

Hieronder een voorbeeld van het gebruik van de extraElementen-constructie. Het betreft de recentelijke invoering van IBAN- en BIC-nummers voor bankrekeningen via Europese wetgeving. Deze nieuwe gegevens kunnen als volgt als extra elementen worden toegevoegd aan een NPS-entiteit in StUF-BG 3.10:

```
<BG:persoon StUF:entiteittype="NPS" xsi:schemaLocation="http://www.egem.nl/StUF/sector/bg/0310
bg0310.ent.basis.xsd" xmlns:BG="http://www.egem.nl/StUF/sector/bg/0310"
xmlns:StUF="http://www.egem.nl/StUF/StUF0301" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <BG:inp.bsn>123456789</BG:inp.bsn>
  <BG:geslachtsnaam>Korver</BG:geslachtsnaam>
  <BG:voornamen>Henri Peter</BG:voornamen>
  <StUF:extraElementen>
    <StUF:extraElement naam="iban">NL06INGB0006053682</StUF:extraElement>
    <StUF:extraElement naam="bic">INGBNL2A</StUF:extraElement>
  </StUF:extraElementen>
</BG:persoon>
```

3.2.4.2 aanvullendeElementen

De constructie met StUF:extraElementen heeft als nadeel dat de inhoud van de extra elementen niet kan worden gevalideerd tegen een schema. Om dit probleem te ondervangen zijn voor deze constructie in het stuf0301.xsd schema [StUFXSD] het volgende element gedefinieerd:

```
<element name="aanvullendeElementen" type="anyType"/> en het attribute <element
name="aanvullendeElementen">
  <complexType>
    <sequence>
      <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
<attribute name="schemaLocation" type="string"/>
```

gedefinieerd. Aan het gereserveerde element StUF:aanvullendeElementen herkent de berichtverwerkende software waar het de valideerbare extra elementen kan verwachten. Het complexType binnen StUF:aanvullendeElementen specificeert:

- Het element stuf:aanvullendeElementen bevat één of meer elementen (maxOccurs="unbounded" en minOccurs de defaultwaarde 1)
- met een namespace ongelijk aan de stuf0301 namespace (namespace="##other"),
- waarbij het bericht geldig is, als de berichtverwerkende software een element binnen StUF:aanvullendeElementen niet kent
- en waarbij het bericht ongeldig is, als de berichtverwerkende software een element binnen StUF:aanvullendeElementen kent, maar de inhoud ervan niet valide is (processContents="lax"). element is van het type anyType en kan daardoor willekeurige XML-expressies bevatten. Om ervoor te zorgen dat deze expressies gevalideerd kunnen worden door een XSD-schema eisen we dat de inhoud altijd moet voldoen aan de volgende structuur:

Hieronder een voorbeeld van de inhoud van stuf:aanvullendeElementen:

```
<StUF:aanvullendeElementen>
  <ns1:mijnElement1 stuf:schemaLocation="..." xmlns:ns1="...">...</ns1:aanvullendeElementen>
  <ns2:mijnElement2 stuf:schemaLocation="..." xmlns:ns2="...">...</ns2:aanvullendeElementen>
  ...
  <nsN:mijnElementN stuf:schemaLocation="..." xmlns:nsN="...">...</nsN:aanvullendeElementen>
</StUF:aanvullendeElementen>
```

De namespaces "ns1", "ns2", ..., "nsN" en de elementnamen "mijnElement1", "mijnElement2", ..., "mijnElementN" zijn vrij te kiezen. en dezelfde namespace mag meer dan één keer voorkomen. Doorgaans bevat het element <StUF:aanvullendeElementen> maar één subelement met dezelfde naam <ns1:aanvullendeElementen> maar in een eigen namespace. r kunnen toepassingen zijn waarin het waardevol is om de aanvullende elementen in verschillende namespaces op te delen. De hierboven beschreven structuur biedt die mogelijkheid.

chter eEBinnen dat subelement kunnen in principe alle aanvullende elementen worden opgenomen. Het attribute xmlns: voor de namespace-declaratie is verplicht en het attribute stuf:schemaLocation is ook verplicht. Echter het is optioneel of je in het laatst genoemde attribute de namespace en de locatie (gescheiden door een spatie) opneemt of

alleen de namespace omdat de ontvangende applicatie doorgaans de schemalocaties van de betreffende namespaces af kent en op een eigen lokale plek heeft opgeslagen. In het geval dat de schema's op een centrale plek worden beheerd (bijvoorbeeld een openbare website) kan de schemalocatie ook worden meegegeven als een URL. In het bovenstaande XML-fragment valideert het element `<stuf:aanvullendeElementen>` altijd ongeacht de inhoud. De subelementen `<ns1:mijnElement1>`, `<ns2:mijnElement2>`, ..., `<nsN:mijnElementN>` kunnen in een aparte stap worden gevalideerd door in het attribute `schemaLocation` de prefix `stuf` te wijzigen in `xsi` (de namespace van XML). De namespace-prefix `stuf` in `stuf:schemaLocation` zorgt ervoor dat de ontvangende applicatie de extra elementen niet hoeft te valideren als die er niet in geïnteresseerd is.

Hieronder hetzelfde voorbeeld als uit de vorige sectie, maar nu met de nieuwe constructie

```
<bg:persoon xsi:schemaLocation="http://www.egem.nl/StUF/sector/bg/0310/bg0310_ent_basis.xsd"
xmlns:bg="http://www.egem.nl/StUF/sector/bg/0310"
xmlns:stufStUF="http://www.egem.nl/StUF/StUF0301" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <bg:inp.bsn>123456789</bg:inp.bsn>
  <bg:geslachtsnaam>Korver</bg:geslachtsnaam>
  <bg:voornamen>Henri Peter</bg:voornamen>
  <stufStUF:aanvullendeElementen>
    <ae:aanvullendeElementen
      stuf:schemaLocation="http://www.egem.nl/StUF/sector/bg/0310/aanvullendeElementen..."
      xmlns:ae="http://www.egem.nl/StUF/sector/bg/0310/aanvullendeElementen"
      xmlns:gml="http://www.opengis.net/gml">
        <ae:bic>INGBNL2A</ae:bic>
        <ae:iban>NL06INGB0006053682</ae:iban>
        <ae:mijnLocatie>
          <gml:Point srsDimension="2" srsName="urn:ogc:def:crs:EPSG:6.6:4326"
            axisLabels="x y">
              <gml:pos>49.27 -123.11</gml:pos>
            </gml:Point>
          </ae:mijnLocatie>
        </ae:aanvullendeElementen>
      </stufStUF:aanvullendeElementen>
    </bg:persoon>
```

BIC en IBAN zijn nu gedefinieerd als echte XSD-elementen die gevalideerd kunnen worden door een schema. Er is tevens een extra element `<ae:mijnLocatie>` toegevoegd om te laten zien dat de nieuwe constructie ook complexe structuren met attributen en geneste elementen zoals een GML-locatie toestaat.

De standaard eist dat de gebruikelijke StUF-berichtstructuur doorloopt binnen `StUF:aanvullendeElementen`. Voor elementen binnen `stuf:aanvullendeElementen/ae:aanvullendeElementen` gelden dezelfde regels als de andere elementen van de StUF-entiteit waarbinnen het element `<stufStUF:aanvullendeElementen>` voorkomt. Het komt erop neer dat de tags `<stufStUF:aanvullendeElementen>`, `<ae:aanvullendeElementen>`, `</ae:aanvullendeElementen>` en `</stufStUF:aanvullendeElementen>` weggedacht kunnen worden en dat er dan een gewone StUF-entiteit overblijft met alle regels die daarvoor gelden.

Hieronder een voorbeeld van een schema waarmee de extra elementen in bovenstaand voorbeeld gevalideerd worden:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ae="http://www.egem.nl/StUF/sector/bg/0310/aanvullendeElementen"
xmlns:StUF="http://www.egem.nl/StUF/StUF0301" xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.egem.nl/StUF/sector/bg/0310/aanvullendeElementen"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="031003">
  <import namespace="http://www.egem.nl/StUF/StUF0301" schemaLocation="stuf0301.xsd"/>
  <import namespace="http://www.opengis.net/gml" schemaLocation="geometryBasic0dld.xsd"/>
  <element name="aanvullendeElementen" type="ae:AanvullendeElementen"/>
  <complexType name="AanvullendeElementen">
    <sequence>
      <element name="bic" type="ae:Bic-e" nillable="true" minOccurs="0"/>
      <element name="iban" type="ae:Iban-e" nillable="true" minOccurs="0"/>
      <element name="mijnLocatie" type="ae:MijnLocatie"/>
    </sequence>
  </complexType>
  <complexType name="MijnLocatie">
```

```
<sequence>
  <element ref="GML:Point"/>
</sequence>
</complexType>
<complexType name="Bic-e">
  <simpleContent>
    <extension base="ae:Bic">
      <attributeGroup ref="StUF:element"/>
    </extension>
  </simpleContent>
</complexType>
<complexType name="Iban-e">
  <simpleContent>
    <extension base="ae:Iban">
      <attributeGroup ref="StUF:element"/>
    </extension>
  </simpleContent>
</complexType>
<simpleType name="Bic">
  <restriction base="string">
    <maxLength value="11"/>
  </restriction>
</simpleType>
<simpleType name="Iban">
  <restriction base="string">
    <maxLength value="34"/>
  </restriction>
</simpleType>
</schema>
```

3.3 Metagegevens

Een object heeft eigenschappen en de waarden van die eigenschappen worden als gegevens van dat object vastgelegd. Bij een persoon zijn de geslachtsnaam (Broek), het e-mailadres (broek009@onsnet.nu) en de geboortedatum 12-10-1958) voorbeelden van gegevens. Ook gegevens of groepen van gegevens kunnen eigenschappen hebben, bijvoorbeeld de periode waarin een bepaalde waarde geldig is (geweest), het feit dat een bepaald gegeven in onderzoek is (geweest) of het brondocument waaraan de gegevens zijn ontleend. Dit soort gegevens worden ook wel metagegevens genoemd.

Metagegevens zijn net zoals de sleutels onafhankelijk van een concreet entiteitstype als Verblijfsobject of Persoon en kunnen dus los van een concreet sectormodel gedefinieerd worden. De StUF-standaard doet dit en definieert voor een relatief brede verzameling metagegevens de betekenis en de functionaliteit. Op deze wijze is er een uniform mechanisme voor het in berichten opnemen van metagegevens. Bij het definiëren van metagegevens heeft de StUF-standaard als uitgangspunt genomen de metagegevens onderkend in de GBA en in de BAG. De StUF-standaard definieert wel van de GBA en BAG afwijkende mechanismen, omdat in StUF een zo eenvoudig en natuurlijk mogelijk gebruik van metagegevens in berichten voorop staat. Twee ontwerpcriteria liggen aan de gemaakte keuzen ten grondslag:

1. Metagegevens zijn een uitbreiding op een entiteitstype. Het toevoegen van metagegevens aan een entiteitstype mag geen consequenties hebben voor de structuur van dat entiteitstype in een bericht anders dan het toevoegen van elementen en attributes voor de metagegevens.
2. Als er voor een entiteitstype metagegevens zijn gedefinieerd, dan dienen gebruikers van berichten voor een entiteitstype met metagegevens zonder problemen ook berichten zonder metagegevens te kunnen maken c.q. dienen ze bij de verwerking de metagegevens eenvoudig te kunnen negeren. StUF heeft de ambitie om met één set objectdefinities zowel simpele als complexe functionaliteit te kunnen ondersteunen, bijvoorbeeld simpele webservices voor het opvragen van een klein setje actuele gegevens en complexe webservices voor het opvragen van een persoon inclusief zijn materiële en formele historie en inclusief brondocumenten.

De StUF-standaard onderkent de volgende groepen metagegevens:

- Metagegevens met betrekking tot historische waarden
- Metagegevens met betrekking tot brondocumenten waaraan de gegevens ontleend zijn
- Metagegevens met betrekking tot de gebeurtenis die aan (een verandering van) gegevens ten grondslag liggen
- Metagegevens met betrekking tot de status van gegevens.

Deze verschillende groepen metagegevens worden in de volgende paragrafen besproken. In de hierna volgende hoofdstukken wordt de functionaliteit voor het omgaan met deze metagegevens in de verschillende soorten berichten besproken.