Zorginstituut Nederland

Ontwerprichtlijnen voor XML-Schemadefinities

Voor gebruik binnen iWlz, iWmo en iJw

Datum 1 juni 2016
Status Definitief

Colofon

Publicatienummer	
Uitgave	Extra exemplaren kunt u downloaden vanaf www.zorginstituutnederland.nl.
Projectnaam	
Projectnummer	
Versienummer	1.3
Projectleider	
Volgnummer	2015063694
Opdracht	
Opdrachtgever	
Opdrachtnemer	
Locatie	
Contactpersoon	
Auteur(s)	
Afdeling	Contactcentrum Zakelijk
Team	
Documenteigenaar	Zorginstituut Nederland
Bijlage(n)	

Versiebeheer

Versie	Datum	Toelichting en status
1.1	26 mei 2015	Eerste concept
1.2	1 februari 2016	Aanscherping regels opstellen XML
1.3	1 juni 2016	Definitief voor release iWlz 1.2, iWmo 2.1 en iJw 2.1

Inhoud

Inleiding

Vanaf 1 januari 2016 zijn ketenpartijen voor uitvoering van de Wet Langdurige Zorg (Wlz) verplicht om gebruik te maken van de XML-standaard binnen het iWlz berichtenverkeer. Binnen de andere twee zorgdomeinen Wet Maatschappelijke Ondersteuning (Wmo) en Jeugdwet (Jw) is dit per 1 april 2017. Tot die tijd moeten partijen ook berichten in het EI-formaat kunnen aanbieden en ontvangen.

Binnen de drie zorgdomeinen worden XML schemadefinities gebruikt voor het definiëren van berichtdefinities. De W3C specificaties van XML Schemadefinities zijn echter omvangrijk en complex waardoor het bijna ondoenlijk is voor een persoon deze tot in detail te kennen. Daarnaast biedt de W3C standaard geen handreiking op het gebied van best-practices of richtlijnen voor het implementeren van XML-schemadefinities.

Het doel van dit document is het definiëren van high-level best-practices en richtlijnen die binnen de drie zorgdomeinen gevolgd zullen worden om consistentie binnen alle XML-schemadefinities te bewerkstelligen. Deze XML-schemadefinities definiëren gegevenssets die tussen ketenpartners kunnen worden uitgewisseld.

Omdat de W3C XML-Schemadefinitie specificatie zich blijft ontwikkelen en steeds meer volwassen wordt, zal ook dit document zich blijven ontwikkelen. Periodiek zal dit document daarom worden bijgewerkt met de laatste inzichten.

1 Namespaces

De volgende paragrafen geven richtlijnen voor het definiëren van namespaces binnen XML-schemadefinities. Nadat de richtlijnen zijn beschreven, wordt een gedeelte van een XML schemadefinitie als voorbeeld gepresenteerd.

Omdat de overgang naar XML niet voor alle zorgdomeinen gelijk is heeft elk domein een eigen namespace. Dit gaat in de toekomst mogelijk veranderen.

1.1 Target namespace

De XML-schemadefinitie moet een "target namespace" definiëren. Deze namespace moet gedefinieerd worden als een URL die dit schema en de definities daarin uniek identificeert.

Vermijd het XML schemadefinities zonder "target namespace" ("chameleon"). Hoewel "chameleon" schema's flexibiliteit bieden, heeft het een negatieve invloed op de performance. De meeste parsers zijn bij "chameleon" schema's namelijk niet in staat om de componenten van het schema te cachen op basis van de namespace. Daarnaast moet er zeer voorzichtig met "chameleon" schema's worden omgegaan om naamconflicten te voorkomen bij het importeren van schemadefinities.

Elke XML-schemadefinitie binnen Wlz heeft een eigen namespace. Hiermee worden naamgevingsconflicten voorkomen met schema's die geïmporteerd of "geinclude" worden in andere schema's. De namespace is een URL die wordt opgebouwd uit een hiërarchie, waarin het Wlz versienummer, het Wlz-bericht en het versienummer van de berichtdefinitie is opgenomen.

De root-URL voor de Wlz namespace is http://www.istandaarden.nl/iwlz. Deze wordt gevolgd door een Wlz versienummer, een Wlz berichtidentificatie en een versienummer voor het Wlz-bericht.

Voorbeeld:

```
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/IO31/1_0/"

Wlz-versie 1.1; berichtindentificatie IO31; berichtversie 1.0
```

De namespace URL kan een locatie betreffen waar de schemadefinitie en –specificaties gepubliceerd zijn, maar dit is niet gegarandeerd.

1.2 Default namespace

De XML-schemadefinitie moet een "default namespace" definiëren die gelijk is aan de "target namespace". Met andere woorden: "XML Schema" (standaard) moet niet de "default namespace" zijn.

Voorbeeld:

```
xmlns="http://www.istandaarden.nl/iwlz/1_1/IO31/1_0/"
```

1.3 Prefix

1.3.1 elementFormDefault en attributeFormDefault

Wanneer een XML-schemadefinitie wordt opgesteld, kunnen er een tweetal attributen worden gedefinieerd (elementFormDefault en attributeFormDefault)

waarvan de impact pas zichtbaar is wanneer een XML-bestand conform de XML_schemadefinitie wordt samengesteld.

De <schema> element attributen **elementFormDefault** en **attributeFormDefault** geven aan of voor de lokale elementen en attributen van het XML bestand expliciet moet worden aangegeven in welke namespace deze zijn gedefinieerd. Met de waarde "qualified" wordt aangegeven dat voor de lokale elementen en attributen van het XML bestand expliciet aangegeven hoeft te worden in welke namespace deze zijn gedefinieerd. Dit gaat door middel van een namespace prefix. Default staan deze attributen op "unqualified".

Voor de XML-schemadefinities is gekozen voor de volgende standaard:
- **elementFormDefault** moet op "qualified" staan. Voor alle elementen moet daarmee expliciet aangegeven worden in welke namespace ze zijn gedefinieerd (prefix aangeven). Deze keuze is gemaakt omdat het de leesbaarheid en de performance van het verwerken van de XML-bestanden verbetert.
- **attributeFormDefault** moet op "unqualified" staan. Voor alle attributen mag niet expliciet aangegeven worden in welke namespace ze zijn gedefinieerd (geen prefix). Dit is de defacto standaard voor XML-schemadefinities.

Omdat de default waarde "unqualified" is, wordt het attributeFormDefault attribuut niet gedefinieerd in de XML-schemadefinities. Het elementFormDefault attribuut wordt expliciet op "qualified" gezet.

Voorbeeld XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:iwlz="http://www.
istandaarden.nl/iwlz/1_0/basisschema/schema/1_0" xmlns:aw310="http://www.
istandaarden.nl/iwlz/1_0/aw310/schema/1_0" targetNamespace="http://www.
istandaarden.nl/iwlz/1_0/aw310/schema/1_0" elementFormDefault="qualified">
<xs:import
namespace="http://www.istandaarden.nl/iwlz/1_0/basisschema/schema/1_0"
schemaLocation="basisschema.xsd"/>
…
```

Voorbeeld XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<aw310:AW310Bericht codeExterneIntegratiebericht="010“
xmlns:aw310="http://www.istandaarden.nl/iwlz/1_0/AW310/schema/1_0"
xmlns:p="http://www.istandaarden.nl/iwlz/1_0/Basisschema/schema/1_0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <aw310:Clienten>
    <aw310:Client>
      <aw310:Bsn>12345678910</aw310:Bsn>
      <aw310:Clientnummer>987654321</Clientnummer>
      <aw310:CodeIndicatieorgaan>0100</aw310:CodeIndicatieorgaan>
    </aw310:Client>
  </aw310:Clienten>
</aw310:AW310Bericht>
```

1.3.2 xs-prefix

Binnen de XML schemadefinitie, moet de namespace prefix voor XML-schema **xs** zijn.

Voorbeeld:

```
<schema targetNamespace=http://www.istandaarden.nl/iwlz/1_0/AW310/schema/1_0
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
xmlns:aw310="http://www.istandaarden.nl/iwlz/1_0/AW310/schema/1_0"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_0/Basisschema/schema/1_0">
```

1.3.3 basisschema-prefix

Binnen de XML-schemadefinities van de drie zorgdomeinen is er voor gekozen om alle generieke typedefinities (simple en complex) binnen één zorgdomein in één XML-schemadefinitie te definiëren. Generiek wil in dit geval zeggen dat de definitie van een type binnen het domein over alle berichten heen gelijk zijn. Er zijn momenteel dus drie basisschema's. Een Wmo basisschema, Jw basisschema en een Wlz Basisschema. In dit laatstgenoemde XML-schemadefinitie worden alle Wlz generieke typedefinities (simple en complex) op een eenduidige manier en centrale plaats gedefinieerd en beheerd. En gelden dus voor alle Wlz berichtdefinities. De typedefinities worden in de schemadefinities van de berichten geïmporteerd en gebruikt. Al deze geïmporteerde generieke Wlz typeberichtdefinities hebben 'iwlz' als prefix.

Voorbeeld:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
xmlns:aw33="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
elementFormDefault="qualified">
<xs:import
namespace="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
schemaLocation="basisschema.xsd"/>
   <xs:element name="AW33Bericht" type="aw33:AW33Bericht"/>
   <xs:complexType name="AW33Bericht">
     <xs:sequence>
          <xs:element name="Client" type="aw33:Client"/>
     </xs:sequence>
    <xs:attribute name="BerichtCode" type="iwlz:BerichtCode" use="required"/>
    <xs:attribute name="BerichtVersie" type="iwlz:BerichtVersie" use="required"/>
   </xs:complexType>
   <xs:complexType name="Client">
       <xs:sequence>
          <xs:element name="Bsn" type="iwlz:BurgerServicenummer" />
          <xs:element name="CizCode" type="iwlz:CizCode" minOccurs="0" />
          <xs:element name="Clientnummer" type="iwlz:Persoonsid" />
       </xs:sequence>
   </xs:complexType>
</schema>
```

1.3.4 bericht-prefixes

Iedere XML schemadefinitie van een bericht heeft een eigen namespace (waarin de berichtnaam is opgenomen, bijv.:
"http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0" met een eigen prefix (de afkorting van het bericht, bijv: 'aw33'). Daarmee wordt aangegeven dat de gedefinieerde typen in de XML schemadefinitie van het bericht specifiek zijn voor het betreffende bericht. Tevens worden problemen voorkomen indien typedefinities in de toekomst toch worden geïmporteerd in andere XML-schemadefinities.

Voorbeeld:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
xmlns:aw33="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
elementFormDefault="qualified">
<xs:import
namespace="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
schemaLocation="basisschema.xsd"/>
   <xs:element name="AW33Bericht" type="aw33:AW33Bericht"/>
   <xs:complexType name="AW33Bericht">
  <xs:sequence>
          <xs:element name="Client" type="aw33:Client"/>
  </xs:sequence>
   </xs:complexType>
   <xs:complexType name="Client">
       <xs:sequence>
     <xs:element name="Bsn" type="iwlz:BurgerServicenummer" />
  </xs:sequence>
   </xs:complexType>
</schema>
```

1.4 Versionering

De "default namespace" en "target namespace" bevatten beide een tweetal versienummers die beide uit een major- en minor-versienummer zijn opgebouwd. (zie ook) Het major- en minor-versienummer worden gescheiden door een underscore (_). Het eerste versienummer in de "default namespace" en "target namespace" betreft het zorgdomein versienummer. Het tweede versienummer (bijv. 1_0), betreft het versienummer van de XML-schemadefinitie. Wanneer een nieuwe Wlz release wordt uitgebracht (eerste versienummer) of wanneer er een nieuwe versie van de XML-schemadefinitie (tweede versienummer) wordt vastgesteld, worden de versienummers in de "default namespace" en "target namespace" aangepast.

De huidige versie van de Wlz is 1.0. De initiële versie van de XML-schemadefinities is **1_0**.

Voorbeeld:

```
targetNamespace=http://www.istandaarden.nl/iwlz/1_1/AW33/1_0/
```

In het algemeen zullen de versienummers van het zorgdomein en de XML-schemadefinities met elkaar overeenkomen. Een xsd bevat om die reden ook een releasedatum om verschil te kunnen aanduiden tussen een eventuele nieuwe versie als gevolg van een correctie binnen eenzelfde release

```
<!-- iWlz basisschema 1.1 - 25-11-2015 -->
```

1.5 Voorbeeld

Het volgende XML voorbeeld omvat alle boven genoemde richtlijnen.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
```

```
xmlns:aw33="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
elementFormDefault="qualified">
<xs:import
namespace="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
schemaLocation="basisschema.xsd"/>
   <xs:element name="AW33Bericht" type="aw33:AW33Bericht"/>
   <xs:complexType name="AW33Bericht">
  <xs:sequence>
          <xs:element name="Client" type="aw33:Client"/>
  </xs:sequence>
   </xs:complexType>
   <xs:complexType name="Client">
       <xs:sequence>
      <xs:element name="Bsn" type="iwlz:BurgerServicenummer" />
  </xs:sequence>
   </xs:complexType>
</schema>
```

2 Design Pattern

2.1 Inleiding

Net als bij softwareontwikkeling, zijn er design patterns voor het ontwerpen van XML-schemadefinities. De meest populaire XML-schemadefinitie patterns zijn Russian Doll, Salami, Bologna, Venetian Blind, en Garden of Eden[1]. Deze patterns hebben allen eigen voor- en nadelen. Voor de iStandaarden XML-schemadefinities is gekozen voor het Venetian Blind design-pattern.

2.2 Venetian Blind

In het Venetian Blind design pattern is er in de XML-schemadefinitie één enkel globaal element gedefinieerd (waarbinnen andere lokale elementen genest zijn). Dit globale element wordt het "root" element genoemd en wordt gedefinieerd in de globale namespace. De lokale elementen worden gedefinieerd op basis van typedefinities (complex en simple).

Voordelen Venetian Blind pattern:

- Herbruikbaarheid wordt gestimuleerd, doordat typedefinities (simple en complex) eenvoudig in andere XML-schemadefinities kunnen worden geïmporteerd.
- Naamgevingsconflicten worden voorkomen, doordat de typedefinities (simple en complex) in een eigen bericht specifieke namespace worden gedefinieerd.
- Omdat er maar één root element is, is er ook maar één geldige vorm van het XML document.

Nadelen Venetian Blind pattern:

- Er zijn geen grote nadelen bij het gebruik van het Venetian Blind pattern.

Het voordeel van herbruikbaarheid in combinatie met slechts één root-element, maakt dat het Venetian Blind pattern als best-practice is aangemerkt voor de XML-schemadefinities.

Voorbeeld:

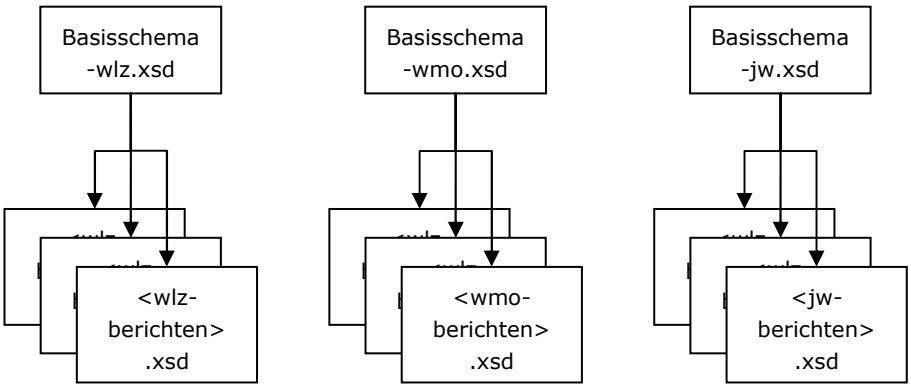
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
xmlns:aw33="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
elementFormDefault="qualified">
<xs:import
namespace="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
schemaLocation="basisschema.xsd"/>
  <xs:element name="AW33Bericht" type="aw33:AW33Bericht"/>
  <xs:complexType name="AW33Bericht">
   <xs:sequence>
     <xs:element name="Client" type="aw33:Client"/>
   </xs:sequence>
   <xs:attribute name="BerichtCode" type="iwlz:BerichtCode" use="required"/>
   <xs:attribute name="BerichtVersie" type="iwlz:BerichtVersie" use="required"/>
  </xs:complexType>
  <xs:complexType name="Client">
```

[1] http://www.xfront.com/GlobalVersusLocal.html

```
   <xs:sequence>
     <xs:element name="Bsn" type="iwlz:BurgerServicenummer" />
     <xs:element name="CizCode" type="iwlz:CizCode" minOccurs="0" />
     <xs:element name="Clientnummer" type="iwlz:Persoonsid" />
   </xs:sequence>
  </xs:complexType>
</schema>
```

2.3 Basisschema

Het Venetian Blind design pattern maakt het mogelijk om eenvoudig typedefinities (complex en simple) in andere XML-schemadefinities te hergebruiken. Binnen de XML-schemadefinities is er voor gekozen om alle generieke[2] typedefinities (simple en complex) in één XML-schemadefinitie te definiëren: het Basisschema. In deze XML-schemadefinitie worden alle generieke typedefinities (simple en complex) op een eenduidige manier en centrale plaats gedefinieerd en beheerd. Daarmee wordt het beheer vereenvoudigd en wordt voor alle berichten een eenduidige definitie van elementen nagestreefd. Het Basisschema wordt middels een "import" statement in de verschillende XML-schemadefinities geïmporteerd. De typedefinities uit het basisschema behouden daarbij een eigen namespace en worden in de XML-schemdefinitie van het bericht gemarkeerd door een zorgdomein specifieke prefix. Bijvoorbeeld: Voor het Wlz domein is de prefix: iwlz



Voorbeeld Wlz Basissschema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
elementFormDefault="qualified">
    <xs:simpleType name="BerichtCode">
        <xs:restriction base="xs:integer">
            <xs:pattern value="[0-9]{3}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="BerichtVersie">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="99"/>
        </xs:restriction>
```

[2] Generiek wil zeggen dat de definitie van het type over alle berichten heen gelijk zijn. Dit geldt onder andere voor veel simple typedefinities in WLZ met codelijsten (toegestane waarden).

```
    </xs:simpleType>
    <xs:simpleType name="BurgerServicenummer">
        <xs:restriction base="xs:integer">
            <xs:pattern value="[0-9]{9}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CizCode">
        <xs:restriction base="xs:integer">
            <xs:pattern value="[0-9]{4}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Persoonsid">
        <xs:restriction base="xs:string">
            <xs:maxLength value="20"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Voorbeeld import en gebruik Wlz-Basisschema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_1"
xmlns:aw33="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_1"
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_1"
elementFormDefault="qualified">
<xs:import
namespace="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_1"
schemaLocation="basisschema.xsd"/>
   <xs:element name="AW33Bericht" type="aw33:AW33Bericht"/>
   <xs:complexType name="AW33Bericht">
      <xs:sequence>
        <xs:element name="Client" type="aw33:Client"/>
      </xs:sequence>
      <xs:attribute name="BerichtCode" type="iwlz:BerichtCode" use="required"/>
      <xs:attribute name="BerichtVersie" type="iwlz:BerichtVersie"
use="required"/>
   </xs:complexType>
   <xs:complexType name="Client">
      <xs:sequence>
        <xs:element name="Bsn" type="iwlz:BurgerServicenummer" />
        <xs:element name="CizCode" type="iwlz:CizCode" minOccurs="0" />
        <xs:element name="Clientnummer" type="iwlz:Persoonsid" />
      </xs:sequence>
   </xs:complexType>
</schema>
```

3 Overige richtlijnen

In dit hoofdstuk worden een aantal richtlijnen gepresenteerd die niet in één van de voorgaande categorieën vallen.

3.1 Richtlijnen elementen en attributen

De richtlijn voor het gebruik van elementen en attributen is om elementen te gebruiken voor gegevens die door applicaties worden verwerkt en attributen voor metadata. Binnen de Wlz, Wmo en Jw XML-schemdefinities worden attributen gebruikt om metadata van het betreffende bericht vast te leggen en worden elementen gebruikt om de inhoudelijke data van het bericht vast te leggen.

3.2 Naamgevingsrichtlijnen

De naamgevingsrichtlijnen zijn gebaseerd op de richtlijnen van ebXML (http://www.oasis-open.org/). Onderstaand overzicht toont de vastgestelde naamgevingsrichtlijnen voor de XML-schemadefinities binnen de drie zorgdomeinen.

Richtlijn	Omschrijving	Voorbeeld
Hoofdlettergebruik in element- en typedefinities	Element- en typedefinities worden gedefinieerd op basis van *Upper Camel Case*.	*<VoorvoegselClient>*
Hoofdlettergebruik in attribuutdefinities	Attribuutdefinities worden gedefinieerd op basis van *Lower Camel Case*.	*<aw310:AW310Bericht codeExterneIntegratiebericht= "010">*
Acroniemen	Acroniemen worden zoveel mogelijk vermeden, maar wanneer ze worden toegepast dan worden hoofdletters gebruikt.	<clientID>
Consistente naamgeving	Zorg voor een consistente naamgeving.	<WoonAdres> <ContactAdres>
Meervoudsaanduidingen	Gebruik enkel meervoudsaanduidingen voor verzamelingen van elementen.	<Adressen>
Lengte naamgeving	Er is geen beperking op de lengte van de naamgeving. De naamgeving moet zinvol zijn.	<MutatieZorgzwaartepakket>

3.3 Richtlijnen typedefinities

De volgende richtlijnen zijn van toepassing op "simple" en "complex" typedefinities. Typedefinities moeten altijd in de globale namespace worden gedefinieerd en vervolgens gebruikt worden door de lokale elementen. Deze regel ondersteunt het Venetian Blind design pattern. Dit heeft ook tot gevolg dat typedefinities van een naam voorzien moeten worden in plaats van gebruik te maken van anonieme typedefinities.

Voorbeeld:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iwlz="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
xmlns:aw33="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
targetNamespace="http://www.istandaarden.nl/iwlz/1_1/aw33/schema/1_0"
elementFormDefault="qualified">
<xs:import namespace="http://www.istandaarden.nl/iwlz/1_1/basisschema/schema/1_0"
schemaLocation="basisschema.xsd"/>
   <xs:element name="AW33Bericht" type="aw33:AW33Bericht"/>
   <xs:complexType name="AW33Bericht">
      <xs:sequence>
          <xs:element name="Client" type="aw33:Client"/>
      </xs:sequence>
      <xs:attribute name="BerichtCode" type="iwlz:BerichtCode" use="required"/>
      <xs:attribute name="BerichtVersie" type="iwlz:BerichtVersie" use="required" />
   </xs:complexType>
   <xs:complexType name="Client">
       <xs:sequence>
          <xs:element name="Bsn" type="iwlz:BurgerServicenummer" />
          <xs:element name="CizCode" type="iwlz:CizCode" minOccurs="0" />
          <xs:element name="Clientnummer" type="iwlz:Persoonsid" />
       </xs:sequence>
   </xs:complexType>
</schema>
```

3.4 Annotations

Annotations zijn een special mechanisme om XML-schemadefinities te documenteren. Verschillende tools kunnen op basis van deze ingebedde documentatie overzichtlijke documentatie van de XML-schemadefinitie genereren. De XML-schemadefinities worden zoveel mogelijk met behulp van annotations gedocumenteerd.

Voorbeeld:

```
<element name="BurgerservicenummerBsnClient" type="iwlz:BurgerServiceNummer">
  <xs:annotation>
     <xs:documentation>Een door de overheid toegekend identificerend nummer in
      het kader van het vereenvoudigen van het contact tussen overheid en
      burgers en het verminderen van de administratieve lasten. Voor meer
      informatie over het BSN kunt u terecht bij het Informatiepunt BSN in de
      zorg (www.infobsnzorg.nl).
     </xs:documentation>
  <xs:annotation>
</element>
```

3.5 Default en vaste waarden

Het is een best practice om geen gebruik te maken van zogenaamde default en fixed values omdat er niet gegarandeerd kan worden dat de ontvangende partij het XML-bestand ook daadwerkelijk valideert tegen de XML-schemadefinitie. Wanneer de ontvangende partij dit niet doet zal de default of vaste waarde ook niet verkregen worden. Andersom vult validatie mogelijk onbedoeld een waarde in wanneer het element opzettelijk is leeg gelaten met een default waarde. Er wordt daarom binnen de XML-schemadefinities geen gebruik gemaakt van default en vaste waarden.

3.6 Substitution Groups en Choice

Omdat het Venetian Blind design pattern als best-practice voor de XML-schemadefinities

geldt, is er geen mogelijkheid om "substitution groups" te gebruiken. Het gebruik van "substitution groups" vereist namelijk dat alle elementen (kandidaten voor vervanging) in de globale namespace worden gedefinieerd, hetgeen binnen het Venetian Blind pattern niet is toegestaan. In plaats daarvan moet de "Choice" optie worden gebruikt om toegestane waarden te definiëren.

3.7 any en anyAttribute

Het "any" element en "anyAttribute" attribuut zijn in de standard gedefinieerd om een bepaalde mate van vrijheid te ondersteunen. In plaats van deze vrijheid op deze manier te ondersteunen, is het binnen de XML schemadefinities de richtlijn om nieuwe versies van XML-schemadefinities op te stellen en oude interfaces (waar nodig) voor een bepaalde tijd te ondersteunen.

n.b. momenteel staat dit ter discussie voor de toekomst, om backwards compatibiliteit te ondersteunen.

3.8 minOccurs en maxOccurs

De default waarde voor de minOccurs en maxOccurs attributen is 1. Voor de overzichtelijkheid is het binnen de XML-schemadefinities de richtlijn om de schemadefinities niet onnodig te vervuilen met deze attributen indien de default waarde wordt gebruikt. Concreet betekend dit dat alleen wanneer een element optioneel is minOccurs = "0" is meegegeven en wanneer het is toegestaan dat en element meerdere keren aangeleverd mag worden maxOccurs = "unbounded" is gedefinieerd.

3.9 Repeterende elementen

Wanneer het is toegestaan om meerdere voorkomens van hetzelfde element op te nemen in het xml-bericht, zullen deze elementen binnen een gelijknamige in meervoudsvorm gespecificeerde collectie worden ondergebracht.

Voorbeeld RetourCode:

```
<xs:complexType name="Client">
  <xs:sequence>
    <xs:element name="Bsn" type="iwlz:BurgerServicenummer" />
    <xs:element name="CizCode" type="iwlz:CizCode" minOccurs="0" />
    <xs:element name="Clientnummer" type="iwlz:Persoonsid" />
    <xs:element name=”Retour” type=”aw34:Retour” />
  </xs:sequence>
</xs:complexType>
<xs:complexType name=”Retour”>
  <xs:sequence>
   <xs:element name=”RetourCodes” type=”aw34:RetourCodes”/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name=”Retourcodes”>
  <xs:sequence>
   <xs:element name=”RetourCode” type=”iwlz:LDT_RetourCode” maxOccurs=”unbounded”/>
  </xs:sequence>
</xs:complexType>
```

4 Afspraken bij het opstellen van het xml-bericht

Hoewel dit document de richtlijnen beschrijft op basis waarvan de xml-schemadefinities zijn opgesteld, volgen in dit hoofdstuk nog enkele aanbevelingen voor het opstellen van de bijbehorende xml-berichten.

4.1 Gebruik Byte-Order-Mark (BOM)

Het toevoegen van een Byte-Order-Mark aan een XML-bericht is niet toegestaan.

Een Byte-Order-Mark is bedoeld om bij het gebruik van UTF-16 encoding aan te kunnen geven wat de gehanteerde byte volgorde is om tot de juiste encoding van de character-set te komen. Deze BOM staat altijd vooraan in de tekenreeks.

Omdat de afgesproken encoding van de xml-berichten 'UTF-8' is, is het niet nodig om een BOM mee te geven. Daarnaast is er gebleken dat het gebruik van een BOM binnen een XML-bericht tot problemen kan leiden bij de verwerking van dat XML document.

Met name Microsoft producten geven standaard wel een BOM mee bij gebruik van 'UTF-8' encoding.

De afspraak is in het informatiemodel op iStandaarden vastgelegd in bedrijfsregel OP192.

4.2 Formatting

Een XML-bericht is 'well-formed' indien deze voldoet aan de syntax-regels. Deze regels beschrijven niet hoe een XML-bericht geformatteerd moet zijn. Wanneer een XML-bericht well-formed is, is het toegestaan om de inhoud van het bericht op één regel te zetten.
Voor de leesbaarheid heeft het de voorkeur dit niet te doen en het bericht over meerdere regels te spreiden en gebruik te maken van inspringen. De consequentie is echter dat het bericht groter wordt door het toevoegen van non-essential characters.

In de praktijk is gebleken dat XML-berichten van grote omvang, waarbij de data op 1 enkele regel staat, bij sommige partijen leidt tot problemen in de verwerking. Wanneer dit probleem zich voordoet, is het aan de aanleverende partij om de bestanden te formatteren volgens de gebruikelijke principes van XML-formatering ('pretty-print') met gebruik van het einde-regel teken "CR/LF" (Windows einde-regel teken).

4.3 Bericht validatie

Berichten worden gevalideerd tegen de bijbehorende XML-schemdadefinitie (xsd) voordat deze worden aangeboden en wanneer deze worden ontvangen.