Vertaalregels voor het generen van StUF-schema’s vanuit een RGB-informatiemodel

# Inleiding

[to do: omzetten bindings naar UGM ipv SIM]

# Notatie

In dit document worden rechte haken […] als placeholder gebruikt om het XSD-template te binden aan het informatiemodel.

|  |  |
| --- | --- |
| M | Waarde van metagegeven M (evaluatie van de expressie M) |
| [ … ] | Verplichte binding naar het informatiemodel |
| [ … ]? | Optionele binding naar het informatiemodel |
| … | … | Of-constructie |
| …\* | Nul of meer keer het voorgaande op een nieuwe regel |
| ( … ) | Haakjes om voorrang te bepalen |

# Objecttype

Een objecttype uit het informatiemodel wordt vertaald naar twee complexType’s. Het eerste complexType (M-basis) vormt de basis en bevat alle attribuut-, groepsattribuut- en relatiesoorten van het corresponderende objecttype. Het tweede complexType (M-kerngegevens) bevat alleen de kerngegevens om objecten uniek aan te duiden.

## Basistype

Een objecttype wordt als volgt vertaald naar een StUF-basistype in het XSD-schema.

<complexType name="[Mnemonic objecttype]-basis">

<annotation>

<documentation>[Naam objecttype]: [Definitie objecttype]</documentation>

</annotation>

<sequence>

[(Attribuutsoort | Groepsattribuutsoort)\*]

[<element ref="StUF:tijdvakObject" minOccurs="0"/>]?

<element ref="StUF:tijdvakGeldigheid" minOccurs="0"/>

<element ref="StUF:tijdstipRegistratie" minOccurs="0"/>

<element ref="StUF:extraElementen" minOccurs="0"/>

<element ref="aanvullendeElementen" minOccurs="0"/>

<element name ="historieMaterieel"

type="[ns prefix]:[Mnemonic objecttype]-basis"   
 minOccurs="0"

maxOccurs="unbounded"/>

<element name="historieFormeel"

type="[ns prefix]:[Mnemonic objecttype]-basis"   
 minOccurs="0" maxOccurs=”unbounded”/>   
 [Relatiesoort\*]

</sequence>

<attribute name="entiteittype"

type="[ns prefix]:Entiteittype[Mnemonic objecttype]"/>

<attributeGroup ref="StUF:entiteit"/>

</complexType>

<simpleType name="Entiteittype[Mnemonic objecttype]">

<restriction base="string">

<enumeration value="[Mnemonic objecttype]"/>

</restriction>

</simpleType>

Alleen als er op minimaal één van de attribuutsoorten of groepsattribuutsoorten materiele historie is gedefinieerd door middel van het attribute “meta:materieleHistorie” (zie sectie Attribuutsoort), dan moet in het bovenstaande schema het element “historieMaterieel” aanwezig zijn. Hetzelfde verhaal geldt voor het element “historieFormeel”. Hieronder een voorbeeld van de vertaling van het objectttype ‘Besluittype’ (BST).

### Voorbeeld

<complexType name="BST-basis">

<annotation>

<documentation>

Besluittype: Generieke aanduiding van de aard van een besluit.

</documentation>

</annotation>

<sequence>

[(Attribuutsoort | Groepsattribuutsoort)\*]

<element ref="StUF:tijdvakGeldigheid" minOccurs="0"/>

<element ref="StUF:tijdstipRegistratie" minOccurs="0"/>

<element ref="StUF:extraElementen" minOccurs="0"/>

<element ref="aanvullendeElementen" minOccurs="0"/>

<element name="historieMaterieel"

type="ztc:BST-basis" minOccurs="0"

maxOccurs="unbounded"/>

<element name="historieFormeel"

type="ztc:BST-basis" minOccurs="0"

maxOccurs="unbounded"/>

[Relatiesoort\*]

</sequence>

<attribute name="ztc:entiteittype" type="EntiteittypeBST"/>

<attributeGroup ref="StUF:entiteit"/>

</complexType>

<simpleType name="EntiteittypeBST">

<restriction base="string">

<enumeration value="BST"/>

</restriction>

</simpleType>

## Kerngegevens

De kerngegevens van een objecttype worden als volgt vertaald naar StUF.

<complexType name="[Mnemonic objectttype]-kerngegevens">

<complexContent>

<restriction base="[ns prefix]:[Mnemonic objectttype]-basis">

<sequence>

[(Attribuutsoort | Groepsattribuutsoorten | Relatiesoort)\*]

</sequence>

<attribute name="entiteittype"   
 type="BG:Entiteittype[Mnemonic objecttype]"   
 use="required"/>

<attribute ref="StUF:sleutelVerzendend"/>

<attribute ref="StUF:sleutelOntvangend"/>

<attribute ref="StUF:sleutelGegevensbeheer"/>

<attribute ref="StUF:noValue" use="prohibited"/>

<attribute ref="StUF:scope" use="prohibited"/>

<attribute ref="StUF:verwerkingssoort"/>

</restriction>

</complexContent>

</complexType>

De attribuutsoorten, groepssoorten en relatiesoorten in de restriction van het bovenstaande complex type worden gespecificeerd door het metagegeven Unieke aanduiding objecttype. Met deze elementen kan een object uniek worden aangeduid. Deze gegevens worden ook wel de kerngegevens genoemd.

Hieronder een voorbeeld hoe de unieke aanduiding van het objecttype Besluittype wordt vertaald naar StUF.

### Voorbeeld

<complexType name="BST-kerngegevens">

<complexContent>

<restriction base="[ns prefix]:BST-basis">

<sequence>

<element name="omschrijving" type="ztc:Omschrijving-e"   
 nillable="true" minOccurs="0"   
 metadata:materieleHistorie="false"   
 metadata:formeleHistorie="false"/>

<element name="omschrijvingGeneriek" type="ztc:Omschrijving-e"   
 nillable="true" minOccurs="0"   
 metadata:materieleHistorie="true"   
 metadata:formeleHistorie="false"/>

<element name="maaktDeelUitVan" type="ztc:BSTCAT-kerngegevens"   
 nillable="true" minOccurs="0"   
 metadata:materieleHistorie="false"   
 metadata:formeleHistorie="false"/>

</sequence>

<attribute name="entiteittype"   
 type="BG:EntiteittypeBST"   
 use="required"/>

<attribute ref="StUF:sleutelVerzendend"/>

<attribute ref="StUF:sleutelOntvangend"/>

<attribute ref="StUF:sleutelGegevensbeheer"/>

<attribute ref="StUF:noValue" use="prohibited"/>

<attribute ref="StUF:scope" use="prohibited"/>

<attribute ref="StUF:verwerkingssoort"/>

</restriction>

</complexContent>

</complexType>

# Attribuutsoort

Bij de vertaling van een attribuutsoort kunnen zowel een element, een complex type als een simple type worden gecreëerd.

## Element

Als het metagegeven ‘Formaat attribuutsoort’ gelijk is aan één van de waarden in onderstaande tabel dan wordt er alleen een element gecreëerd omdat er gebruik kan worden gemaakt van een bestaande complex types uit de StUF-onderlaag (zie ‘stuf0302.xsd’).

<element name="[Xml-tag attribuutsoort]"   
 type="[Formaat attribuutsoort]"   
 minOccurs="0"   
 maxOccurs="[***upperBound***(Indicatie kardinaliteit)]"  
 metadata:materieleHistorie="***Boolean***([Indicatie materiele historie])"   
 metadata:formeleHistorie="***Boolean***([Indicatie formele historie])"/>

|  |  |
| --- | --- |
| **Formaat attribuutsoort** | **Type in XSD-schema** |
| DATUM | stuf:MogelijkOnvolledigeDatum-e |
| DATUM? | stuf:MogelijkOnvolledigeDatum-e |
| JAAR | stuf:Jaar-e |
| JAARMAAND | stuf:JaarMaand-e |
| DT | stuf:MogelijkOnvolledigeDatumTijd-e |
| DT? | stuf:MogelijkOnvolledigeDatumTijd-e |

In de andere gevallen gaat de vertaling als volgt:

<element name="[Element.naam]"   
 type="[ns prefix]:***Capitalize***([Element.naam])-e"   
 minOccurs="0"   
 maxOccurs="[***UpperBound***(Indicatie kardinaliteit)]"  
 metadata:materieleHistorie="[***Boolean***(Indicatie materiele historie)]"   
 metadata:formeleHistorie="[***Boolean***(Indicatie formele historie)]"/>

In de twee bovenstaande XSD-templates worden de volgende functies gebruikt:

* *UpperBound* geeft de waarde van het veld ‘Upper bound’ onderdeel van het samengestelde veld Multiplicity in EA. In geval de waarde “N” is in het veld ‘Upperbound’ dan zet de functie dit om in “unbounded”.
* *Boolean*zet de waarde “Ja” en “Nee” om in respectievelijk “true” en “false”.

## Complex type

<complexType name="[***Capitalize***(Element.naam)]-e">

<choice>

<element name="w" type="[ns prefix]:[***Capitalize***(Element.naam)]"/>

<element name="l" type="stuf:NoValue"/>

</choice>

</complexType>

<complexType name="[***Capitalize***(Element.naam)]-s">

<choice>

<element name="w" type="[ns prefix]:[***Capitalize***(Element.naam)]-w"/>

</choice>

</complexType>

<complexType name="[***Capitalize***(Element.naam)]-w">

<simpleContent>

<extension base="[ns prefix]:[***Capitalize***(Element.naam)]">

<attribute ref="stuf:wildcard"/>

</extension>

</simpleContent>

</complexType>

De functie *Capitalize* zet de eerste letter van de string om in een hoofdletter.

## Simple type

<simpleType name="[***Capitalize***(Element.naam)]">

[Element.formaat]

</simpleType>

Onderstaande tabel specificeert hoe het formaat van het element in het bovenstaande simpleType vertaald wordt.

|  |  |
| --- | --- |
| **Formaat** | **XSD-schema** |
| AN | <restriction base="string">  …  </restriction> |
| AN[n] | <restriction base="string">  <maxLength value="[n]"/>  …  </restriction> |
| N | <restriction base="integer">  …  </restriction> |
| N[n] | <restriction base="integer">  <maxLength value="[n]"/>  …  </restriction> |
| N[n],[d] | <restriction base="integer">  <totalDigits value="[n+d]"/>  <fractionDigits value="[d]"/>  …  </restriction> |
| INDIC | <restriction base="boolean"/> |
| TXT | <restriction base="string">  …  </restriction> |
| URI | <restriction base="anyURI">  …  </restriction> |
| Enumeratiesoort | <restriction base="string">  <enumeration value="…"/>  <enumeration value="…"/>  …  <enumeration value="…"/>  </restriction> |
| Tabel-entiteit | Ga naar de definitie van de correponderende Tabel-entiteit en haal het formaat van het gelijknamige attribuutsoort op. Vertaal dit formaat naar schema volgens de eerdere regels in deze tabel. |
| POSTCODE | Zie GAB |

Onderstaande tabel specificeert hoe de facetten bij het formaat van het element in het bovenstaande simpleType vertaald worden.

|  |  |
| --- | --- |
| **Facet** | **XSD-schema** |
| Patroon | <pattern value="…"/> |
| Minimum lengte | <minLength value="…"/> |
| Minimum waarde (inclusief) | <minInclusive value="…"/> |
| Maximum waarde (inclusief) | <maxInclusive value="…"/> |

## Voorbeeld

Hieronder een voorbeeld van de vertaling van het element ‘omschrijvingGeneriek’ met formaat AN80 van het entiteittype ‘Besluittype’.

<element name="omschrijvingGeneriek"   
 type="ztc:OmschrijvingGeneriek-e"   
 minOccurs="0"  
 maxOccurs="1"  
 metadata:materieleHistorie="true"   
 metadata:formeleHistorie="false"/>

<complexType name="OmschrijvingGeneriek-e">

<choice>

<element name="w" type="ztc:OmschrijvingGeneriek"/>

<element name="l" type="stuf:NoValue"/>

</choice>

</complexType>

<complexType name=" OmschrijvingGeneriek-s">

<choice>

<element name="w" type="ztc:OmschrijvingGeneriek-w"/>

</choice>

</complexType>

<complexType name=" OmschrijvingGeneriek-w">

<simpleContent>

<extension base="ztc:OmschrijvingGeneriek">

<attribute ref="stuf:wildcard"/>

</extension>

</simpleContent>

</complexType>

<simpleType name="OmschrijvingGeneriek ">

<restriction base="string">

<maxLength value="80"/>

</restriction>

</simpleType>

# Groepsattribuutsoort

## Element

<element name="[Xml-tag groepsattribuutsoort]"   
 type="[ns prefix]:[***Capitalize***(Xml-tag groepsattribuutsoort)Grp]"   
 minOccurs="0"   
 maxOccurs="[***UpperBound***(Indicatie kardinaliteit)]"  
 metadata:materieleHistorie="[***Boolean***(Indicatie materiele historie)]"   
 metadata:formeleHistorie="[***Boolean***(Indicatie formele historie)]"/>

## Complex type

<complexType name="[***Capitalize***(XML-tag groepsattribuutsoort)]Grp">

<annotation>

<documentation>

[Definitie groepsattribuutsoort]

</documentation>

</annotation>

<sequence>

[( Attribuutsoort | Groepsattribuutsoort | Relatiesoort )\*]

</sequence>

</complexType>

In het bovenstaande schema zet de functie *Capitalize(…)* de eerste letter van de string om in een hoofdletter. In StUF beginnen de namen van een complex type altijd met een hoofdletter. Hieronder een voorbeeld van de vertaling van de groep Checklistitem naar StUF.

## Voorbeeld

<element name="checklistitem"   
 type="ztc:ChecklistitemGrp"

minOccurs="0"

maxOccurs="unbounded"

metadata:formeleHistorie="false"

metadata:materieleHistorie="false"/>

<complexType name="ChecklistitemGrp">

<annotation>

<documentation>

Te controleren aandachtspunt voorafgaand aan het bereiken   
 van een status van het STATUSTYPE

</documentation>

</annotation>

<sequence>

<element name="naam" type="ztc:Naam-e" nillable="true" minOccurs="0"   
 metadata:materieleHistorie="true"   
 metadata:formeleHistorie="false"/>

<element name="vraag" type="ztc:Vraag-e" nillable="true" minOccurs="0"   
 metadata:materieleHistorie="true"  
 metadata:formeleHistorie="false"/>

<element name="verplicht" type="bg:Indicatie-e" nillable="true"   
 minOccurs="0" metadata:materieleHistorie="true"   
 metadata:formeleHistorie="false"/>

<element name="toelichting" type="ztc:Toelichting-e" nillable="true"

minOccurs="0" metadata:materieleHistorie="true"

metadata:formeleHistorie="false"/>

</sequence>

</complexType>

# Relatiesoort en relatieklasse

De vertaling van een relatiesoort met zijn eventuele relatieklasse bestaat uit twee delen: de heen- en de terugrelatie.

## Heenrelatie

De heenrelatie vertalen we naar een element en twee complex types in het schema.

### Element

Een relatiesoort wordt als volgt vertaald naar een elementdefinitie.

<element name="[Xml-tag naam relatiesoort]"

type="[ns qualifier]:[Mnemonic relatiesoort]-basis"

minOccurs="0"

maxOccurs="[***maxVoorkomens***(Target role Multiplicity)]"

metadata:materieleHistorie="[***Boolean***(Indicatie materiele historie)]"

metadata:formeleHistorie="[***Boolean***(Indicatie formele historie)]"/>

**Let op**: in de definitie van het bovenstaande element is de waarde van minOccurs altijd gelijk aan “0” ongeacht de kardinaliteit die gespecificeerd is in het informatiemodel. In de basisschema’s van StUF zijn alle elementen optioneel. Op die manier wordt hergebruik bevordert. Bijvoorbeeld in vraagberichten wil je eigenschappen van een object die niet relevant zijn voor het stellen van de vraag kunnen weg laten. Of wil je bijvoorbeeld in een mutatiebericht alleen de gegevens opnemen die gewijzigd zijn.

De functie *maxVoorkomens* geeft het maximale aantal voorkomens terug van de kardinaliteit van de doelrol in de relatie (Target role Multiplicity). Op een analoge wijze geeft de functie *minVoorkomens* het minimale aantal voorkomens terug. De twee functie zijn gedefinieerd in onderstaande tabel.

|  |  |  |  |
| --- | --- | --- | --- |
| **Kardinaliteit (Multiplicity)** | **minVoorkomens** | **maxVoorkomens (zonder historie)** | **maxVoorkomens  (in geval van historie)** |
| \* | 1 | unbounded | unbounded |
| 0 | 0 | 1 | unbounded |
| 0..\* | 0 | unbounded | unbounded |
| 0..1 | 0 | 1 | unbounded |
| 1 | 1 | 1 | unbounded |
| 1..\* | 1 | unbounded | unbounded |

In geval er materiële historie is gedefinieerd op de relatie (metadata:materieleHistorie=”true”) dan geeft de functie *maxVoorkomens* altijd de waarde “unbounded” terug. Dan kan het hierboven gedefinieerde element voor de relatiesoort ook worden gebruikt voor het opnemen van (meerdere) historische voorkomens.

In het informatiemodel is de naam van een relatiesoort niet altijd uniek. In de vertaling moeten (de XML-tags van) deze namen gedisambigueerd worden door de naam van het gerelateerde objecttype in camelCase eraan te plakken. Dit is bijvoorbeeld het geval bij de volgende relaties in het complex type STT-basis.

<element name="heeftVerplichte**Informatieobjecttype**" type="ztc:STTIOT-basis"   
 nillable="true" minOccurs="0" maxOccurs="unbounded"/>

<element name="heeftVerplichte**Eigenschap**" type="ztc:STTEIG-basis"   
 nillable="true" minOccurs="0" maxOccurs="unbounded"/>

<element name="heeftVerplichte**ZaakObjecttype**" type="ztc:STTZOT-basis"   
 nillable="true" minOccurs="0" maxOccurs="unbounded"/>

Een andere mogelijkheid is om te eisen dat de XML-tags van de relatienamen altijd op voorhand uniek zijn in het informatiemodel.

### Basistype

<complexType name="[Mnemonic relatiesoort]-basis">

<annotation>

<documentation>[Naam relatiesoort]: [Definitie relatiesoort]</documentation>  
 </annotation>

<choice>

<element ref="stuf:leeg"/>

<sequence>

<element name="gerelateerde"

type="[ns prefix]:[Mnemonic target objectttype]-basis"

nillable="true"   
 minOccurs="0"/>

[( Attribuutsoort | Groepsattribuutsoorten )\*]?

<element ref="StUF:tijdvakRelatie" minOccurs="0"/>

[<element ref="StUF:tijdvakGeldigheid" minOccurs="0"/>]?

<element ref="StUF:tijdstipRegistratie" minOccurs="0"/>

<element ref="StUF:extraElementen" minOccurs="0"/>

<element ref="aanvullendeElementen" minOccurs="0"/>

[<element name="historieMaterieel"

type="[ns prefix]:[Mnemonic relatiesoort]-basis" minOccurs="0"

maxOccurs="unbounded"/>]?

[<element name="historieFormeel"

type="[ns prefix]:[Mnemonic relatiesoort]-basis" minOccurs="0"   
 maxOccurs="unbounded"/>]?

[<element name="historieFormeelRelatie"

type="[ns prefix]:[Mnemonic relatiesoort]-basis" minOccurs="0"  
 maxOccurs="unbounded"/>]?

[Relatiesoort\*]?

</sequence>

<choice>

<attribute name="entiteittype"

type="[ns prefix]:Entiteittype[Mnemonic relatiesoort]"/>

<attributeGroup ref="StUF:entiteit"/>

</complexType>

<simpleType name="Entiteittype[Mnemonic relatiesoort]">

<restriction base="string">

<enumeration value="[Mnemonic relatiesoort]"/>

</restriction>

</simpleType>

Een complex type voor een relatiesoort kan ook vertalingen van (groeps)attribuut- en relatiesoorten bevatten als de relatiesoort een relatieklasse is. Sommige elementen zijn niet altijd aanwezig zoals wordt aangegeven door de […]? notatie. Deze elementen worden alleen opgenomen als er aan de volgende condities wordt voldaan:

|  |  |
| --- | --- |
| **Element** | **Voorwaarde voor opname in complex type voor relatiesoort** |
| StUF:tijdvakGeldigheid | De relatiesoort bevat (groeps)attribuutsoorten. |
| historieMaterieel | Voor minimaal één van de (groeps)attribuutsoorten geldt dat metadata:materieleHistorie="true”. |
| historieFormeel | Voor minimaal één van de (groeps)attribuutsoorten geldt dat metadata:formeleHistorie="true”. |
| historieFormeelRelatie | metadata:formeleHistorie="true” in het element voor de relatiesoort (zie sectie Element). |

### Kerngegevens

<complexType name="[Mnemonic relatiesoort]-kerngegevens">

<annotation>

<documentation>Kerngegevens van de relatie</documentation>

</annotation>

<complexContent>

<restriction base="[ns prefix]:[Mnemonic relatiesoort]-basis">

<sequence>

<element name="gerelateerde"

type="[ns prefix]:[Mnemonic target objectttype]-kerngegevens"   
 nillable="true" minOccurs="0"/>

</sequence>

<attribute name="entiteittype"   
 type="[ns prefix]:Entiteittype[Mnemonic relatiesoort]"   
 use="required"/>

<attribute ref="stuf:sleutelVerzendend"/>

<attribute ref="stuf:sleutelOntvangend"/>

<attribute ref="stuf:sleutelGegevensbeheer"/>

<attribute ref="stuf:noValue" use="prohibited"/>

<attribute ref="stuf:scope" use="prohibited"/>

<attribute ref="stuf:verwerkingssoort"/>

</restriction>

</complexContent>

</complexType>

### Voorbeeld

Hieronder een voorbeeld van de vertaling van de relatiesoort “maakt deel uit van” die gebruikt wordt om relaties te leggen tussen objecten van het type Besluit (BSL) en Catalogus (CAT).

<element name="maaktDeelUitVan"

type="ztc:BSTCAT-basis"

minOccurs="0"

maxOccurs="1"

metadata:materieleHistorie="false"

metadata:formeleHistorie="false"/>

<complexType name="BSTCAT-basis">

<annotation>

<documentation>Relatie naar de catalogus waarvan het besluittype deel uit   
 maakt.  
 </documentation>  
 </annotation>

<choice>

<element ref="stuf:leeg"/>

<sequence>

<element name="gerelateerde" type="ztc:CAT-basis" nillable="true"   
 minOccurs="0"/>

<element ref="stuf:extraElementen" minOccurs="0"/>

<element ref="stuf:aanvullendeElementen" minOccurs="0"/>

</sequence>

</choice>

<attribute name="ztc:EntiteittypeBSTCAT"/>

<attributeGroup ref="stuf:entiteit"/>

</complexType>

<simpleType name="EntiteittypeBSTCAT">

<restriction base="string">

<enumeration value="BSTCAT"/>

</restriction>

</simpleType>

<complexType name="BSTCAT-kerngegevens">

<annotation>

<documentation>Kerngegevens van de relatie</documentation>

</annotation>

<complexContent>

<restriction base="ztc:BSTCAT-basis">

<sequence>

<element name="gerelateerde" type="ztc:CAT-kerngegevens"   
 nillable="true" minOccurs="0"/>

</sequence>

<attribute name="ztc:entiteittypeBSTCAT" use="required"/>

<attribute ref="stuf:sleutelVerzendend"/>

<attribute ref="stuf:sleutelOntvangend"/>

<attribute ref="stuf:sleutelGegevensbeheer"/>

<attribute ref="stuf:noValue" use="prohibited"/>

<attribute ref="stuf:scope" use="prohibited"/>

<attribute ref="stuf:verwerkingssoort"/>

</restriction>

</complexContent>

</complexType>

Omdat er voor deze relatie geen historie is gedefinieerd zijn de elementen “historieMaterieel”, “historieFormeel” en “historieFormeelRelatie” niet opgenomen in het complex type “BSTCAT-basis”.

## Terugrelatie

Op analoge wijze als de heenrelatie kunnen we de terugrelatie naar schema vertalen.

### Element

<element name="[Xml-tag naam relatiesoort terugrelatie]"

type="[ns qualifier]:[Mnemonic relatiesoort]-basis"

minOccurs="0"

maxOccurs="[***maxVoorkomens***(source role Multiplicity)]"

metadata:materieleHistorie="[Indicatie materiele historie]"

metadata:formeleHistorie="[Indicatie formele historie]"/>

### Basistype

<complexType name="[Mnemonic relatiesoort terugrelatie]-basis">

<annotation>

<documentation>  
 [Naam relatiesoort terugrelatie]: [Definitie relatiesoort]  
 </documentation>  
 </annotation>

<choice>

<element ref="stuf:leeg"/>

<sequence>

<element name="gerelateerde"

type="[ns prefix]:[Mnemonic source objectttype]-basis"

nillable="true"   
 minOccurs="0"/>

[( Attribuutsoort | Groepsattribuutsoorten )\*]?

<element ref="StUF:tijdvakRelatie" minOccurs="0"/>

[ <element ref="StUF:tijdvakGeldigheid" minOccurs="0"/> ]?

<element ref="StUF:tijdstipRegistratie" minOccurs="0"/>

<element ref="StUF:extraElementen" minOccurs="0"/>

<element ref="aanvullendeElementen" minOccurs="0"/>

[ <element name="historieMaterieel"

type="[ns prefix]:[Mnemonic relatiesoort terugrelatie]-basis"   
 minOccurs="0"

maxOccurs="unbounded"/> ]?

[ <element name="historieFormeel"

type="[ns prefix]:[Mnemonic relatiesoort terugrelatie]-basis"   
 minOccurs="0"   
 maxOccurs="unbounded"/> ]?

[ <element name="historieFormeelRelatie"

type="[ns prefix]:[Mnemonic relatiesoort terugrelatie]-basis"   
 minOccurs="0"  
 maxOccurs="unbounded"/> ]?

[Relatiesoort\*]?

</sequence>

</choice>

<attribute name="entiteittype"

type="[ns prefix]:Entiteittype[Mnemonic relatiesoort terugrelatie]"/>

<attributeGroup ref="StUF:entiteit"/>

</complexType>

<simpleType name="Entiteittype[Mnemonic relatiesoort terugrelatie]">

<restriction base="string">

<enumeration value="[ Mnemonic relatiesoort terugrelatie]"/>

</restriction>

</simpleType>

### Kerngegevens

<complexType name="[Mnemonic relatiesoort terugrelatie]-kerngegevens">

<annotation>

<documentation>Kerngegevens van de relatie</documentation>

</annotation>

<complexContent>

<restriction base="[ns prefix]:[Mnemonic relatiesoort terugrelatie]-basis">

<sequence>

<element name="gerelateerde"

type="[ns prefix]:[Mnemonic source objectttype]-kerngegevens"   
 nillable="true" minOccurs="0"/>

</sequence>

<attribute name="entiteittype"   
 type="BG:Entiteittype[Mnemonic relatiesoort terugrelatie]"   
 use="required"/>

<attribute ref="stuf:sleutelVerzendend"/>

<attribute ref="stuf:sleutelOntvangend"/>

<attribute ref="stuf:sleutelGegevensbeheer"/>

<attribute ref="stuf:noValue" use="prohibited"/>

<attribute ref="stuf:scope" use="prohibited"/>

<attribute ref="stuf:verwerkingssoort"/>

</restriction>

</complexContent>

</complexType>

### Voorbeeld

Hieronder een voorbeeld van de vertaling van de terugrelatie van BSTCAT. Deze terugrelatie (CATBST) wordt als volgt omgezet naar StUF.

<element name="bestaatUitBesluittype"

type="ztc:CATBST-basis"

minOccurs="0"

maxOccurs="unbounded"

metadata:materieleHistorie="false"

metadata:formeleHistorie="false"/>

<complexType name="CATBST-basis">

<annotation>

<documentation>Relatie naar het besluittype dat tot de catalogus   
 behoort.

</documentation>

</annotation>

<choice>

<element ref="stuf:leeg"/>

<sequence>

<element name="gerelateerde" type="ztc:BST-basis" nillable="true"   
 minOccurs="0"/>

<element ref="stuf:extraElementen" minOccurs="0"/>

</sequence>

</choice>

<attribute name="ztc:EntiteittypeCATBST"/>

<attributeGroup ref="stuf:entiteit"/>

</complexType>

<simpleType name="EntiteittypeCATBST">

<restriction base="string">

<enumeration value="CATBST"/>

</restriction>

</simpleType>

<complexType name="CATBST-kerngegevens">

<annotation>

<documentation>Kerngegevens van de relatie</documentation>

</annotation>

<complexContent>

<restriction base="ztc:CATBST-basis">

<sequence>

<element name="gerelateerde" type="ztc:BST-kerngegevens"   
 nillable="true" minOccurs="0"/>

</sequence>

<attribute name="ztc:entiteittypeCATBST" use="required"/>

<attribute ref="stuf:sleutelVerzendend"/>

<attribute ref="stuf:sleutelOntvangend"/>

<attribute ref="stuf:sleutelGegevensbeheer"/>

<attribute ref="stuf:noValue" use="prohibited"/>

<attribute ref="stuf:scope" use="prohibited"/>

<attribute ref="stuf:verwerkingssoort"/>

</restriction>

</complexContent>

</complexType>

# Referentielijst

Een referentielijst wordt vertaald naar een StUF-tabelentiteit:

<complexType name="[Mnemonic referentielijst]-tabel">

<annotation>

<documentation>

[Naam referentielijst]: [Definitie referentielijst]

</documentation>

</annotation>

<sequence>

[Referentiegegeven\*]

[<element ref="StUF:tijdvakObject" minOccurs="0"/>]?

<element ref="StUF:tijdvakGeldigheid" minOccurs="0"/>

<element ref="StUF:tijdstipRegistratie" minOccurs="0"/>

<element ref="StUF:extraElementen" minOccurs="0"/>

<element ref="StUF:aanvullendeElementen" minOccurs="0"/>

</sequence>

<attribute name="[ns prefix]:Entiteittype[Mnemonic referentielijst]"/>

<attribute ref="StUF:scope"/>

<attribute ref="StUF:verwerkingssoort"/>

</complexType>

# Supertype

Als een objecttype een generalisatie (supertype) is met subtypes dan wordt het anders vertaald dan een gewoon objecttype.

## Basistype

Het objectttype wordt als volgt vertaald naar een StUF-basistype in het XSD-schema.

<complexType name="[Mnemonic objecttype]-basis">

<annotation>

<documentation>[Naam objecttype]: [Definitie objecttype]</documentation>

</annotation>

<sequence>

[(Attribuutsoort | Groepsattribuutsoort)\*]

<element ref="StUF:extraElementen" minOccurs="0"/>  
 [Relatiesoort\*]

<choice minOccurs="0" maxOccurs="[aantal subtypes]">

<element name="[Xml-tag subtype1]"   
 type="[ns prefix]:[Mnemonic subtype1]-basis"   
 nillable="true" minOccurs="0"/>

<element name="[Xml-tag subtype2]"   
 type="[ns prefix]:[Mnemonic subtype2]-basis"   
 nillable="true" minOccurs="0"/>

…

</choice>

</sequence>

<attribute ref="StUF:entiteittype" fixed="[Mnemonic objecttype]"/>

<attributeGroup ref="StUF:entiteit"/>

</complexType>

## Kerngegevens

De kerngegevens van een objecttype worden nu als volgt vertaald naar StUF.

<complexType name="[Mnemonic objectttype]-kerngegevens">

<complexContent>

<restriction base="[ns prefix]:[Mnemonic objectttype]-basis">

<choice minOccurs="0">

<element name="[Xml-tag subtype1]"

type="BG:[Mnemonic subtype1]-kerngegevens"   
 nillable="true"/>

<element name="[Xml-tag subtype2]"

type="BG:[Mnemonic subtype2]-kerngegevens"   
 nillable="true"/>

…

</choice>

<attribute ref="StUF:entiteittype" use="prohibited"/>

</restriction>

</complexContent>

</complexType>

# Referentiegegeven

Bij de vertaling van een referentiegegeven worden zowel een element, een complex type als een simple type worden gecreëerd analoog aan de vertaling van een attribuutsoort.

## Element

<element name="[Xml-tag referentiegegeven]"   
 type="[ns prefix]:***Capitalize***([Xml-tag referentiegegeven])-e"   
 nillable="true"   
 minOccurs="0"   
 maxOccurs="[***UpperBound***(Indicatie kardinaliteit)]"/>

## Complex type

<complexType name="[***Capitalize***(Xml-tag referentiegegeven)]-e">

<simpleContent>

<extension base="[ns prefix]:[***Capitalize***([Xml-tag referentiegegeven])]">

<attributeGroup ref="stuf:element"/>

</extension>

</simpleContent>

</complexType>

## Simple type

<simpleType name="[***Capitalize***([Xml-tag referentiegegeven])">

[Formaat referentiegegeven]

</simpleType>