML-berichten is UTF-8 (encoding="utf-8"). Het toevoegen van een Byte-Order-Mark (BOM) aan een XML-bericht is niet toegestaan. Toelichting: Omdat de afgesproken encoding van de XML-berichten 'UTF-8' is, is het niet nodig om een BOM mee te geven (een BOM is bedoeld voor UTF-16 berichten). Daarnaast kan het voorkomen dat het ontvangende systeem een bericht met daarin een BOM niet kan verwerken.
3.	De verzender formatteert het XML bericht volgens de gebruikelijke principes van XML-formatting. Dit kan zijn 'pretty-print' met gebruik van einderegel CR/LF, of 'single line'. De verzender dient er rekening mee te houden dat het gebruik van 'pretty print' grotere berichten oplevert dan het gebruik van 'single line'. De maximale berichtgrootte betreft de grootte van het bericht met de formattering zoals aangeleverd door de verzender. Toelichting:

	Het ontvangende systeem kan naar wens zelf formattering verwijderen of toevoegen als dit voor de verwerking van het bestand gewenst is.
4.	Het is niet toegestaan een lege klasse of leeg element op te nemen in een bericht. Wanneer een klasse of element aanwezig is, moet deze ook vulling hebben. Dit zal ook zoveel mogelijk worden afgedwongen vanuit de XSD. Voor verplichte klassen en (samengestelde) elementen houdt dit in dat deze altijd aanwezig zijn en vulling hebben. Voor conditionele klassen en (samengestelde) elementen betekent dit dat deze alleen worden opgenomen als aan de conditie is voldaan. Toelichting: Het alleen opnemen van velden als ze daadwerkelijk inhoud hebben, zorgt ervoor dat de berichten klein blijven en niet vol staan met onnodige lege klassen of (samengestelde) elementen.

3.2. Opbouw en inhoud XSD

Over de opbouw en inhoud van XML Schemadefinities zijn een aantal ontwerpkeuzes gemaakt. Deze zijn gebaseerd op herbruikbaarheid van elementen, leesbaarheid en efficiënte automatische verwerking van XML-berichten. Onderstaande principes beschrijven deze ontwerpkeuzes.

1.	XSD's staan op zichzelf, er wordt (voorlopig) geen gebruik gemaakt van een te importeren basisschema waarin generiek gedefinieerde klassen en elementen zijn opgenomen. Toelichting: Een basisschema kan worden gebruikt als technische vertaling van het generiek definiëren van gegevens(blokken). Vektis onderzoekt echter nog in hoeverre de tooling die wordt gebruikt voor het opstellen van de XSD's de mogelijkheid bevat om een bouwdoos te gebruiken zoals beschreven in paragraaf 2.1. Op zichzelf staande XSD's hebben namelijk als voordeel dat deze leesbaarder en makkelijker te doorzoeken zijn dan XSD's die andere XSD's importeren.

	NB: Omdat alle XSD's dezelfde namespace hanteren (zie paragraaf 0), blijven berichten die worden gegenereerd op basis van op zichzelf staande XSD's ook geldig als in de toekomst toch wordt besloten een basisschema te hanteren.
2.	XML Schemadefinities volgen het ontwerppatroon Venetian Blind. Dit houdt in dat er één globaal gedefinieerd root element is. Binnen de EI-standaarden zal dit altijd het 'Bericht' element zijn. Alle overige elementen zijn gedefinieerd binnen dit element. Daarbij worden (herbruikbare) gegevens als complexType of simpleType gedefinieerd. Vanuit de elementen wordt naar deze typen verwezen. Voor meer informatie over het ontwerppatroon Venetian Blind, zie bijvoorbeeld https://www.oracle.com/technical-resources/articles/java/design-patterns.html. Voorbeeld: <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ds801="http://ei.vektis.nl/berichten" targetNamespace="http://ei.vektis.nl/berichten" elementFormDefault="qualified"> **Alleen Bericht is mogelijk als root element van het XML-bericht.** <xs:element name="Bericht" type="ds801:Bericht573Type"></xs:element> **Overige elementen zoals Header, Overzicht en Verzekerde zijn gedefinieerd binnen het element bericht. Deze bestaan dus niet buiten het bericht en kunnen daarom niet als root worden gebruikt.** <xs:complexType name="Bericht573Type"> <xs:sequence> **Header verwijst naar het complexType Header573Type** <xs:element name="Header" type="ds801:Header573Type"/> <xs:element name="Overzicht" type="ds801:Overzicht573Type"/> <xs:element name="Verzekerde" type="ds801:Verzekerde573Type" maxOccurs="unbounded"/> </xs:sequence> </xs:complexType> **De gegevens van de header zijn opgenomen in het complexType Header573Type en kunnen hierdoor op verschillende plekken in het bericht worden aangeroepen. (In dit geval bevat het bericht maar één header en zal er dus maar één aanroep worden gedaan.)**

	<xs:complexType name="Header573Type"> <xs:sequence> <xs:element name="Berichtcode"> <xs:simpleType> **Berichtcode verwijst naar simpleType BerichtcodeType** <xs:restriction base="ds801:BerichtcodeType"> <xs:enumeration value="573"/> </xs:restriction> </xs:simpleType> </xs:element> … </xs:sequence> </xs:complexType> **De berichtcode is gedefinieerd in het simpleType BerichtcodeType en kan hierdoor op verschillende plekken in het bericht worden aangeroepen.** <xs:simpleType name="BerichtcodeType"> <xs:restriction base="xs:string"> <xs:enumeration value="573"/> <xs:enumeration value="574"/> </xs:restriction> </xs:simpleType> <u>Toelichting:</u> Binnen de EI-standaarden moet 'Bericht' altijd het root element zijn van het XML bericht. We willen gebruikers van de XSD daarom ook geen mogelijkheid bieden om een ander root element te kiezen. De herbruikbaarheid die in de Venetian Blind wordt gebruikt, sluit aan bij het doel van uniformiteit in de berichten.
3.	Elementen die van een standaard type zijn (bijvoorbeeld string of integer) worden alleen als apart, element specifiek, type gedefinieerd als het vermoeden bestaat dat er in de toekomst meer restricties kunnen volgen op het element. <u>Voorbeeld:</u> In de Header zijn de verzender en verzenderrol opgenomen. Beide bevatten een numerieke waarde. Voor beide geldt ook dat de waarde niet verder gespecificeerd wordt dan numeriek (de verzender kan zowel een AGB-code als

	een UZOVI bevatten, dus kan niet verder worden gespecificeerd). Deze velden zijn daarom op de volgende manier opgenomen in de XSD: <xs:element name="Verzender" type="ds801:NumeriekType"/> <xs:element name="VerzenderRol" type="ds801:NumeriekType"/> We maken dus geen simpleType VerzenderType en VerzenderRolType. Toelichting: Standaard typen zijn al herbruikbare typen. Als de restrictie niet verder gaat dan de herbruikbare typen, is het niet nodig losse typen te definiëren.
4.	Attributen worden alleen gebruikt voor metadata. Alle gegevens die door applicaties worden verwerkt, worden opgenomen als elementen. Toelichting: Attributen zijn niet uitbreidbaar. Van de gegevens die door applicaties worden verwerkt, is niet zeker of deze ooit uitgebreid moeten worden. Daarom willen we deze niet vastleggen in attributen.
5.	Gegevensblokken worden opgenomen als complexType met indicator 'sequence' (in sommige gevallen kan ook voor 'choice' worden gekozen). Er wordt geen gebruik gemaakt van all. Toelichting: De 'all' indicator geeft aan dat de elementen in elke willekeurige volgorde kunnen worden opgenomen. Het is hierbij niet mogelijk om een element meerdere keren voor te laten komen. Dezelfde optie zou dan namelijk meerdere keren op verschillende plekken in de keuze voor kunnen komen.
6.	Elementen die repeterend kunnen voorkomen, worden niet opgenomen in een collectie (een complex type met als naam bijvoorbeeld het meervoud van het element). **Opmerking:** op het moment kunnen nog niet alle verzekeraars omgaan met repeterende simpleTypes. Bij het toepassen van dit principe moet hierrekening mee worden gehouden. Toelichting: Het gebruik van collecties zorgt ervoor dat een wijziging die van een element dat maar één keer voorkomt een repeterend element maakt, niet backwards

	compatible is. Dit is niet wenselijk, het repeterend maken van een element moet backwards compatible zijn.

7.	Het aantal voorkomens van een klasse of (samengesteld) element wordt op de volgende wijze opgenomen in de XSD:
	• Verplicht element dat één keer voor mag komen (cardinaliteit = 1)
	  • minOccurs is niet opgenomen (default = 1)
	  • maxOccurs is niet opgenomen (default =1)
	• Verplicht element dat vaker voor mag komen (cardinaliteit = 1 – n)
	  • minOccurs is niet opgenomen (default = 1)
	  • maxOccurs = “unbounded”
	• Optioneel element dat één keer voor mag komen (cardinaliteit = 0 – 1)
	  • minOccurs = “0”
	  • maxOccurs is niet opgenomen (default =1)
	• Optioneel element dat vaker voor mag komen (cardinaliteit = 0 – n)
	  • minOccurs = “0”
	  • maxOccurs = “unbounded”
	
	Voorbeelden:
	• Verplicht element dat één keer moet voorkomen:
	  <xs:element name="Berichtcode" type="BerichtcodeType"/>
	• Optioneel samengesteld element dat vaker voor mag komen:
	  <xs:element name="AanvullendPrestatieKenmerk "
	  type="AanvullendPrestatieKenmerkType" **minOccurs="0"**
	  maxOccurs="unbounded"/>
	
	Toelichting:
	We maken zoveel mogelijk gebruik van de default instellingen van XSD.

8.	Wanneer voor een gegeven slechts één waarde mogelijk is, wordt dit expliciet afgedwongen met een restrictie. Er wordt geen gebruik gemaakt van default of vaste waarden. Toelichting: Bij gebruik van default en vaste waarden, wordt tijdens de validatie tegen het schema, het originele XML-bericht gewijzigd: het verplichte veld of de default waarde wordt toegevoegd. Dat is niet wenselijk omdat het element mogelijk opzettelijk weg of leeg is gelaten.
9.	Numerieke elementen worden op de volgende manieren gedefinieerd:

	• nummers waar alle cijfers, ook voorloopnullen, zichtbaar zijn worden met een string en een numerieke restrictie gedefinieerd; • getallen waarmee je kan rekenen of nummers die opgehoogd worden, worden met een integer of decimal gedefinieerd. Voorbeelden: • Nummer met mogelijk voorloopnullen: <xs:element name="BSN" type="ds801:NumeriekType" minOccurs="0"/> <xs:simpleType name="NumeriekType"> **<xs:restriction base="xs:string">** <xs:minLength value="1"/> **<xs:pattern value="[0-9]*"/>** </xs:restriction> </xs:simpleType> • Decimaal getal waarmee kan worden gerekend: <xs:element name="BtwPercentageDeclaratiebedrag" type="ds801:Decimal2Type" minOccurs="0"/> <xs:simpleType name="Decimal2Type"> **<xs:restriction base="xs:decimal">** <xs:minInclusive value="0"/> <xs:fractionDigits value="2"/> </xs:restriction> </xs:simpleType>
10.	Elementen worden opgenomen als string op het moment dat het niet mogelijk is expliciet iets over het formaat te zeggen. Wanneer het formaat wel duidelijk is, wordt het gepaste datatype gekozen. Voorbeeld: • Het formaat van het patiënt identificatienummer hangt af van het systeem dat dit nummer genereert. Het patiënt identificatienummer is daarom opgenomen als string. • Het formaat van het UZOVI-nummer is bekend, namelijk numeriek. Het UZOVI-nummer is daarom opgenomen als numeriek. Toelichting: Een string kan heel veel waarden bevatten. Wanneer het mogelijk is specifiek te zijn over de waarde, willen we dit ook duidelijk maken in het gekozen datatype.

11.	Voorlopig bevatten de XML Schemadefinities geen functionele documentatie van de elementen en typen die zijn opgenomen in het schema (bijvoorbeeld met behulp van annotations). Wanneer een XSD technische verduidelijking vraagt, nemen we deze documentatie op door middel van annotations. Toelichting: De functionele documentatie van de XSD ligt vast in de berichtspecificatie. Voorlopig genereren we de berichtspecificatie niet op basis van de XSD, maar stellen we deze apart op. Wanneer we de functionele documentatie ook in de XSD zelf zouden opnemen, zou dit betekenen dat we deze dubbel vastleggen. Hierbij bestaat het risico dat XSD en berichtspecificatie uit elkaar gaan lopen.
12.	De titel van de XSD en informatie over de gebruiksrechten en een eventuele toelichting op het gebruik van de XSD in relatie tot de overige documenten uit de standaard, is aan het begin van de XSD opgenomen als commentaar (tussen <!-- --->). Toelichting: Deze informatie is alleen van belang voor degene die de XSD implementeert en hoeft daarom niet leesbaar te zijn voor applicaties.
13.	Informatie over de XSD zelf is gedocumenteerd binnen appinfo. Binnen de appinfo zijn de volgende gegevens opgenomen: • standaard: verwijzing naar de standaard, inclusief versie en uitgave • publicatiedatum: publicatiedatum van de XSD Voorbeeld: <xs:annotation> <xs:appinfo> <doc: standaard>gds801v1.0_573_XSDu1</doc:standaard> <doc:publicatiedatum>2021-03-15</doc:publicatiedatum> </xs:appinfo> </xs:annotation> Toelichting: Applicaties kunnen appinfo uitlezen en gebruik maken van deze documentatie bij het gebruik van de XSD.

3.3. Namespaces

Vektis hanteert onderstaande principes voor het gebruik van namespaces. In XML wordt de namespace gebruikt om elementnamen uniek te identificeren. De namespace is gedefinieerd als een URI (uniform resource identifier).

1.	De gemeenschappelijke URI voor Vektis namespaces is http://ei.vektis.nl.
2.	Alle berichten maken gebruik van dezelfde target namespace (targetNamespace="http://ei.vektis.nl/berichten"). Toelichting: Zoals beschreven in paragraaf 2.1 worden alle gegevens(blokken) die binnen de EI-standaarden voor kunnen komen op één plek gedefinieerd. Dit betekent dat er maar één definitie kan zijn van een gegeven(sblok). Binnen de XSD's van verschillende berichten kan er daardoor ook geen conflict ontstaan tussen betekenissen van klassen en elementen met dezelfde naam. Om dit expliciet te maken, hanteren we één namespaces voor alle XSD's.
3.	Binnen de XSD zelf worden de volgende namespaces gebruikt: • W3C XMLSchema (xmlns:xs="http://www.w3.org/2001/XMLSchema"); • een namespace voor documentatie (xmlns:doc="http://ei.vektis.nl/documentatie"); • een namespace voor het bericht zelf (xmlns:ei="http://ei.vektis.nl/berichten").
4.	Binnen de XSD zelf worden geen default namespaces gebruikt. Alle namespaces hebben een namespace-alias.
5.	Alle namespaces zijn expliciet gekwalificeerd (elementFormDefault = "qualified"). Dit betekent dat alle elementen onderdeel zijn van een namespace (de EI-berichten namespace). De berichten kunnen geen elementen bevatten waarvan niet is aangegeven tot welke namespaces deze behoren.
6.	De URI die wordt gebruikt om de namespace aan te duiden, verwijst (nog) niet naar een daadwerkelijke locatie. Toelichting: Alle gegevens die nodig zijn om de XSD te kunnen interpreteren, worden gepubliceerd bij de standaard waar de XSD toe behoort. In de toekomst koppelt Vektis mogelijk wel extra informatie aan de namespace die te raadplegen valt via de URL die de namespace dan beschrijft.

4. Versiebeheer

Bij het opstellen van het versiebeheer voor de EI-standaarden heeft Vektis rekening gehouden met het volgende:

- Uit de versionering blijkt of het gaat om een correctie of verscherping op een bestaande standaard, of dat de standaard daadwerkelijk is gewijzigd.
  Daarbij legt Vektis de reden voor de nieuwe (sub)versie vast. Het komt er op neer dat de partij die de standaard gebruikt moet weten welke versie het bestand/bericht betreft, zodat het tegen de juiste set controles kan worden aangehouden.
- Wanneer meerdere partijen gebruik maken van een standaard, moet een wijziging die maar voor een aantal partijen van belang is niet een nieuwe implementatie van de standaard voor álle partijen betekenen.
- Het moet altijd mogelijk zijn voor partijen om over te stappen naar de laatste versie van de standaard.
- Wanneer meerdere partijen gebruik maken van een standaard, hoeven verwerkers van de berichten, zoals VECOZO en zorgverzekeraars, maar één versie van de standaard te ondersteunen (met uitzondering van een eventuele overgangsfase).
- Het versiebeheer moet de situatie ondersteunen waarbij de ene partij al over is op een nieuwe versie en/of subversie en een andere partij nog niet.

4.1. Versionering

Vektis hanteert de volgende gelaagdheid in het versiebeheer van de EI-standaarden:

- hoofdversie;
- subversie;
- uitgave.

Hoofdversies en subversies zijn van toepassing op de gehele standaard, dat wil zeggen[1]:

- berichtspecificatie
- standaardbeschrijving
- invulinstructie
- registratie bedrijfs- en controleregels
- OAF-document (bij schadelast standaarden)
- XML Schema Definities (XSD)

[1] Andere documentatie die bij een standaard in paragraaf 4. Documentatie te vinden is, zoals b.v. de Koppeltabel WLZ en Controlematrix bij de AW319 of een GPH-productcodetabel bij de LH307, vallen buiten scope van dit beleid. De reden daarvoor is dat nieuwe versies van dergelijke documenten los staan van wijzigingen aan de definitie van een standaard en een eigen nummering hebben of waarbij nummering niet van toepassing is.

- controle XSLT's

Uitgaven zijn van toepassing op specifieke documenten binnen de standaard. Binnen één (sub)versie van een standaard kunnen de verschillende documenten daarom een verschillende uitgave hebben.

Opmerkingen:
- (Sub)versies en uitgaven zijn altijd cumulatief. Dat wil zeggen dat wijzigingen in de standaard altijd alleen in de laatste hoofdversie, subversie en uitgave worden doorgevoerd. Een wijziging wordt dus nooit in twee verschillende (sub)versies of uitgaven doorgevoerd.
- Bij de ophoging van het versie- of subversienummer krijgen alle documenten weer 'u1' als uitgavenummer
- Wijzigingen worden in het mutatieoverzicht van het betreffende document beschreven. Bij een nieuwe versie of subversie wordt dit in de revisiehistorie van de berichtspecificatie (BER), invulinstructie (INV) en standaardbeschrijving (STB) vermeld.
- Wijzigingen in de inhoud van codelijsten die in een EI-standaard worden gebruikt, zijn geen onderdeel van het versiebeheer van de standaard. Uitzondering hierop is als er in de standaard specifieke controles gelden voor toegevoegde of verwijderde codes.

4.1.1. Hoofdversie

De hoofdversie heeft als doel aan te geven dat de wijze waarop het bericht moet worden opgesteld zodanig is veranderd dat een nieuwe implementatie van de standaard noodzakelijk is voor alle partijen die de standaard gebruiken. Generieke aanpassingen aan een standaard, die door alle partijen die de standaard gebruiken moeten worden doorgevoerd, leiden daarmee tot een nieuwe hoofdversie van de standaard.
Voorbeelden van wijzigingen die leiden tot een nieuwe hoofdversie zijn:
- er wordt een nieuw verplicht gegeven toegevoegd; (Voor ASCII-standaarden geldt dat het toevoegen van een nieuw gegeven altijd tot een nieuwe hoofdversie leidt, ongeacht of dit gegeven verplicht is of niet. Door het toevoegen van een gegeven aan een ASCII-standaard wijzigt de recordlengte namelijk.)
- een gegeven dat niet verplicht was, wordt verplicht;
- een gegeven wordt verwijderd uit de standaard;
- een gegeven wordt verplaatst binnen de standaard;
- er wordt een algemene restrictie gelegd op de vulling van een element die eerst niet bestond;

- een gegeven verandert van datatype waarbij de restricties van het datatype krapper zijn dan voorheen.

Verschillende hoofdversies hoeven niet backwards compatible met elkaar te zijn. Dat wil zeggen dat berichten die zijn opgesteld op basis van de oude hoofdversie, geen geldige berichten hoeven te zijn volgens de nieuwe hoofdversie. Een bericht dat is opgesteld op basis van versie 1.7 hoeft bijvoorbeeld geen geldig bericht te zijn volgens versie 2.0.

4.1.2. Subversie

De subversie heeft als doel aan te geven dat er een wijziging in de wijze waarop het bericht moet worden opgesteld heeft plaatsgevonden. Wijzigingen die tot een nieuwe subversie leiden, hoeven echter niet noodzakelijk door alle partijen die de standaard gebruiken te worden doorgevoerd (dit mag natuurlijk wel).
Voorbeelden van wijzigingen die leiden tot een nieuwe subversie zijn:

- er wordt een nieuw conditioneel gegeven toegevoegd; (Zoals al aangegeven leidt dit bij ASCII-standaarden tot een nieuwe hoofdversie.)
- een verplicht gegeven wordt conditioneel gemaakt;
- een gegeven dat éénmaal voor kon/moest komen, kan vaker voorkomen. (Voor ASCII-standaarden leidt dit tot een nieuwe hoofdversie omdat de recordlengte hierdoor wijzigt.)

Verschillende subversies moeten backwards compatible met elkaar te zijn. Dat wil zeggen dat berichten die zijn opgesteld op basis van de oude subversie, ook geldig zijn volgens de nieuwe subversie. Een bericht dat is opgesteld volgens versie 1.1, is bijvoorbeeld ook nog een geldig bericht volgens versie 1.2, 1.3, etc.
Dit houdt in dat de wijziging die voor de nieuwe subversie zorgt, in het algemeen conditioneel/optioneel toe te passen moet zijn. Via controles kan worden afgevangen dat in specifieke situaties de wijziging wel verplicht is.

Merk op dat elke wijziging in de wijze waarop het bericht moet worden opgesteld, tenminste tot een nieuwe subversie leidt. Op deze manier is altijd vanuit de versienummers die in het bericht zijn opgenomen af te leiden aan welke voorwaarden het bericht zou moeten voldoen.

4.1.2.1. Subversies in de Generieke Declaratiestandaard

De Generieke Declaratiestandaard (GDS) is een voorbeeld van een standaard die door veel verschillende partijen wordt gebruikt. Wijzigingen aan de GDS zullen in de meeste

gevallen maar op een paar partijen betrekking hebben en daarom vaak tot nieuwe subversies leiden. Om een overzicht te houden van welke subversies voor welke partijen van belang zijn, is in de GDS (in de Standaardbeschrijving) een overzicht opgenomen van de verschillende prestatiecodelijsten die de standaard ondersteunt en welke subversie minimaal moet worden gehanteerd voor elke prestatiecodelijst. In de controles bij de GDS wordt ook gecontroleerd op deze minimale subversie.

Voor andere EI-standaarden die ook door veel partijen worden gebruikt, kan op een vergelijkbare wijze overzicht worden gecreëerd.

4.1.3. Uitgave

De uitgave heeft als doel aan te geven dat er een correctie of verduidelijking heeft plaatsgevonden in een document binnen een (sub)versie van de standaard. Deze correcties of verduidelijkingen hebben geen invloed op de wijze waarop het bericht moet worden opgesteld.
Voorbeelden van wijzigingen die leiden tot een nieuwe uitgave zijn:

- correctie van onjuiste beschrijving;
- tekstuele verduidelijkingen vanwege verschillende interpretaties;
- onvolledige beschrijving;
- niet meer actuele beschrijving;
- XSD komt niet overeen met de beschrijving in de berichtspecificatie.

4.1.4. Versiebeer tijdens de implementatiefase

Zolang een (sub)versie van een standaard nog niet in productie is genomen, worden alle soorten wijzigingen via een nieuwe uitgave opgeleverd. Reden hiervoor is dat alle betrokken partijen nog bezig zijn met de implementatie van de standaard en er daarom geen rekening hoeft te worden gehouden met eventuele backwards compatibiliteit of (sub)versies die naast elkaar moeten kunnen bestaan.

4.1.5. Versiebeheer voor een berichtenfamilie

Wanneer berichten voorkomen in families, zijn de volgende situaties mogelijk:

- alle partijen maken gebruik van (ontvangen of versturen) alle berichten in de familie;
- er zijn partijen die gebruik maken van (ontvangen of versturen) een subset van de berichten in de familie.

In het versiebeheer van berichten die een familie vormen, moet rekening worden gehouden met beide situaties.

In de eerste situatie zal bij een wijziging in een bericht, ongeacht welk bericht dit is, dit op elke partij van invloed zijn. In de tweede situatie kan het voorkomen dat bij een wijziging in een bericht, dit bericht geen onderdeel is van de subset die een partij gebruikt. Het is in dit geval niet wenselijk dat de partij die het bericht niet gebruikt alsnog een nieuwe versie moet implementeren. Inhoudelijk is er in dit geval namelijk niets gewijzigd voor die partij.

Vanuit dit oogpunt hanteert Vektis geen overkoepelende versie voor alle berichten in de berichtenfamilie, maar een versie per bericht ongeacht of deze in een familie voorkomt.

4.2. Implementatie in de keten

Wanneer Vektis een nieuwe (sub)versie van een EI-standaard uitbrengt, moeten in de keten afspraken worden gemaakt vanaf welk moment de (sub)versie kan/moet worden gebruikt.

Wanneer over wordt gegaan op een nieuwe (sub)versie, zijn er twee mogelijkheden:

- Er is één moment vanaf wanneer alle partijen berichten moeten aanleveren volgens de nieuwe standaard. Dit kan bijvoorbeeld het geval zijn wanneer er een wettelijke grondslag is voor de wijzigingen in de nieuwe (sub)versie.
- Er is geen hard moment waarop iedereen de nieuwe versie moet/kan gebruiken. In dit geval kan gedurende een periode afhankelijk van de aanleverende partij de nieuwe of de oude versie worden gebruikt.

In de keten wordt afgesproken welk van deze twee mogelijkheden wordt gekozen en hoe moet worden omgegaan met de overgangsperiode.

Wanneer een standaard door veel verschillende partijen wordt gebruikt, zal het vaak voorkomen dat binnen de standaard meerdere subversies tegelijk in productie zijn. Omdat de subversies backbards compatible zijn, hoeven de verwerkers van de berichten echter alleen de laatste subversie geïmplementeerd te hebben. Een mogelijke uitzondering hierop is wanneer er wordt gekozen om voor één zorgsoort meerdere subversies in productie te hebben.

4.3. Business cases

Deze paragraaf beschrijft een aantal business cases waaruit blijkt hoe het beschreven versiebeheer zich vertaalt naar de praktijk. De beschreven business cases hebben betrekking tot wijzigingen aan de Generieke Declaratiestandaard (GDS), maar zijn ook te vertalen naar andere standaarden.

4.3.1. Toevoegen prestatiecodelijst

Het toevoegen van een prestatiecodelijst aan de GDS mag geen effect hebben op partijen die al gebruik maken van de GDS. Het toevoegen van een prestatiecodelijst aan de GDS leidt daarom altijd tot een nieuwe subversie van de GDS. Dit betekent dat alle aanpassingen ten behoeve van de nieuwe prestatiecodelijst backwards compatible moeten zijn met de vorige subversie van de GDS.

Bij het toevoegen van een prestatiecodelijst aan de GDS, kan onder andere het volgende voorkomen:

- Voor de prestatiecodelijst moet een gegeven worden opgenomen in de declaratie dat nog niet beschikbaar is in de GDS.
    - Oplossing: Toevoegen van een conditioneel element of gegevensblok aan de standaard. Dit kan nooit een verplicht element of gegevensblok zijn omdat partijen die al gebruik maken van de GDS dit element of gegevensblok niet zullen aanleveren. Wanneer het gegeven dat moet worden opgenomen wel verplicht is voor de toe te voegen prestatiecodelijst, kan dit worden afgedwongen via een prestatiecodelijst specifieke controle.
- Voor de prestatiecodelijst kan een bestaand gegeven dat verplicht moet worden opgenomen in de GDS, niet worden aangeleverd.
    - Oplossing: Het bestaande element of gegevensblok wordt conditioneel gemaakt. Voor de prestatiecodelijsten waarvoor dit gegeven wel verplicht moet worden aangeleverd, kunnen prestatiecodelijst specifieke controles worden toegevoegd.
- Voor de prestatiecodelijst moet een gegeven dat maar één keer kan worden opgenomen in de GDS, vaker voor kunnen komen.
    - Oplossing: De cardinaliteit van het element of gegevensblok wordt aangepast. Voor de prestatiecodelijsten waarvoor dit gegeven maar één keer kan voorkomen, kunnen eventueel prestatiecodelijst specifieke controles worden toegevoegd. Hierbij moet wel worden gekeken wat de noodzaak van de controle is. Als het geen problemen oplevert wanneer andere

prestatiecodelijsten het gegeven ook vaker aanleveren, zijn extra controles niet nodig.

- Voor de prestatiecodelijst mag een gegeven dat in de GDS vaker kan voor komen, maar één keer voorkomen.
    - Oplossing: Wanneer het problemen oplevert als dit gegeven voor de prestatiecodelijst vaker wordt aangeleverd, kan een prestatiecodelijst specifieke controle worden toegevoegd.

4.3.2. Doorvoeren wijziging voor een prestatiecodelijst

Het doorvoeren van een wijziging aan de GDS die niet van toepassing is voor alle prestatiecodelijsten die gebruik maken van de GDS, mag geen effect hebben op prestatiecodelijsten waarvoor de wijziging niet van toepassing is. Het doorvoeren van zo'n wijziging leidt daarom altijd tot een nieuwe subversie of uitgave van de GDS. Dit betekent dat alle aanpassingen ten behoeve van de wijziging backwards compatible moeten zijn met de vorige subversie van de GDS.

Voor een prestatiecodelijst kunnen bijvoorbeeld de volgende wijzigingen worden doorgevoerd die leiden tot een nieuwe subversie van de standaard[2]:

- Er moet voor de prestatiecodelijst een conditioneel gegeven worden toegevoegd.
    - Oplossing: Toevoegen van een conditioneel element of gegevensblok aan de standaard.
- Er moet voor de prestatiecodelijst een verplicht gegeven worden toegevoegd.
    - Oplossing: Toevoegen van een conditioneel element of gegevensblok aan de standaard. Voor de prestatiecodelijst waarvoor het gegeven verplicht is, kan dit worden afgedwongen via een prestatiecodelijst specifieke controle.
- Voor de prestatiecodelijst moet een gegeven niet meer worden aangeleverd.
    - Oplossing: In de prestatiecodelijst specifieke invulinstructie wordt aangegeven dat het gegeven niet meer hoeft te worden opgenomen in het bericht. Eventuele prestatiecodelijst specifieke controles op het gegeven worden verwijderd.
- De wijze waarop een gegeven voor een prestatiecodelijst wordt opgenomen wijzigt.
    - Oplossing: De prestatiecodelijst specifieke invulinstructie wordt aangepast.
- De condities waaronder een gegeven voor een prestatiecodelijst wordt opgenomen wijzigen.

[2] Bij het opstellen van deze voorbeelden heeft Vektis gekeken naar het soort wijzigingen dat de afgelopen 3 jaar via RfC's is doorgevoerd.

    - Oplossing: De prestatiecodelijst specifieke invulinstructie en prestatiecodelijst specifieke controles worden aangepast.
- De codelijst die voor een prestatiecodelijst voor een gegeven wordt gebruikt wijzigt.
    - Oplossing: De prestatiecodelijst specifieke controle op de gebruikte codelijst wordt aangepast.
- De maximale lengte die een gegeven kan hebben voor een prestatiecodelijst wijzigt.
    - Oplossing: Wanneer de maximale lengte groter moet worden, wordt de maximale lengte van het gegeven vergroot. Eventueel kunnen prestatiecodelijst specifieke controles worden toegevoegd voor andere prestatiecodelijsten om te controleren dat daar de oude maximale lengte wordt gebruikt.
- Er moet voor de prestatiecodelijst voor een gegeven een extra controle worden uitgevoerd.
    - Oplossing: Er wordt een prestatiecodelijst specifieke controle toegevoegd.
- Er moet voor de prestatiecodelijst voor een gegeven een controle worden aangepast.
    - Oplossing: De prestatiecodelijst specifieke controle wordt gewijzigd.
- Er moet voor de prestatiecodelijst voor een gegeven een controle worden verwijderd.
    - Oplossing: De prestatiecodelijst specifieke controle wordt verwijderd.

Daarnaast kunnen bijvoorbeeld de volgende wijzigingen worden doorgevoerd die leiden tot een nieuwe uitgave van de standaard[3]:

- De wijze waarop een gegeven voor een prestatiecodelijst wordt opgenomen moet worden verduidelijkt.
    - Oplossing: De prestatiecodelijst specifieke invulinstructie wordt aangepast.
- De condities waaronder een gegeven voor een prestatiecodelijst wordt opgenomen moet worden verduidelijkt.
    - Oplossing: De prestatiecodelijst specifieke invulinstructie wordt aangepast.

4.3.3. Doorvoeren wijziging voor alle prestatiecodelijsten

Het doorvoeren van een wijziging aan de GDS die van toepassing is voor alle prestatiecodelijsten die gebruik maken van de GDS hoeft niet backwards compatible te zijn en mag daarom leiden tot een nieuwe hoofdversie van de GDS. Afhankelijk van de situatie kan dit ook leiden tot een nieuwe subversie of uitgave van de GDS.

[3] Bij het opstellen van deze voorbeelden heeft Vektis gekeken naar het soort wijzigingen dat de afgelopen 3 jaar via RfC's is doorgevoerd.

De volgende wijzigingen kunnen bijvoorbeeld worden doorgevoerd die leiden tot een nieuwe hoofdversie van de standaard:

- Er moet voor alle prestatiecodelijsten een verplicht gegeven worden toegevoegd.
    - Oplossing: Toevoegen van een verplicht element of gegevensblok aan de standaard.
- Er moet voor alle prestatiecodelijsten een conditioneel gegeven worden toegevoegd.
    - Oplossing: Toevoegen van een conditioneel element of gegevensblok aan de standaard. In principe is dit een wijziging die backwards compatible is. Afhankelijk van de situatie kan er daarom ook voor worden gekozen om een nieuwe subversie aan te maken.
- Een gegeven dat conditioneel was moet voor alle prestatiecodelijsten verplicht worden.
    - Oplossing: Verplicht maken van het element of gegevensblok.
- Een gegeven dat niet meer wordt gebruikt moet worden verwijderd uit de standaard.
    - Oplossing: Verwijderen van het element of gegevensblok uit de standaard.

Daarnaast kunnen bijvoorbeeld de volgende wijzigingen worden doorgevoerd die leiden tot een nieuwe uitgave van de standaard:

- Een gegeven of restrictie in de XSD komt niet overeen met de specificaties zoals beschreven in de berichtspecificatie.
    - Oplossing: De XSD wordt aangepast zodat deze weer overeenkomt met de berichtspecificatie.
- De wijze waarop een gegeven voor alle prestatiecodelijsten wordt opgenomen moet worden verduidelijkt.
    - Oplossing: De generieke invulinstructie wordt aangepast.
- De condities waaronder een gegeven voor alle prestatiecodelijsten wordt opgenomen moet worden verduidelijkt.
    - Oplossing: De generieke invulinstructie en eventueel berichtspecificatie en RBC worden aangepast.

4.3.4. Mutaties op prestatiecodelijsten

In de GDS wordt onderscheid gemaakt tussen verschillende situaties op basis van de prestatiecodelijst waar de gedeclareerde prestatie op staat. Wetswijzigingen kunnen er voor zorgen dat er mutaties plaatsvinden op het niveau van de prestatiecodelijsten. Dit soort mutaties zullen niet van toepassing zijn voor alle prestatiecodelijsten die gebruik maken van de GDS en mogen geen effect hebben op prestatiecodelijsten waarvoor de

wijziging niet van toepassing is. Het doorvoeren van zo'n wijziging leidt daarom altijd tot een nieuwe subversie van de GDS.

Naar aanleiding van wetswijzigingen kunnen bijvoorbeeld de volgende wijzigingen worden doorgevoerd:

- Prestaties die op meerdere prestatiecodelijsten stonden, worden samengevoegd op één prestatiecodelijst.
    - Oplossing: Er worden nieuwe instructies en controles toegevoegd aan de standaard specifiek voor de (nieuwe) prestatiecodelijst waarop de prestaties zijn samengevoegd. Controles de golden voor de prestatiecodelijsten die niet meer worden gebruikt, krijgen een einddatum.
- Een prestatiecodelijst expireert.
    - Oplossing: Controles de golden voor de geëxpireerde prestatiecodelijst krijgen een einddatum.
- De prestaties die op de prestatiecodelijst staan vallen onder andere wetgeving (verplaatsen bijvoorbeeld van de ZvW naar de WLZ).
    - Oplossing: De GDS hoeft in de toekomst niet alleen gebruikt te worden voor het declareren van zorg die onder de ZvW valt. Wanneer wordt besloten om ook vanuit de nieuwe wetgeving voor de declaraties gebruik te maken van de GDS, hoeft de standaard niet te worden aangepast. Wanneer geen gebruik wordt gemaakt van de GDS, krijgen controles de golden voor de prestatiecodelijst een einddatum.
- Er wordt een (groep) prestatiecode(s) toegevoegd waarvoor een specifiek declaratiebeleid geldt.
    - Oplossing: Wanneer nodig kan de prestatiecodelijst specifieke invulinstructie worden aangepast. Daarnaast kunnen er LCB's worden ontwikkeld voor de betreffende (groep) prestatiecode(s).

4.4. Implementatie versiebeheer in XML-standaarden

In XML-standaarden wordt het versiebeheer op de volgende wijze door Vektis geïmplementeerd:

- Aan de XSD wordt het schema attribuut 'version' toegevoegd zodat uit de XSD duidelijk blijkt om welke versie het gaat. In dit attribuut worden de versie, subversie en uitgave opgenomen.
- In de XML-berichten moet in het root-element 'Bericht' het attribuut 'SchemaVersie' worden opgenomen waarin de versie moet worden opgenomen van de XSD waarop het bericht is gebaseerd (deze is dus gelijk aan het eerder genoemde version attribuut

van het schema). Hierdoor is voor verwerkers van het bericht duidelijk op welke versie van de XSD het bericht gebaseerd is en kunnen eventueel parallelle stromen voor verschillende versies worden ingericht.

- In XML-berichten worden in de Header elementen voor de versie en de subversie opgenomen. Dit is in lijn met de wijze waarop de versie en subversie in de ASCII-standaarden zijn opgenomen. Daarnaast blijft de informatie over de versie en de subversie behouden wanneer het 'Bericht' envelopje niet meer beschikbaar is.
- Via de XSD wordt afgedwongen welke SchemaVersies zijn toegestaan zodat versies die niet backwards compatible zijn al op het niveau van de SchemaVersie kunnen worden afgekeurd. (De wijze waarop dit wordt afgedwongen, via een pattern of een enumeration, wordt nog bepaald.) Dezelfde controles worden ook op de elementen voor de versie en de subversie in de Header toegepast.
- Voor XSLT's geldt dat het stylesheet attribuut 'version' al is gereserveerd voor de versie van XSLT die wordt gebruikt. De versie van de stylesheet zelf wordt daarom opgenomen in een commentaarregel.
- In de XSLT's wordt verder geen controle uitgevoerd op de SchemaVersie van het bericht. De XSLT's vormen namelijk samen met de XSD de controle van het bericht. Omdat de XSD al heeft gecontroleerd dat het bericht de juiste versie heeft, is het niet meer nodig dat de XSLT dit doet.
  Opmerking: Het is bij de implementatie van de standaard van belang dat de XSD en XSLT van dezelfde subversie zijn geïmplementeerd. De XSD en XSLT van verschillende subversies hoeven namelijk niet congruent te zijn.

De XSD ziet er bijvoorbeeld als volgt uit:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generieke Declaratie Standaard (gds) -->
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
        xmlns:doc=http://ei.vektis.nl/documentatie
        xmlns:gds801=http://ei.vektis.nl/declaratiebericht573
        targetNamespace=http://ei.vektis.nl/declaratiebericht573
        elementFormDefault="qualified"
        version="1.0.0">
    <xs:element name="Bericht" type="gds801:Bericht573Type">
    <xs:complexType name="Bericht573Type">
        <xs:sequence>
            …
        </xs:sequence>
        <xs:attribute name="SchemaVersie" type="gds801:SchemaVersie"
                use="required"/>
    </xs:complexType>
    <xs:simpleType name="SchemaVersie">
        <xs:restriction base="gds801:StringType">
            <xs:pattern value="[1].[0-1].[0-5]"/>
        </xs:restriction>
    </xs:simpleType>
```

In het XML-bericht komt de versie als volgt terug:

```
<?xml version="1.0" encoding="UTF-8"?>
<gds801:Bericht
    xsi:schemaLocation=http://ei.vektis.nl/declaratiebericht573 gds801_573.xsd
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xmlns:gds801=http://ei.vektis.nl/declaratiebericht573
    schemaVersion="1.0.0">
```

In de XSLT's komt de versie nu als volgt terug:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- gds801_573val_Vecozo_stap1_v1, versie 1.0.0, 14 december 2021 -->
        <xsl:stylesheet version="1.0"
                xmlns:xsl=http://www.w3.org/1999/XSL/Transform
                xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                xmlns:cod=http://ei.vektis.nl/codelijsten
                xmlns:gds801=http://ei.vektis.nl/declaratiebericht573
```

NB: Bovenstaande werkwijze wordt zowel in het heen als in het retourbericht toegepast. Voor het retourbericht geldt echter het volgende:

- Het retourbericht bestaat alleen uit gegevens die door de originele aanleverende partij zijn aangeleverd in het heenbericht. Hierdoor sluiten de gegevens die in het retourbericht zijn opgenomen in principe altijd aan bij de implementatie van de originele aanleverende partij.
- Het retourbericht wordt aangemaakt door een verwerker van het heenbericht. Omdat verwerkers altijd de laatste versie hanteren, waar de aanleverende partijen dit niet hoeven, kan het zijn dat de subversie of uitgave van het retourbericht hoger is dan de subversie of uitgave die de originele aanleverende partij zelf heeft geïmplementeerd.

In de XSD van het retourbericht wordt daarom alleen gecontroleerd of de hoofdversie van het retourbericht overeenkomt. (Er is een klein risico dat het retourbericht toch gegevens bevat die niet aansluiten bij de versie die de originele aanleverende partij heeft geïmplementeerd en daardoor bij de originele aanleverende partij tot foutmeldingen zal leiden. De complexiteit om dit te voorkomen, weegt echter niet op tegen dit risico.)

4.4.1. Bij gebruik basisschema

Een aantal van de XML-standaarden van Vektis maken nog gebruik van een basisschema. Het basisschema heeft een eigen versie die los staat van de versie van de berichten die gebruik maken van het basisschema. De criteria voor het uitgeven van een nieuwe versie, subversie of uitgave van een basisschema zijn gelijk aan de criteria voor XML-standaarden.

Wanneer de berichten in een familie gebruik maken van een gedeeld basisschema, hanteert Vektis voor dit basisschema ook een losse versie. Wijzigingen aan het basisschema die niet voor alle berichten in de familie van toepassing zijn, worden zo doorgevoerd dat het nieuwe basisschema backwards compatible is met eerdere versies

van het basisschema. Hierdoor kunnen berichten waar de wijziging geen impact op heeft, gebruik blijven maken van de oudere versie van het basisschema.

Opmerking:
De technische uitwerking van de versieaanduiding in het basisschema van bestaande XML-standaarden, maakt het onmogelijk om de versie van het basisschema los te beheren van de versies van gerelateerde berichten. Bij de eerst volgende wijziging aan deze standaarden zal een andere versieaanduiding worden geïmplementeerd waardoor de versie van het basisschema wel los komt te staan van de versie van de gerelateerde berichten.

5. Bijlagen

5.1. Mutatieoverzicht

Voor toekomstig gebruik van (sub)versies en uitgaven.

Datum	Documentdeel	Aard wijziging
23-02-2022	Paragraaf 3.1 Opstellen XML Bericht	Principe met betrekking tot de formattering van de XML aangepast.
23-02-2022	Hoofdstuk 4 Versiebeheer	Hoofdstuk toegevoegd.