Standaardisatie

EI-standaarden

Architectuurprincipes

Versie document	2.2
Versie datum	16-01-2025
Uitgave document	1
Uitgave datum:	16-1-2025
Kenmerk:	Architectuur EI-standaarden v2.2u1.pdf

Adres- en contactgegevens

Correspondentie-adres
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

Revisiehistorie Architectuur EI-standaarden document

Versie	Uitgave	Omschrijving	Datum
2.2	1	Versiebeheer verplaatst naar separaat document. Voorwoord, Opbouw retourbericht, Controles en XSLT toegevoegd.	16-01-2024
2.1	1	Versiebeheer beschreven.	23-2-2022
2.0	1	Tweede versie, vervangt XML Schema Definition Architectuurprincipes (v1.3).	15-3-2021

Voorwoord

Dit document beschrijft de architectuurprincipes die Vektis hanteert bij het opstellen van een standaard.

Naast dit document is er ook een beschrijving betreffende Versie- en Releasebeheer, (specifiek voor de GDS801-GDS802) en een beschrijving betreffende het releaseproces van RFC's.

Al deze documenten zijn te vinden op de Vektis website.

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 (paragraaf 4.2). 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 prestatiecodelijsten ~~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 prestatiecodelijsten ~~zorgsoorten~~ heen.
  Binnen verschillende domeinen en prestatiecodelijsten~~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 2.6).

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.
Hoofdstuk 4 beschrijft de in dit document gehanteerde afkortingen en een mutatieoverzicht.

~~1.4 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 berichtrouteringen, 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.

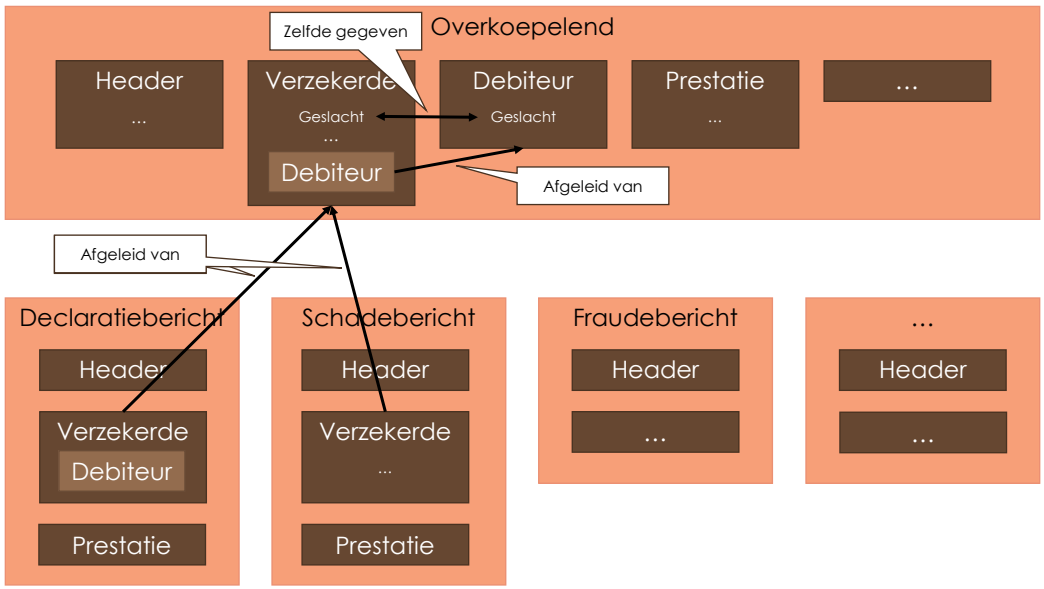
2.1. Generiek opstellen berichten

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 gegevens (blok) 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 Zorgaanbiederscode en ZorgaanbiederSoort. 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 prestatiecodelijst ~~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 prestatiecodelijsten ~~zorgsoorten~~ kunnen worden opgenomen compact (in plaats van voor elke prestatiecodelijst ~~zorgsoort~~ een apart veld opnemen, volstaat één veld).

8.	Prestatiecodelijst ~~Zorgsoort~~ specifieke informatie wordt vastgelegd in de gegevensblokken die de zorg beschrijven. Voorbeeld: Met betrekking tot declaraties wordt de prestatiecodelijst ~~zorgsoort~~ specifieke informatie vastgelegd bij de prestatie. Toelichting: Door de prestatiecodelijst ~~zorgsoort~~ specifieke informatie vast te leggen bij het gegevensblok dat de zorg beschrijft, kunnen andere gegevensblokken eenvoudiger prestatiecodelijst ~~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 terugkomen.
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 prestatiecodelijsten ~~zorgsoorten~~ declaratieberichten worden verstuurd. Al deze berichten hebben als doel de geleverde zorg bij de debiteur te declareren. In plaats van een declaratiebericht per prestatiecodelijst~~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~~ prestatiecodelijsten van toepassing zijn, geeft dit ook de mogelijkheid om hun gegevens in te dienen via één bericht (in plaats van een bericht per prestatiecodelijst~~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. <u>Voorbeeld:</u> 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. <u>Toelichting:</u> 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. <u>Voorbeeld:</u> 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'. <u>Toelichting:</u> 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 plus (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. Toelichting: 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. Voorbeeld: Voor sommige prestatiecodelijsten ~~zorgsoorten~~ is bij een prestatie alleen de datum van de prestatie relevant, terwijl voor andere prestatiecodelijsten ~~zorgsoorten~~ zowel de datum als de tijd van belang zijn. Voor prestatie worden daarom een losse prestatiedatum en prestatie tijd opgenomen. Toelichting: 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. Opbouw retourbericht

Voor de opbouw van het retourbericht, hanteert Vektis onderstaande principes.

1.	**Het retourbericht heeft een 3-ledige functie. Nadat een bericht op structuur is gecontroleerd middels de XSD volgt de inhoudelijke controle waarbij als resultaat een retourbericht wordt gestuurd.**
2.	In het retourbericht kunnen naast gegevens uit het heenbericht, de volgende gegevens opgenomen zijn: 1. Retourcodes als gevolg van de bedrijfs- en controleregels zoals vastgelegd in de RBC 2. Retourcodes als gevolg van aanvullende controles (zoals bijvoorbeeld de LCB controles (formele controles) en verzekeraarspecifieke controles op bijvoorbeeld polisvoorwaarden) door de ontvanger. 3. Antwoordinformatie voor de oorspronkelijke verzender. Dit zijn elementen die in het heenbericht niet aanwezig zijn en dienen om informatie te geven aan de verzender van het heenbericht over de inhoudelijke verwerking hiervan door de ontvanger van het heenbericht.
3.	De klasse Header bevat informatie over het bericht zelf. • De inhoud is specifiek voor het retourbericht zelf. In de header wordt gerefereerd aan het bijbehorende heenbericht. • Een retourcode wordt opgenomen in een feedbackcontainer. • Aan de header kan een of meer keer een feedbackcontainer worden toegevoegd.
4.	De klassen Context en Overzicht uit het heenbericht worden in hun geheel in het retourbericht opgenomen, met de waarden uit het heenbericht. • Aan beide klassen kan antwoordinformatie toegevoegd worden, zoals bijvoorbeeld declaratienummer verzekeraar in de klasse Context en bijvoorbeeld Totaal ToegekendBedragInclBtw in de klasse Overzicht. • Een retourcode wordt opgenomen in een feedbackcontainer. • Aan de Context of Overzicht kan een of meer keer een feedbackcontainer worden toegevoegd.
5.	Van alle overige klassen wordt alleen het identificerende pad opgenomen, indien er een retourcode in die klasse (of onderliggende klasse) wordt opgenomen. • Deze retourcode wordt opgenomen in een feedbackcontainer. • De feedbackcontainer kan meerder keren in een klasse worden opgenomen (per retourcode een feedbackcontainer). • Een feedbackcontainer is hier dus altijd voorzien van een identificerende pad.
6.	De feedbackcontainer kan naast de retourcode aanvullende informatie bevatten:

- 0 of meer betrokken elementen
  Een BetrokkenElement geeft de inhoud weer van de elementen uit het bericht die betrokken zijn bij het uitvoeren van de controle.
- 0 of meer aanvullende kenmerken.
  Bij het uitvoeren van formele (LCB-) controles kan ook informatie betrokken worden van buiten het bericht die noodzakelijk was voor de controle. Deze informatie wordt met de feedback meegestuurd middels één of meer aanvullende kenmerken.

~~2.5.~~2.6. 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 omgegaan 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~~2.6.1. Wanneer gebruik maken van een ZIB

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

1.	Uitgangspunt is om bij het opstellen van generieke gegevensblokken zo veel mogelijk gebruik te maken van de ZIB's.
2.	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.
3.	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.
4.	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.~~2.6.2. Hoe een ZIB te gebruiken

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

1.	Gegevensblokken hoeven niet één op één met een ZIB overeen te komen (zie ook principe 3). 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.
2.	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
3.	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.

4.	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 • …
5.	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. Toelichting: 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.
6.	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 2.7 niet tegenspreekt. Deze codelijst wordt dan ook opgenomen in het bericht.
7.	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. Toelichting: 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.
8.	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.
9.	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. Toelichting: 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.
10.	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.
11.	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.~~2.7. Gebruik codelijsten

1.	Voor gegevens waar over alle prestatiecodelijsten ~~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 **prestatiecodelijsten** ~~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 prestatiecodelijsten ~~zorgsoorten~~ verschillende codes gebruikt kunnen worden, worden verschillende codelijsten per prestatiecodelijst ~~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 prestatiecodelijsten ~~zorgsoort~~ waarvoor een gegeven is ingevuld. Voorbeeld: De diagnose is afhankelijk van de prestatiecodelijst~~zorgsoort~~. Voor de diagnose bestaan daarom codelijsten per prestatiecodelijst~~zorgsoort~~.
4.	Codes hebben altijd een begindatum. Wanneer een code niet meer geldig is, heeft de code een expiratiedatum. Toelichting:
5.	Codes kunnen gecontroleerd worden op bestaanbaarheid en op geldigheid. Toelichting: Voor statische codelijsten zoals soort bericht (P of T) is controle op geldigheid niet zinvol. Soms is voor een codelijst geen geschikte peildatum in het bericht aanwezig en wordt alleen op bestaanbaarheid gecontroleerd.
6.	Er wordt alleen op geldigheid van codes gecontroleerd, niet op codelijsten. Toelichting:

	Voor codelijsten wordt ook een ingangsdatum en expiratiedatum opgegeven, echter wordt niet op geldigheid van een codelijst gecontroleerd. Dat gebeurt alleen op de codes in de lijst zelf. Dat betekent dat als een codelijst expireert, alle codes van die lijst een expiratiedatum krijgen.
7.	Per standaard worden de gebruikte codelijsten aangegeven in de berichtspecificatie en op de website.
~~5.~~8.	~~**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.~~2.8. 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: Diagnose~~Agb~~Code
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 bericht~~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 3.3), 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="gds801: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="gds801:Header573Type"/>
      <xs:element name="Overzicht" type="gds801:Overzicht573Type"/>
      <xs:element name="Verzekerde" type="gds801: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="gds801: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 El-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="gds801:NumeriekType"/> <xs:element name="VerzenderRol" type="gds801:NumeriekType"/> We maken dus geen simpleType VerzenderType en VerzenderRolType. <u>Toelichting:</u> 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. <u>Toelichting:</u>

	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 hiermee rekening 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="gds801: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="gds801: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.

3.4. Controles

Vektis hanteert onderstaande principes voor het controleren van XML berichten.

1.	**Vektis definieert een xsd op basis van de berichtspecificatie.**
2.	VECOZO voert de xsd controle uit met behulp van de door Vektis opgeleverde XSD.
3.	In het geval VECOZO fouten constateert dat het xml bericht niet voldoet aan de XSD, dan worden deze fouten door VECOZO via een foutberciht aan de inzender teruggekoppeld.
4.	Vektis definieert referentiële controles en verbandcontroles (XSLT's).
5.	Met de XSLT levert Vektis één of meer testbestanden op voor elke verbandscontrole die is gedefinieerd voor de berichtstandaard.
6.	Met behulp van de door Vektis opgeleverde XLST's voert VECOZO de volgende controles uit op de ontvangen xml-bestanden: • referentiële controles; • verbandscontroles.
7.	In het geval VECOZO fouten constateert, dan worden deze fouten door VECOZO via het retourbericht aan de indiener teruggekoppeld. Als verzender van het retourbericht zorgt VECOZO ervoor dat het bericht voldoet aan de xsd en de verbandscontroles van het retourbericht.
8.	Fouten worden teruggekoppeld middels een retourcode die wordt opgenomen in de feedbackcontainer (zie 2.5). Een retourcode heeft een unieke betekenis en zou voldoende informatie moeten aanreiken om de fout te herstellen.
9.	In het geval VECOZO geen fouten constateert, dan wordt het oorspronkelijke XML bericht ongewijzigd doorgestuurd.
10.	Zorgverzekeraars definiëren en voeren inhoudelijke controles uit en koppelen het resultaat van deze controles terug naar de indiener van het bericht. Als verzender van het retourbericht zorgt de zorgverzekeraar ervoor dat het bericht voldoet aan de xsd en de verbandscontroles van het retourbericht.

3.5. Opbouw en inhoud XSLT

Vektis hanteert onderstaande principes voor de toepassing van XSLT.

1.	XSLT is het instrument voor de controle van XML berichten en de transformatie naar retourberichten.
2.	Een XSLT module is gerelateerd aan een specifieke berichtstandaard.
3.	De XSLT maakt gebruik van versie 1.0 (xsl:stylesheet version="1.0").
4.	De XSLT maakt gebruik van het bestand config_Vecozo.xml dat variabelen bevat die niet in het te controleren XML bericht voorkomen. Denk aan 'OmgevingVecozo' en 'Huidige datum' op basis waarvan de actieve VECOZO omgeving (test of productie) respectievelijk de systeemdatum wordt bepaald.
5.	De XSLT maakt voor de referentiele controles gebruik van de codelijsten in xml formaat (zie 2.7). Deze codelijsten moeten zich bevinden in een subdirectory 'Codelijsten' ten opzicht van de locatie van de XSLT's.
6.	De XSLT transformeert het XML bericht naar een retourbericht aangevuld met terugkoppeling van de fouten in het ontvangen XML bericht. N.B. Als het resultaat van de transformatie een retourbericht zonder fouten is, dan is dit voor VECOZO de trigger om het oorspronkelijke XML bericht door te sturen.
7.	Een retourbericht kan in twee varianten bestaan: 1. Als één of meer fouten zijn gevonden in de klassen met algemene gegevens (header, context, overzicht), dan bestaat het retourbericht uit die klassen met terugkoppeling van de gevonden fouten. 2. Als geen fouten zijn gevonden in de algemene gegevens maar wel in de klassen met de detailinformatie, dan transformeert de XSLT het XML bericht naar een retourbericht met de algemene gegevens aangevuld met de onderdelen van het XML bericht waarin één of meer fouten zijn gevonden.
8.	De transformatie van het XML bericht tot één van de hierboven beschreven varianten van het retourbericht is opgesplitst in 2 stappen: 1. Een XSLT die het XML bericht kopieert (binnen de beperkingen die zijn gesteld door de XSD van het retourbericht) en verrijkt met de informatie over gevonden fouten. N.B. De elementen Verzender, VerzenderRol, Ontvanger, OntvangerRol en Verzenddatum worden niet gekopieerd, maar aangepast aan de situatie die geldt voor het retourbericht. 2. Een XSLT die het resultaat van de eerste stap transformeert tot een retourbericht waarin de onderdelen die geen fouten bevatten (en niet verplicht zijn in het retourbericht), zijn verwijderd.
9.	Voor de referentiële controles en verbandscontroles zijn generieke xslt routines ontwikkeld. Door het gebruik van parameters worden deze ingezet voor het uitvoeren van een specifieke controle. In geval van een fout retourneren deze routines in feite een stukje retourbericht met daarin de feedback met betrekking tot de fout.

	Let op: dit stukje retourbericht wordt geretourneerd met namespace ‘gen’; in stap 2 van de transformatie wordt deze namespace omgezet naar de namespace die geldt voor het retourbericht.
10.	Als een klasse niet verplicht voorkomt in het retourbericht en er geen referentiële controles en verbandscontroles bestaan voor de klasse, wordt de klasse in stap 1 van de transformatie niet gekopieerd.

11.	In stap 2 van de transformatie wordt in het retourbericht opgenomen welke XSD van toepassing is voor het retourbericht. Dit ziet er dan uit als onderstaand.
	`<fz822:Bericht>`
	`<!-- neem in retourbericht op volgens welke XSD het retourbericht is opgebouwd -->`
	`<xsl:attribute name="xsi:schemaLocation">http://ei.vektis.nl/fz/fz822_479 fz822_479.xsd</xsl:attribute>`
	`<xsl:apply-templates select="fz822:Header"/>`
	`<xsl:apply-templates select="fz822:PlaatsingsContext"/>`
	`<xsl:apply-templates select="fz822:Verzekerde"/>`
	`</fz822:Bericht>`
	`</xsl:template>`

4. Bijlagen

4.1. Afkortingen

De lijst met afkortingen heeft betrekking op dit architectuurdocument.

Tabel 4-1 Lijst met afkortingen architectuurdocument

Afkorting	Betekenis
ASCII	American Standard Code for Information Interchange
BER	Berichtspecificatie
BOM	Byte Order Mark
CD	Conditie
CR/LF	Carriage Return/Line Feed
CS	Constraint
EI	Externe integratie
EESSI	Electronic Exchange of Social Security Information
FZ	Forensische zorg
GDS	Generieke Declaratiestandaard
ggz	Geestelijke gezondheidszorg
GPH	Generieke productcode hulpmiddelen
INV	Invulinstructie
OAF	Operationele afspraken
RBC	Registratie bedrijfs- en controleregels
RfC	Request for Change
STB	Standaardbeschrijving
URI	Uniform Resource Identifier
UTF-8	8-bit Unicode Transformation Format
UTF-16	16-bit Unicode Transformation Format
VECOZO	VEilige COmmunicatie in de ZOrg
W3C	World Wide Web Consortium

Afkorting	Betekenis
Wlz	Wet langdurige zorg
WMO	Wet maatschappelijke ondersteuning
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations
ZIB	Zorginformatiebouwstenen
ZN	Zorgverzekeraars Nederland
zpm	Zorgprestatiemodel
Zvw	Zorgverzekeringswet

~~4.1.~~4.2. Mutatieoverzicht

Voor toekomstig gebruik van (sub)versies en uitgaven.

Tabel 4-2 Mutatieoverzicht

Datum	Documentdeel	Aard wijziging
16-01-2024	Hele document	Zorgsoort vervangen door prestatiecodelijst
16-01-2024	2.5 Opbouw retourbericht	Paragraaf toegevoegd.
16-01-2024	2.7 Codelijsten	Punt 5 en 6 toegevoegd en opmerking 'Nog uit te werken' verwijderd..
16-01-2024	3.4 Controle	Paragraaf toegevoegd.
16-01-2024	3.5 Opbouw en inhoud XSLT	Paragraaf toegevoegd.
16-01-2024	4.1 Afkortingen	Paragraaf toegevoegd.
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.