Omkeerbare XML-JSON conversie

voor StUF-berichten

*Auteur: Henri Korver*

Inhoudsopgave

[Inleiding 1](#_Toc440619723)

[Mapping tussen XML en JSON 2](#_Toc440619724)

[Patroon 1: Leeg element 3](#_Toc440619725)

[Patroon 2: Leeg element met attribute xsi:nil=”true” 3](#_Toc440619726)

[Patroon 3: Element met een waarde 3](#_Toc440619727)

[Patroon 4: Leeg element met een attribuut 4](#_Toc440619728)

[Patroon 5: Gevuld element met een attribuut 4](#_Toc440619729)

[Patroon 6: Geneste elementen met verschillende namen 4](#_Toc440619730)

[Patroon 7: Geneste elementen met dezelfde naam 5](#_Toc440619731)

[Patroon 8: Namespace declaraties 5](#_Toc440619732)

# Inleiding

In deze notitie wordt een vertaling voorgesteld tussen XML en JSON in de context van StUF-berichtuitwisseling. De conversie is *omkeerbaar*, dat wil zeggen dat het resulterende JSON-bericht altijd terugvertaald kan worden naar het originele XML-bericht. Omkeerbaarheid is belangrijk omdat hiermee JSON-berichten gevalideerd kunnen worden met de bestaande XSD-schema’s voor StUF. Als de vertaling niet omkeerbaar is zouden er aparte JSON schema’s voor de StUF-berichten gedefinieerd moeten worden. Dat zou een behoorlijke extra beheerlast geven, met name het in overeenstemming houden van de XML- en JSON-schema’s. Bovendien is JSON Schema nog niet officieel gestandaardiseerd en minder expressief en uitgekristalliseerd als XML Schema (XSD).

Er is nog geen internationale standaard voor een (omkeerbare) vertaling tussen XML en JSON. Bijvoorbeeld, in XMLSpy levert een heen en weer vertaling niet hetzelfde bericht op. Misschien gaat die standaard er ook niet komen omdat er binnen elk toepassingsdomein weer andere eisen en wensen zijn. Echter binnen de context van StUF is een omkeerbare vertaling wel mogelijk. De voornaamste redenen hiervan zijn:

* StUF is een berichtstandaard en geen opmaaktaal. StUF maakt dus geen gebruik van mixed-content.
* In de XML-syntax van StUF komen de namen van subelementen maar één keer voor of de subelementen met identieke namen staan in volgorde

De vertaling die hieronder wordt beschreven is gebaseerd op een artikel van Stefan Goessner op XML.com.[[1]](#footnote-1)

# Mapping tussen XML en JSON

De onderstaande mapping gaat ervan uit dat de XML-elementen en –attributen voorzien zijn van een namespace-prefix wat doorgaans het geval is in StUF. De mapping werkt ook zonder namespace-prefixes. Indien de prefix “ns:” in de onderstaande vertaalregels wordt weggelaten is de vertaling nog steeds valide.

|  |  |  |
| --- | --- | --- |
|  | **XML** | **JSON** |
| 1 | <ns:elem/> | "ns:elem": null |
| 2 | <ns:elem xsi:nil=”true”/> | "ns:elem": {  “@xsi:nil”: true  } |
| 3 | <ns:elem>waarde</ns:elem> | "ns:elem": "waarde” |
| 4 | <ns:elem ns:attr=”waarde”/> | “ns:elem”: {  “@ns:attr”: “waarde”  } |
| 5 | <ns:elem ns:attr=”waarde1”>waarde2</ns:elem> | “ns:elem”: {  “@ns:attr”: “waarde1”,  “#text”: “waarde2”  } |
| 6 | <ns:elemA>  <ns:elemB>waarde1</ns:elemB>  <ns:elemC>waarde2</ns:elemC>  …  </ns:elemA> | “ns:elemA”: {  “ns:elemB”: “waarde1”,  “ns:elemC”: “waarde2”,  …  } |
| 7 | <ns:elemA>  <ns:elemB>waarde1</ns:elemB>  <ns:elemB>waarde2</ns:elemB>  …  </ns:elemA> | “ns:elemA”: {  “ns:elemB”: [  “waarde1”,  “waarde2”,  …  ]  } |
| 8 | <ns1:elem xmlns:ns1=”uri1” xmlns:ns2=”uri2”>  …  </ns1:elem> | “ns1:elem”: {  “@xmlns:ns1”: “uri1”,  “@xmlns:ns2”: “uri2”,  …  } |

In de bovenstaande mapping worden de volgende conventies gevolgd:

* Het @-symbool is gereserveerd voor het aanduiden van XML-attributen in JSON (zie patronen 2, 4, 5 en 8).
* De dubbele punt is gereserveerd voor de scheiding tussen de namespace-prefix en de naam van het element in JSON (deze conventie wordt in alle bovenstaande patronen toegepast).
* De term “xsi:nil” is gereserveerd voor de semantiek van het corresponderende XML-attribuut “xsi:nil” in JSON (zie patroon 2).
* De term “#text” is gereserveerd voor het aanduiden van de inhoud van een XML-element in JSON (zie patroon 5).
* De term “xmlns” is gereserveerd voor het aanduiden van namespace-declaraties in JSON (zie patroon 8).

Bovenstaande afspraken zijn nodig voor de terugvertaling van JSON naar XML. De consequentie hiervan is dat in de XML-syntax van StUF geen @-symbool gebruikt mag worden in attribuutnamen en de term “#text” niet mag voorkomen in elementnamen.

XML-expressies kunnen geneste c.q. lokale namespace-declaraties bevatten. De bovengenoemde vertaalpatronen gaan ervan uit dat alle namespace declaraties zich bevinden in het root-element van het te vertalen bericht. Indien dat niet het geval zal er een preprocessingsstap moeten plaatsvinden om alle geneste declaraties naar het rootelement te verhuizen. Een dergelijke transformatie is eenvoudig te automatiseren met een (XSLT)-scriptje.

# Patroon 1: Leeg element

Voor de volledigheid is dit patroon opgenomen hoewel het niet voorkomt in de huidige StUF-berichten. In StUF 3.01 moeten lege elementen voorzien zijn van een xsi:nil en/of een StUF:noValue attribute.

|  |  |
| --- | --- |
| **XML** | **JSON** |
| <ns:elem/> | "ns:elem": null |
| <BG:sub.telefoonnummer/> | "BG:sub.telefoonnummer": null |

# Patroon 2: Leeg element met attribute xsi:nil=”true”

Als het goed is zal dit patroon niet meer voorkomen in StUF 3.02 omdat het gebruik van xsi:nil dan niet meer is toegestaan omdat codegeneratoren niet goed om kunnen gaan met dit attribuut.

|  |  |
| --- | --- |
| **XML** | **JSON** |
| <ns:elem xsi:nil=”true”/> | "ns:elem": {  “@xsi:nil”: true  } |
| <BG:sub.telefoonnummer xsi:nil="true"/> | "BG:sub.telefoonnummer": {  "@xsi:nil": true  } |

# Patroon 3: Element met een waarde

In dit patroon worden alle waarden vertaald als een string, dus ook datatypes zoals integers en booleans worden als string naar JSON vertaald. Op XML-niveau kun je het onderscheid tussen deze datatypen niet zien, tenzij je het attribuut xsi:type gebruikt, echter die constructie komt in StUF 3.01 niet voor. Een andere optie is om in de vertaling het XML Schema erbij te betrekken. In dat geval kun je wel bepalen op welke datatypen de elementwaarde betrekking hebben. Gemakshalve is daar nu niet voor gekozen.

|  |  |
| --- | --- |
| **XML** | **JSON** |
| <ns:elem>waarde</ns:elem> | "ns:elem": "waarde” |
| <BG:geslachtsnaam>Korver</BG:geslachtsnaam> | "BG:geslachtsnaam": "Korver" |
| <BG:geboortedatum>20060731</BG:geboortedatum> | "BG:geboortedatum": "20060731" |

# Patroon 4: Leeg element met een attribuut

Voor de volledigheid is dit patroon opgenomen hoewel het niet voorkomt in de huidige StUF-berichten. In StUF 3.01 moeten lege elementen voorzien zijn van een xsi:nil en/of een StUF:noValue attribute.

|  |  |
| --- | --- |
| **XML** | **JSON** |
| <ns:elem ns:attr=”text”/> | “ns:elem”: {  “@ns:attr”: “text”  } |
| <BG:geslachtsnaam StUF:noValue=”waardeOnbekend”/> | “BG:geslachtsnaam”: {  “@StUF:noValue”: “waardeOnbekend”  } |

# Patroon 5: Gevuld element met een attribuut

|  |  |
| --- | --- |
| **XML** | **JSON** |
| <ns:elem ns:attr=”waarde1”>waarde2</ns:elem> | “ns:elem”: {  “@ns:attr”: “waarde1”,  “#text”: “waarde2”  } |
| <StUF:extraElement naam="urgentie">Hoog</StUF:extraElement> | “StUF:extraElement”: {  “@naam”: “urgentie”,  “#text”: “Hoog”  } |

# Patroon 6: Geneste elementen met verschillende namen

|  |  |
| --- | --- |
| **XML** | **JSON** |
| <ns:elemA>  <ns:elemB>waarde1</ns:elemB>  <ns:elemC>waarde2</ns:elemC>  …  </ns:elemA> | “ns:elemA”: {  “ns:elemB”: “waarde1”,  “ns:elemC”: “waarde2”,  …  } |
| <StUF:ontvanger>  <StUF:organisatie>KING</StUF:organisatie>  <StUF:applicatie>app001</StUF:applicatie>  <StUF:administratie>adm001</StUF:administratie>  </StUF:ontvanger> | “StUF:ontvanger”: {  “StUF:organisatie”: “KING”,  “StUF:applicatie”: “app001”,  “StUF:adminstratie”: “adm001”  } |

# 

# Patroon 7: Geneste elementen met dezelfde naam

|  |  |
| --- | --- |
| **XML** | **JSON** |
| <ns:elemA>  <ns:elemB>waarde1</ns:elemB>  <ns:elemB>waarde2</ns:elemB>  …  </ns:elemA> | “ns:elemA”: {  “ns:elemB”: [  “waarde1”,  “waarde2”,  …  ]  } |

Het onderstaande voorbeeld is een combinatie van patroon 5 en 7.

|  |  |
| --- | --- |
| **XML** | <StUF:extraElementen>  <StUF:extraElement naam="clientvolgnummer">1</StUF:extraElement>  <StUF:extraElement naam="vreemdelingnummer">23455</StUF:extraElement>  </StUF:extraElementen> |
| **JSON** | “StUF:extraElementen”: {  “StUF:extraElement”: [  {  “@naam”: “clientvolgnummer”,  “#tekst”: “1”  },{  “@naam”: “vreemdelingnummer”,  “#tekst”: “23455”  }  ]  } |

# 

# Patroon 8: Namespace declaraties

|  |  |
| --- | --- |
| **XML** | <JZ:vtoDi01 xmlns:StUF="http://www.egem.nl/StUF/StUF0301" xmlns:BG="http://www.egem.nl/StUF/sector/bg/0310" xmlns:ZKN="http://www.egem.nl/StUF/sector/zkn/0310" xmlns:JZ="http://www.stufstandaarden.nl/koppelvlak/jz0100" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  …  </JZ:vtoDi01> |
| **JSON** | "JZ:vtoDi01": {  "@xmlns:StUF": "http://www.egem.nl/StUF/StUF0301",  "@xmlns:BG": "http://www.egem.nl/StUF/sector/bg/0310",  "@xmlns:ZKN": "http://www.egem.nl/StUF/sector/zkn/0310",  "@xmlns:JZ": "http://www.stufstandaarden.nl/koppelvlak/jz0100",  "@xmlns:xmime": "http://www.w3.org/2005/05/xmlmime",  "@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",  …  } |

1. <http://www.xml.com/pub/a/2006/05/31/converting-between-xml-and-json.html> [↑](#footnote-ref-1)