



ElsterLohn

verwendete XML-Techniken

Version: 1.1

Stand : 20.07.2018





ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

Inhaltsverzeichnis

1	Allgemeines	3
1.1	Kurzbeschreibung	3
1.2	Änderungsübersicht	3
2	Einleitung	4
2.1	Links	4
3	XML	5
3.1	Allgemeines	5
3.2	Allgemeine XML Grundsätze	5
3.3	Wohlgeformtheit und Gültigkeit	7
3.4	XML- Version	7
3.5	Zeichenkodierung/Zeichenumfang	7
3.6	Namensraum	8
3.6.1	Allgemeines	8
3.6.2	Namensraumdeklaration	8
3.6.3	Namensräume im ElsterLohn-Verfahren	9
4	XML-Schemata	10
4.1	Allgemeines	10
4.2	Das Schema-Wurzelement	11
4.2.1	Namensräume in Schemata	11
4.2.2	elementFormDefault	11
4.2.3	attributeFormDefault	11
4.3	mehrere Schemata	11
4.3.1	Das include Tag	11
4.3.2	Das import Tag	12
4.4	Kommentare	12
4.5	Komplexe <-> einfache Strukturen	12
4.5.1	simple Typen	12
4.6	komplexe Typen	16
5	XML-Stylesheets	17
6	Parser und Parserfehlermeldungen	19
6.1	Fehlernummer: 200033002	19
6.1.1	Beispiele für nicht wohlgeformte XML-Dokumente und die zu erwartenden Fehlermeldungen	19
6.1.2	Beispiele für nicht valide XML-Dokumente und die zu erwartenden Fehlermeldungen	21
7	Die HTML-Dokumentation zu den Schemata	25
7.1	Allgemeines	25
7.2	Beschreibung	25



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

1 Allgemeines

1.1 Kurzbeschreibung

In diesem Dokument werden die im ElsterLohn-Verfahren verwendeten XML-Techniken beschrieben. Der Schwerpunkt wird dabei auf folgende Punkte gelegt:

- eine Kurzeinführung in die XML-Syntax
- die im ElsterLohn-Verfahren definierten XML-Schemata
- Parser und Parserfehlermeldungen
- die Verwendung von Stylesheets zur Visualisierung von XML-Dokumenten
- Eine Lesehilfe/Bedienungsanleitung zu der im ElsterLohn-Verfahren veröffentlichten HTML-Dokumentation der XML-Schemata.

1.2 Änderungsübersicht

Version	Bearbeiter	Änderungsdatum	Durchgeführte Änderung
1.0	Wilz	29.08.2006	Erstellung Ausgliederung aus der SST_Lohnsteuerbescheinigung Version 4.1.3
1.0.1	Wilz	15.10.2007	Kleinere Layoutänderungen
1.1	Adam	20.07.2018	Redaktionelle Überarbeitung und Hinweis auf geplante UTF-8 Unterstützung (vgl. 3.5)

Tabelle 1 / Änderungsübersicht



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

2 Einleitung

Dieses Dokument soll einen schnellen Einstieg in die ElsterLohn Dokumentationen ermöglichen. Desweiteren werden allgemeine XML- und XML-Schema-Grundlagen beschrieben. Insbesondere bezüglich der Grundlagen dient es nicht als Ersatz der entsprechenden Spezifikation, sondern soll lediglich als Einstieg in die XML-Technik dienen.

An einzelnen Stellen werden allgemeine Grundsätze für Datenlieferungen an ElsterLohn definiert.

2.1 Links

Nr	Link	Inhalt
1	https://www.elster.de	Die Portalseite des Projektes Mein ELSTER (Elektronische Steuer Erklärung). Über diese Seite können Sie sich umfassend über alle ELSTER-Projekte informieren. Des Weiteren werden alle Basisdienstleistungen: <ul style="list-style-type: none"> - Downloadbereich für Softwarehersteller - Anmeldung zum Newsletter - Registrierung als Softwarehersteller über diese Seite abgewickelt.
2	https://www.elster.de/elsterweb/infoseite/entwickler	Schnittstellendokumentationen von ELSTER
3	http://www.w3.org/XML/	Hier kann u.a die offizielle XML-Spezifikation des W3C-Konsortium heruntergeladen werden.
4	http://www.w3.org/XML/Schema	Hier kann u.a die offizielle XML-Schema-Spezifikation des W3C-Konsortium heruntergeladen werden
5	http://edition-w3c.de/	Für alle, die eine deutsche Dokumentation bevorzugen, bietet das „Edition W3C“-Projekt auch eine deutsche Übersetzung der XML-Spezifikation des W3C-Projektes an.
6	http://de.wikipedia.org/wiki/XML	Mehr zu XML bei Wikipedia
7	http://www.sql-und-xml.de/xml-lernen/xml-version-1.1-unicode.html	Unterschiede zwischen XML 1.0- und XML 1.1 - Standard
8	http://www.usegroup.de/software/xmltutorial/	Einführung in XML

Tabelle 2 / Links

In dieser Tabelle aufgelistet Links werden durch ein **L** und die laufende Nr referenziert z.B. **<L1>** für den Link auf die Portalseite <https://www.elster.de>



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

3 XML

3.1 Allgemeines

Im Projekt ElsterLohn erfolgt der Datenaustausch zwischen Datenlieferanten und ELSTER ausschließlich über das plattformunabhängige XML-Format.

3.2 Allgemeine XML Grundsätze

- Jedes XML-Dokument beginnt mit einer XML-Deklaration
z.B.: `<?xml version="1.0" encoding="ISO-8859-15"?>`

- Jedes XML-Dokument besitzt genau ein Wurzelement

Richtig	Falsch
<pre><?xml version="1.0" encoding="ISO-8859-15"?> <a> </pre>	<pre><?xml version="1.0" encoding="ISO-8859-15"?> <a> </pre>

- alle Tags mit Inhalt besitzen ein Beginn- und ein End-Tag

Richtig	Falsch
<pre><a> <c> </c> </pre>	<pre><a> <c> </c> </pre>

- alle Tags ohne Inhalt können auch mit `/>` abschließen

Richtig	Falsch
<pre><a> <b version="01"/> </pre>	<pre><a> <b version="1"> </pre>

- innerhalb des Wurzelementes sind alle Elemente eindeutig ineinander verschachtelt

Richtig	Falsch
<pre><a> <c> ... </c> </pre>	<pre><a> <c> </c> </pre>



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

- Tag und Attributnamen sind casesensitive

Richtig	Falsch
<pre><a> <c> ... </c> </pre>	<pre><a> <c> </C> </pre>

- ein Attribut darf nicht zweimal im gleichen Tag vorkommen

Richtig	Falsch
<pre><Person> <b familienname="Mustermann" Geburtsname="Musterfrau" vorname="Inge"/> </Person></pre>	<pre><a> <b name="Mustermann" name="Musterfrau" vorname="Inge"/> </pre>

- Attribute sind nur in dem „Start“-Tag zulässig

Richtig	Falsch
<pre><a> <b c="test"> </pre>	<pre><a> </b c="test"> </pre>

- Entities / Schlüsselbegriffe / Schlüsselwörter

In XML müssen einige Zeichen durch entsprechende „Entities“ ersetzt werden, da diese in XML besondere Bedeutungen haben. Folgende Zeichen müssen in XML durch folgende Entities ersetzt werden:

zu ersetzendes Zeichen	Entity
&	&
'	'
>	>
<	<
"	"

- Kommentare sind müssen mit „<!--“ beginnen und mit „-->“ enden

Richtig	Falsch
<pre><a> <!-- dies ist ein Kommentar --> </pre>	<pre><a> <!-- dieser Kommentar wird nicht richtig beendet -> </pre>
<pre><a> <!-- Hierdurch wird das ganze Tag „b“ mit seinem Inhalt auskommentiert. --> </pre>	<pre><a> <!-- dieser Kommentar führt zu einem Fehler, da das Tag „b“ nicht geöffnet wurde ... --> </pre>



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

- **verwendete Entitäten müssen vor ihrer Referenzierung deklariert worden sein**

Da im ELSTER Verfahren keine Entitäten verwendet werden / unterstützt werden, wird auf diesen Punkt nicht weiter eingegangen.

3.3 Wohlgeformtheit und Gültigkeit

Ein XML-Dokument ist „wohlgeformt“ (wellformed), wenn alle Markup-Regeln (die allgemeinen XML-Grundsätze) erfüllt werden.

Ein wohlgeformtes XML-Dokument kann zusätzlich auch „gültig“ (valide) sein, wenn es mit einer DTD¹ oder einer XSD² konform ist.³

3.4 XML- Version

Gemäß den XML-Grundsätzen beginnt jedes XML-Dokument mit der XML-Deklaration.

z.B.: `<?xml version="1.0" encoding="ISO-8859-15"?>`

In der XML-Deklaration wird die Version der zugrunde liegenden XML-Spezifikation definiert. Zurzeit sind 2 Standards vom W3C-Konsortium verabschiedet: „1.0“ und „1.1“. Die Unterschiede zwischen den beiden Standards werden unter [L7](#) genauer beschrieben.

Wichtig:

Auf Grund des im ELSTER Projekt verwendeten Parsers wird nur die XML-Version 1.0 unterstützt.

Sollte dennoch ein XML-Dokument mit der XML-Version 1.1 versendet werden, wird direkt beim senden der Daten folgender Fehlercode und folgender Fehlertext zurückgemeldet:

- `<Code>200127001</Code>`
- `<Text>Fehler beim Einspeichern der Daten: java.lang.RuntimeException: : XML version "1.1" is not supported.</Text>`

3.5 Zeichenkodierung/Zeichenumfang

Gemäß den XML-Grundsätzen beginnt jedes XML-Dokument mit der XML-Deklaration. Innerhalb der XML-Deklaration ist der Zeichensatz (encoding) anzugeben, in dem die Daten des XML-Dokumentes codiert sind.

z.B.: `<?xml version="1.0" encoding="ISO-8859-15"?>`

Wichtig:

Für alle übermittelten XML-Dokumente wird im ElsterLohn-Verfahren eine Codierung gemäß **ISO-8859-15** unterstellt. Eine serverseitige Zeichensatzwandlung findet nicht statt.

Hinweis:

Voraussichtlich im 2. Quartal 2019 wird neben der derzeit zulässigen Zeichencodierung gemäß ISO-8859-15 auch die Datenannahme von UTF-8 kodierten Daten im Zeichenumfang String.Latin ermöglicht werden.

¹ DTD = Dokument-Type-Definition

² XML-Schema-Definition

³ Im Elster-Projekt werden ausschließlich XML-Schemata zur Definition/Prüfung der XML-Dokumente verwendet, da diese weitergehende Prüfmechanismen bieten als die DTD.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

3.6 Namensraum

3.6.1 Allgemeines

Namensräume dienen der eindeutigen Identifikation von XML-Elementen. Da in XML grundsätzlich alle Tagnamen und Attributnamen frei wählbar sind, dient das Namensraumkonzept dazu, nichteindeutige XML-Elemente unterscheiden zu können.

z.B.:

```
<Name xmlns="http://www.elsterlohn.de/arbeitgeber"/>
  <Firmenname>...<Firmenname>
</Name>
```

und

```
<Name xmlns=http://www.elsterlohn.de/mitarbeiter/>
  <Familienname>...   < Familienname >
  <Geburtsname>...   < Geburtsname >
  <Vorname>...       < Vorname >
</Name>
```

Nur durch die Zuweisung eines „*eindeutigen*“ Namensraum, kann bereits beim Lesen des Elements „Name“ die weitere Verarbeitung anhand des Namensraums gesteuert werden.

Die offizielle Spezifikation zu XML-Namensräumen kann unter [<L3>](#) bezogen werden. (auch hierfür bietet das edition-W3C-Projekt eine geeignete Übersetzung an).

3.6.2 Namensraumdeklaration

Gemäß XML-Spezifikation gibt es verschiedene Möglichkeiten Namensräume für Elemente in XML-Dokumenten zu deklarieren. Die Deklaration kann einerseits mittels Attribut (hierdurch wird dann dieses Element und jedes seiner Kindelemente automatisch diesem Namensraum zugeordnet), andererseits mittels eines Präfix erfolgen (Im Falle einer Deklaration über ein Präfix, muss diesem vorher ein Namensraum zugewiesen werden).

Beispiele:

- Beispiel einer Deklaration mittels Attribut:

```
<Lohnsteuerbescheinigung xmlns="http://www.elsterlohn.de/2018-01/XMLSchema">
  <Dauer>...</Dauer>
<!-- Durch die zentrale Deklaration muss für das Kindelement „Dauer“ und alle anderen
      Kindelemente (auch die von Dauer) der Namensraum nicht erneut zugewiesen werden... -->
  ...
</Lohnsteuerbescheinigung>
```
- Beispiel einer Deklaration mittels Präfix:

```
<Elster xmlns:elo201801="http://www.elsterlohn.de/2018-01/XMLSchema ">
<!-- Deklaration des Präfix „elo201801“ im Wurzelement... -->
  <elo201801:Lohnsteuerbescheinigung>
    < elo201801:Dauer>...</ elo201801:Dauer>
    ...
  </ elo201801:Lohnsteuerbescheinigung>
</Elster>
```

Wichtig:

Im Elster-Verfahren werden Namensraumdeklarationen mittels Präfix **nicht** unterstützt.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

3.6.3 Namensräume im ElsterLohn-Verfahren

Das Wurzelement „Elster“ gehört grundsätzlich zum Elster-Namensraum:

- „<http://www.elster.de/2002/XMLSchema>“
Somit gehören grundsätzlich (soweit nichts Anderes deklariert ist) alle anderen Elemente dieses XML-Dokuments zum Elsternamensraum.

Für Lohnsteuerbescheinigungen des ElsterLohn-Verfahrens ist der Namenraum für die jährliche Version der Lohnsteuerbescheinigung wie folgt aufgebaut:

- „<http://www.elsterlohn.de/<kalenderjahr-versionsnummer>/XMLSchema>“
z.B. <http://www.elsterlohn.de/2018-01/XMLSchema>



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

4 XML-Schemata

4.1 Allgemeines

Das XML-Schema ist ein „neuer Ansatz“ zur Definition von Dokument Typen und damit zur Spezifikation von XML-Sprachen. Dabei bieten XML-Schemata weitergehende Möglichkeiten XML-Dokumente zu beschreiben, als die aus dem SGML (Standardized Generalized Markup Language) -Standard übernommene Document Type Definition (DTD).

Neben der Möglichkeit die Struktur eines XML-Dokumentes zu beschreiben, lassen sich mittels XML-Schemata (im Gegensatz zur DTD) auch die Inhalte der einzelnen Elemente genau spezifizieren und beschränken.

Dabei stellt ein XML-Schema Regeln zur Verfügung, nach denen ein XML-Dokument auf seine Gültigkeit hin überprüft wird (Validierung). Da jedes XML-Schema auch ein XML-Dokument ist, lässt es sich selbst auch durch ein Schema oder eine DTD beschreiben.

Unterschiede zur DTD

- XML-Schema ist XML.
D.h. es existiert sowohl eine DTD zur Beschreibung von XML-Schemata als auch ein selbstbeschreibendes XML Schema. Ein gegen ein XML-Schema geprüftes XML-Dokument (document instance) kann wohlgeformt (well-formed) sein oder gültig (valid). Der Test auf Gültigkeit beinhaltet neben der Überprüfung auf korrekte Reihenfolge und Struktur der Tags auch die Überprüfung auf Einhaltung der Wertebereiche und gültige Datentypen.
- Umfangreicher Vorrat an vordefinierten, einfachen Datentypen.
Ein XML-Schema bietet zahlreiche Datentypen an, zusammen mit der Möglichkeit, den Wertebereich explizit anzugeben. Mit diesen Basistypen können weitere, komplexe Elementtypen definiert werden. Die einmalige Definition von Daten- oder Elementtypen bei u.U. häufiger Verwendung fördert zum einen die Lesbarkeit des Schemas als auch die spätere Verarbeitung durch einen parser/validator, etc.
- XML-Schemata erlauben explizites Gruppieren von Attributen.
Attribute, die wiederkehrend in Gruppen verwendet werden (z.B. HTML: width + height) können nun als Attributgruppe einmalig definiert werden. Dieser Mechanismus, der bei DTD's mit Parameter Entities realisiert wurde, kann nun explizit angegeben und damit auch von einem parser/validator genutzt werden.
- Ein XML-Schema ermöglicht die Definition neuer Elementtypen auf Basis vorangegangener Definitionen (Vererbung). Durch die Erweiterung / Einschränkung bestehender komplexer Elementtypen um weitere Elemente oder Attribute lassen sich gleichartige Elementfamilien deklarieren.
- Unterstützung von Namensräumen (namespaces).
Durch die Berücksichtigung von Namensräumen lassen sich document instances erstellen, die auf Elemente in verschiedenen XML-Schema-Beschreibungen zugreifen. Elemente mit gleichem Namen, aber unterschiedlicher Struktur können, sofern durch Namensräume getrennt, in einer document instance erscheinen.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

4.2 Das Schema–Wurzelement

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.elster.de/2002/XMLSchema"
  xmlns:elster="http://www.elster.de/2002/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
```

4.2.1 Namensräume in Schemata

XML-Namensräume bieten eine einfache Möglichkeit, um Element- und Attributnamen, die in "Extensible Markup Language"-Dokumenten verwendet werden können, eindeutig zu benennen. Die Element- und Attributnamen werden mit Namensräumen verknüpft, die durch URI-Verweise identifiziert werden.

- der „targetNamespace“
Mit dem Attribut „targetNamespace“ wird das gesamte Dokument (alle in diesem Dokument enthaltenen Elemente, simple- und complexTypes) unter den definierten Zielnamensraum gestellt.
 - das Präfix „xs“
Mit dem Attribut „xmlns:xs="http://www.w3.org/2001/XMLSchema"“ wird dem Präfix „xs“ der XML-Schema-Namensraum zugeordnet. Durch diese explizite Kennzeichnung der XML-Schema-Elemente und -Typen, können diese besser von selbst definierten Elementen und Typen unterschieden werden.
 - das Präfix „elster“
Das Präfix „elster“ wird benötigt um vollqualifiziert auf die definierten Elemente und Typen über „Prefix:localName“ referenzieren zu können.

4.2.2 elementFormDefault

Um zu spezifizieren, dass alle lokalen Elemente qualifiziert angegeben werden müssen, setzt man den Wert von elementFormDefault auf qualified.

4.2.3 attributeFormDefault

Anders als Elemente müssen Attribute, die qualifiziert werden müssen, ein Präfix tragen, weil *XML-Namespaces* keinen voreingestellten Namensraum für Attribute vorsehen. Deswegen wird dieses Flag auf unqualified gesetzt.

4.3 mehrere Schemata

Ein XML-Schema kann physikalisch aus vielen verschiedenen Schemata-Dateien bestehen. Hierzu müssen die weiteren Schemata lediglich in das „Root“-Schema eingebunden werden.

Für die Integration weiterer Schemata gibt es zwei Alternativen:

- mittels „include“ wenn das zu integrierende Schema denselben Namensraum hat
- mittels „import“ wenn das zu integrierende Schema aus einem anderen Namensraum stammt.

4.3.1 Das include Tag

Mit dem Include Befehl wird eine Schemadatei (referenziert über das Attribut schemaLocation) in das aktuelle Schema-Dokument eingebunden.

```
<xs:include schemaLocation="./allg000008.xsd">
  <xs:annotation>
    <xs:documentation xml:lang="DE">In der allg.xsd werden simple Typen, die in weiteren Schemata
      verwendet werden, global definiert z.B. die eTIN </xs:documentation>
    </xs:annotation>
  </xs:include>
```



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

4.3.2 Das import Tag

Mit dem Import-Befehl wird eine Schemadatei aus einem anderen Namensraum (referenziert über das Attribut `schemaLocation`) in das aktuelle Schema-Dokument eingebunden.
Ein Import eines Schemas, das demselben Namensraum angehört ist nicht möglich, dies kann nur durch das `include`-Statement erfolgen.

```
<xs:import namespace="" schemaLocation=""/>
```

4.4 Kommentare

Wie in jedem anderen XML-Dokument werden Kommentare durch „`<!--`“ eingeleitet und durch „`-->`“ beendet. Zusätzlich können Kommentare innerhalb des `<documentation>`-Tag eingefügt werden. Das Attribut `xml:lang="DE"` bestimmt die Sprache des Kommentars (optionales Attribut). Das `<documentation>`-Tag ist nur innerhalb des `<annotation>`-Tag zulässig.

```
<xs:simpleType name="AnschriftenzusatzSType">
  <xs:annotation>
    <xs:documentation xml:lang="DE">In der allg.xsd werden simple Typen, die in weiteren Schemata
      verwendet werden, global definiert z.B. die eTIN </xs:documentation>
    </xs:annotation>
  </xs:simpleType>
```

4.5 Komplexe <-> einfache Strukturen

In XML Schema gibt es eine grundlegende Unterscheidung zwischen komplexen Typen (complex types), die Elemente enthalten und Attribute tragen dürfen, und einfachen Typen (simple types), bei denen das nicht erlaubt ist. Außerdem gibt es eine wichtige Unterscheidung zwischen Definitionen, die neue Typen (einfach oder komplex) erzeugen, und Deklarationen, die ermöglichen, dass Elemente und Attribute mit speziellen Namen und Typen (einfach oder komplex) in Instanzdokumenten auftreten.

Neue komplexe Typen werden mit dem `complexType`-Element definiert. Solche Definitionen enthalten üblicherweise eine Menge von Element-Deklarationen, Element-Referenzen und Attribut-Deklarationen. Die Deklarationen sind selbst keine Typen, sondern Assoziationen zwischen einem Namen und den Beschränkungen (constraints), welche das Auftreten dieses Namens in Dokumenten regeln, die durch das entsprechende Schema bestimmt werden. Elemente werden mit dem `element`-Element deklariert und Attribute mit dem `attribute`-Element.

4.5.1 simple Typen

Das Schema bietet die Möglichkeit, alle bestehende Datentypen (Integer, String... vgl. hierzu: XML Schema Teil 2: Datentypen) abzuleiten und zu erweitern, so dass eigene Datentypen geschaffen werden können. Erst durch die Vielzahl der vordefinierten Typen und die Möglichkeit eigene Typen zu erstellen lässt sich neben der „wellformed“-Prüfung, der Strukturprüfung (die auch die DTD ermöglicht), auch inhaltlich jeder einzelne Wert eines Elements oder Attributs prüfen.

Verwendung der einfachen Datentypen der XML-Schema-Referenz

```
<xs:element name="ETIN" type="xs:string">
  <xs:annotation>
    <xs:documentation xml:lang="DE">die eTIN </xs:documentation>
  </xs:annotation>
</xs:element>
```



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

4.5.1.1 Allgemeines:

Die einzelnen im Folgenden beschriebenen Einschränkungen sind nicht abschließend und lassen sich teilweise auch miteinander kombinieren. Grundsätzlich ist nicht jede Einschränkung bei jedem Basis-Datentypen zulässig. Welche Kombinationen von Einschränkungen für welchen Basis-Datentypen zulässig sind, entnehmen Sie bitte der W3C-Dokumentation.

Hinweis:

Beim Prüfen ob ein XML-Schema-Dokument valide ist, wird dieses nur auf Wohlgeformtheit (es ist ja ein XML-Dokument) und auf eine zulässige Struktur mit zulässigen Datentypen geprüft.

Ein XML-Schema-Dokument ist zum Beispiel nicht valide, wenn ein einfacher Datentyp auf eine Länge „B“ beschränkt werden soll, da bei dieser Einschränkung ein Integer als Parameter erwartet wird.

Eine logische Prüfung, z.B. dass die Mindestlänge eines Strings 100 Zeichen ist und die Maximallänge „nur“ 50 Zeichen beträgt, wird bei diesen Prüfungen durch den Parser nicht gefunden.

4.5.1.2 Schablonen

Einfache Datentypen (die der XML-Schema Definition als auch eigene) lassen sich durch Schablonen (Pattern) weiter einschränken.

Grundsätzlich entsprechen die Regeln der Pattern-Definition denen der regulären Ausdrücke – in Zweifelfragen sollte jedoch die XML-Schema-Referenz des W3C-Konsortiums herangezogen werden.

Beispiel:

```
<xs:simpleType name="ETINSType">
  <xs:annotation>
    <xs:documentation xml:lang="DE"> die eTIN </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Z]{8}\d{2}[A-Z]{1}\d{2}[A-Z]{1}"/>
  </xs:restriction>
</xs:simpleType>
```

Erklärung des Patterns:

In diesem Beispiel wird der einfache Datentyp „ETINSType“ deklariert. Dieser einfache Datentyp ist vom Typ xs:string abgeleitet.

In diesem Beispiel verweist „xs“ auf den Namensraum „http://www.w3.org/2001/XMLSchema“ und verweist auf den Standard-String-Typ der XML-Schema-Sprache.

Des Weiteren wird dieser String erweitert um ein Pattern (Schablone), das besagt:

- die ersten 8 Zeichen müssen vom Typ A-Z sein (es werden nur Großbuchstaben akzeptiert)
- die folgenden 2 Zeichen müssen vom Typ dezimal (0-9) sein
- das folgende Zeichen muss vom Typ A-Z sein
- die folgenden 2 Zeichen müssen vom Typ dezimal (0-9) sein
- das letzte Zeichen muss vom Typ A-Z sein

Hinweis:

Es sind auch mehrere verschiedene Patterns zulässig, dabei muss der zu prüfende einfache Datentyp nur einem dieser Patterns entsprechen.

Soll der simple Datentyp mehreren Patterns entsprechen, müssen diese in verschiedenen simplen Datentypen definiert werden, die dann als Basis eines anderen simplen Typs dienen.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

4.5.1.3 Feldlänge

Die Länge der einzelnen Datentypen lässt sich mittels folgender Einschränkungen:

- minLength
- maxLength
- length

festlegen.

Beispiele:

```
<xs:simpleType name="allg_HausnummernZusatzSType">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="15"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="allg_HerstellerIDSType">
  <xs:restriction base="xs:string">
    <xs:length value="5"/>
  </xs:restriction>
</xs:simpleType>
```

Erklärung der Einschränkungen:

Im ersten Beispiel wird der zulässige Länge des Typs "allg_HausnummernZusatzSType" auf mindestens ein Zeichen und maximal 15 Zeichen begrenzt.

Im zweiten Beispiel wird die zulässige Länge fix auf 5 Zeichen begrenzt.

Hinweis: Bei Längenbeschränkungen von Strings werden die „Platzhalter“ wie z. B.: das „&“, nur als ein Zeichen gewertet.

Beispiel: Der Inhalt des Tags `<ArbGName>Mustermann & Mustermann GmbH</ArbGName>` besteht aus 32 Zeichen – nach dem Schema ist der ArbGName mit max. 30 Stellen zulässig. Da das „&“, jedoch nur als 1 Zeichen gewertet wird, beträgt die interne (für den Parser- und den Validierungsprozess maßgebliche) Länge nur 28 Stellen und ist damit gültig.

4.5.1.4 Aufzählungen

Datenfelder vom Typ „string“ bzw. dessen Ableitungen lassen sich durch abschließende Aufzählungen einschränken.

Dies erfolgt über das Element `<enumeration>`.

Beispiel:

```
<xs:simpleType name="allg_BundeslandSType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Baden-Württemberg"/>
    <xs:enumeration value="Bayern"/>
    <xs:enumeration value="Berlin"/>
    <xs:enumeration value="Brandenburg"/>
    <xs:enumeration value="Bremen"/>
    <xs:enumeration value="Hessen"/>
  </xs:restriction>
</xs:simpleType>
```



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

Erklärung der Einschränkungen:

In diesem Fall wird casesensitive der simple Datentyp auf eines der in der Auflistung aufgeführten Felder geprüft. Während „Bayern“ korrekt ist, wäre „bayern“ falsch.

4.5.1.5 Wertebereich

Während bei von String abgeleiteten Datentypen eine Beschränkung über die Feldlänge erfolgen kann, wird bei primitiven Zahlen (z.B. float, double, decimal und deren Ableitungen) die Einschränkungen über Wertebereiche erfolgen. Hierzu dienen folgende Befehle:

- maxExklusiv
- minExklusiv
- minInklusiv
- maxInklusiv

Beispiele:

```
<xs:simpleType name="Monat1">
  <xs:restriction base="xs:integer">
    <xs:maxInklusiv value="12"/>
    <xs:minInklusiv value="1"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="Monat2">
  <xs:restriction base="xs:integer">
    <xs:maxExklusiv value="13"/>
    <xs:minExklusiv value="0"/>
  </xs:restriction>
</xs:simpleType>
```

Erklärung der Einschränkungen:

In beiden Fällen ist der Datentyp von Integer abgeleitet (muss also eine Ganzzahl sein). Der Wertebereich ist in beiden Beispielen 1-12.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

4.6 komplexe Typen

4.6.1.1 Sequenzen

Innerhalb einer Sequenz können einzelne Elemente mehrfach (in einer vorher nicht fest bestimmten Anzahl) geliefert werden. Dafür muss bei der Elementdeklaration mittels des Attributs „maxoccurs“ die Maximalanzahl dieser Elemente bestimmt werden – ggf. „unbounded“ für unbeschränkt.

```
<xs:complexType name="BesteuerungsgrundlagenCType">
  <xs:sequenz>
    <xs:element name="Kirchensteuerabzug" type="elster:KirchensteuerabzugCType"/>
    <xs:element name="Kinderfreibetrag" type="elster:KinderfreibetragCType"/>
  ...</xs:sequenz>
</xs:complexType/>
```

4.6.1.2 Choice

Innerhalb eines complexType können auch wahlweise unterschiedliche Elemente geschachtelt werden durch eine entsprechende Verschachtelung von Sequenzen und Choice können somit dynamische Strukturen gut abgebildet werden.

Zum Beispiel:

```
<xs:complexType name="NutzdatenCType">
  <xs:sequenz>
    <xs:choice>
      <xs:element name="Kirchensteuerabzug" type="elster:KirchensteuerabzugCType"/>
      <xs:element name="Kinderfreibetrag" type="elster:KinderfreibetragCType"/>
    <xs:choice>
  ...</xs:sequenz>
</xs:complexType/>
```



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

5 XML-Stylesheets

Mittels XML-Stylesheets können XML-Dateien auf „einfache“ Weise transformiert werden z.B. um diese in einem Browser zu visualisieren.

Beispiel: XML-Datei

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<?xml-stylesheet href="stylesheet.xml" type="text/xsl"?>
<Elster xmlns="http://www.elster.de/2002/XMLSchema">
  <Person geschlecht="M">
    <Familiennamen>
      <Name>Ühlmann</Name>
      <Vorname>Herbert</Vorname>
    </Familiennamen>
    <Geburtsdatum>19760421</Geburtsdatum>
    <Adresse>
      <Str>In Der Tracht</Str>
      <Ort>Köln</Ort>
      <PLZ>50939</PLZ>
    </Adresse>
  </Person>
</Elster>
```

Beispiel : Stylesheet (stylesheet.xml)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:elster="http://www.elster.de/2002/XMLSchema"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" indent="no" encoding="ISO-8859-1"/>
  <xsl:template match="elster:Elster">
    <html> <head> <title>Personenanzeige</title> </head> <body>
  <b>Personenanzeige</b><br></b>
    <xsl:if test="elster:Person[@geschlecht='M']">Herrn</xsl:if>
    <xsl:if test="elster:Person[@geschlecht='W']">Frau</xsl:if>
    <br></br>
    <c>
      <xsl:value-of select="elster:Person/elster:Familiennamen/elster:Vorname"/>
    <c> </c>
      <xsl:value-of select="elster:Person/elster:Familiennamen/elster:Name"/>
    </c>
    <br></br>
    <xsl:value-of select="elster:Person/elster:Adresse/elster:Str"/> <br></br>
    <xsl:value-of select="elster:Person/elster:Adresse/elster:PLZ"/>
    <c> </c>
    <xsl:value-of select="elster:Person/elster:Adresse/elster:Ort"/>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

Im oben genannten Beispiel wird die XML-Datei durch die Verknüpfung zum XML-Stylesheet in ein HTML-Format gewandelt, so dass die Datei, wenn sie z.B. in einem Browser geöffnet wird, wie das im Stylesheet definierte HTML aufbereitet angezeigt wird.

Personenanzeige

Herrn
Herbert Uhlmann
In Der Tracht
50939 Köln



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

6 Parser und Parserfehlermeldungen

6.1 Fehlernummer: 200033002

Der Fehler 200033002 kann folgende Ursachen haben:

- das übermittelte XML-Dokument ist nicht wohlgeformt, das heißt es entspricht nicht der Spezifikation des W3C-Konsortiums. [<L7>](#).
- das übermittelte XML-Dokument ist ungültig, das heißt die Gültigkeitsprüfung⁴ gegen die XML-Schemata hat einen oder mehrere Fehler gemeldet.

In beiden Fällen wird die Fehlernummer 200033002 zurückgemeldet.

Im Fehlertext wird der Fehler näher spezifiziert.

Der Fehlertext beginnt grundsätzlich mit „*Fehler beim Parsen der XML-Daten* : “ erweitert um den vom Xerces⁵ gemeldeten Fehlertext.

Die Fehlertexte werden mit jeder Xerces-Version modifiziert. Die in diesem Dokument beschriebenen Fehlertexte wurden mit der in Produktion befindlichen Xerces-Version erzeugt.

Ein Fehler bedingt oftmals mehrere Fehlertexte.

6.1.1 Beispiele für nicht wohlgeformte XML-Dokumente und die zu erwartenden Fehlermeldungen

6.1.1.1 ein Tag wird nicht korrekt geschlossen

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Dauer jahr="2018"> <Anfang>0101 <Ende>3112</Ende> </Dauer></pre>	<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>

Fehler:

Das Tag **Anfang** wird nicht beendet.

folgende Fehlertexte werden vom Xerces gemeldet

- org.xml.sax.SAXParseException: Element type "Ende" must be declared.
- org.xml.sax.SAXParseException: The element type "Anfang" must be terminated by the matching end-tag "</Anfang>".
- org.xml.sax.SAXParseException: Element type "Allgemein" must be declared.
- org.xml.sax.SAXParseException: Element type "ETIN" must be declared.
- org.xml.sax.SAXParseException: Element type "Ordnungsmerkmal" must be declared.
- ... (*jedes Element und Attribut wird nun aufgelistet*)
- org.xml.sax.SAXParseException: The element type "Anfang" must be terminated by the matching end-tag "</Anfang>".

⁴ Die Gültigkeitsprüfung eines XML-Dokument gegen ein XML-Schemata wird auch Validierung genannt.

⁵ Xerces ist ein Parser aus dem Apache-Projekt. Mehr zum Xerces unter <http://xml.apache.org/>



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Dauer> </Ende></pre>	<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>

Fehler:

Das Tag Ende (ein Kindelement von Dauer) muss vor Dauer beendet sein.

folgende Fehlertexte werden vom Xerces gemeldet

- org.xml.sax.SAXParseException: The element type "Ende" must be terminated by the matching end-tag "</Ende>".
- org.xml.sax.SAXParseException: Datatype error: In element 'Ende' : Value '3112' does not match regular expression facet '[\p{L}]{1}[\p{P}]{0}[\p{Nd}]{1}[\p{L}]{1}[\p{P}]{0}[\p{Zs}]{0}[\p{Nd}]{1}[\p{L}]{1}[\p{Nd}]{1}[\p{P}]{0}[\p{L}]{1}[\p{P}]{0}[\p{Nd}]{1}..
- org.xml.sax.SAXParseException: Element type "Allgemein" must be declared.
- org.xml.sax.SAXParseException: Element type "ETIN" must be declared.
- org.xml.sax.SAXParseException: Element type "Ordnungsmerkmal" must be declared.
- ... *(jedes Element und Attribut wird nun aufgelistet)*
- org.xml.sax.SAXParseException: The element type "Dauer" must be terminated by the matching end-tag "</Dauer>".

6.1.1.2 ein Attribut wird nicht korrekt geschlossen

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>	<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>

folgende Fehlertexte werden vom Xerces gemeldet

- org.xml.sax.SAXParseException: The value of attribute "jahr" must not contain the '<' character.
- ...
- org.xml.sax.SAXParseException: Element type "Dauer" must be followed by either attribute specifications, ">" or "/>".
- org.xml.sax.SAXParseException: Element type "Familiennamen" must be declared.
- org.xml.sax.SAXParseException: Element type "Name" must be declared.
- org.xml.sax.SAXParseException: Element type "Vorname" must be declared.
- ...
- org.xml.sax.SAXParseException: The element type "Lohnsteuerbescheinigung" must be terminated by the matching end-tag "</Lohnsteuerbescheinigung>".
- org.xml.sax.SAXParseException: The element type "Lohnsteuerbescheinigung" must be terminated by the matching end-tag "</Lohnsteuerbescheinigung>".
- org.xml.sax.SAXParseException: The content of element type "Lohnsteuerbescheinigung" must match "(Dauer,Allgemein,Besteuerungsmerkmale,Besteuerungsgrundlagen)".



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

6.1.2 Beispiele für nicht valide XML-Dokumente und die zu erwartenden Fehlermeldungen

6.1.2.1 falscher bzw. nicht zugewiesener Namespace

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Lohnsteuerbescheinigung xmlns="http://www.elsterlohn.de/" art="ELStAM" version="201801"></pre>	<pre><Lohnsteuerbescheinigung xmlns="http://www.elsterlohn.de/2018- 01/XMLSchema" art="ELStAM" version="201801"></pre>

folgende Fehlertexte werden vom Xerces gemeldet

- Fehler = org.xml.sax.SAXParseException: General Schema Error: Grammar with uri: http://www.elsterlohn.de/ , can not be found; schema namespace may be wrong: Xerces supports schemas from the "http://www.w3.org/2001/XMLSchema" namespace or the instance document's namespace may not match the targetNamespace of the schema.
- org.xml.sax.SAXParseException: Element type "Lohnsteuerbescheinigung" must be declared.
- org.xml.sax.SAXParseException: Attribute "art" must be declared for element type "Lohnsteuerbescheinigung".
- org.xml.sax.SAXParseException: Attribute "version" must be declared for element type "Lohnsteuerbescheinigung".
- org.xml.sax.SAXParseException: General Schema Error: Grammar with uri: http://www.elsterlohn.de/ , can not be found; schema namespace may be wrong: Xerces supports schemas from the "http://www.w3.org/2001/XMLSchema" namespace or the instance document's namespace may not match the targetNamespace of the schema.
- org.xml.sax.SAXParseException: Element type "Dauer" must be declared.
- org.xml.sax.SAXParseException: Attribute "jahr" must be declared for element type "Dauer".
- (...)
- org.xml.sax.SAXParseException: The content of element type "Nutzdaten" must match "((Lohnsteuerbescheinigung|Lohnsteuerbescheinigung))".

6.1.2.2 falscher Wert eines Attributs

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Dauer jahr="18"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>	<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>

folgender Fehlertext wird vom Xerces gemeldet

- org.xml.sax.SAXParseException: Datatype error: 6 is not one of the specified enum values.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

6.1.2.3 falsche Reihenfolge der Tags einer Sequenz

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Dauer jahr="18"> <Ende>3112</Ende> <Anfang>0101</Anfang> </Dauer></pre>	<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>

Fehler:

Die Reihenfolge der Tags „Anfang“ und „Ende“ wurde vertauscht.

folgender Fehlertext wird vom Xerces gemeldet

- org.xml.sax.SAXParseException: The content of element type "Dauer" must match "(Anfang,Ende)".

6.1.2.4 Ein Pflicht-Tag wurde nicht übermittelt

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Dauer jahr="18"> <Ende>3112</Ende> </Dauer></pre>	<pre><Dauer jahr="2018"> <Anfang>0101</Anfang> <Ende>3112</Ende> </Dauer></pre>

Fehler:

Es fehlt das Tag „Anfang“.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: The content of element type "Dauer" is incomplete, it must match "(Anfang,Ende)".

6.1.2.5 Mindestlänge eines Elements wurde nicht eingehalten

fehlerhaftes Beispiel	korrektes Beispiel
<pre><Name></ Name></pre>	<pre><Name>Nachname</ Name></pre>

Fehler:

Der Inhalt des Tags Name fehlt, die Mindestlänge von einem Zeichen wurde unterschritten.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: cvc-pattern-valid: Value "" is not facet-valid with respect to pattern "[\p{L}|\p{P}|\p{Nd}][\p{L}|\p{P}|\p{Zs}|\p{Nd}]*[\p{L}|\p{Nd}|\p{P}][\p{L}|\p{P}|\p{Nd}]" for type 'allg_NachNameSType'.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

6.1.2.6 Maximallänge eines Elements wurde nicht eingehalten

fehlerhaftes Beispiel	korrektes Beispiel
<code><Name>Mustermann mit Bart und Brille ohne Bauch ist Nichtraucher</ Name></code>	<code><Name>Nachname</ Name></code>

Fehler:

Der Inhalt des Tags Name überschreitet die Maximallänge für den Nachnamen.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: Datatype error: In element 'Name' : Value 'Mustermann mit Bart und Brille ohne Bauch ist Nichtraucher' with length '58' exceeds maximum length facet of '30'.

6.1.2.7 Prüfungen von Formaten

fehlerhaftes Beispiel	korrektes Beispiel
<code><Name>Mustermann </ Name></code>	<code><Name>Mustermann</ Name></code>

Fehler:

Der Inhalt des Tags Name entspricht nicht dem definierten Format, da der Nachname mit einem „ „ (Space) endet.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: Datatype error: In element 'Name' : Value 'Mustermann ' does not match regular expression facet
`'[p{L}][p{P}][p{Nd}][p{L}][p{P}][p{Zs}][p{Nd}]*[p{L}][p{Nd}][p{P}][p{L}][p{P}][p{Nd}]'`...

fehlerhaftes Beispiel	korrektes Beispiel
<code><AuslandsPLZ>NL12345678</AuslandsPLZ></code>	<code>< AuslandsPLZ>NL 12345678</ AuslandsPLZ></code>

Fehler:

Der Inhalt des Tags AuslandsPLZ entspricht nicht den definierten Format, da es zwingend „NL_****“ lauten muss.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: Datatype error: In element 'AuslandsPLZ' : Value 'NL12345678' does not match regular expression facet 'AFG.*|ET.*|AL.*|DZ.*|AJ.*|AS.*|AND.*|AGO.*|ANG.*|AT.*|ANT.*|AQU.*|RA.*|ARM.*|ASE.*|ETH.*|AUS.*|BS.*|BRN.*|BD.*|BDS.*|B.*|BH.*|DY.*|BER.*|BHT.*|BOL.*|BIH.*|RB.*|BR.*|BJ.*|BRU.*|BG.*|HV.*|RU.*|CUE.*|RCH.*|TJ.*|COI.*|CR.*|CI.*|DK.*|WD.*|DOM.*|DSC.*|EC.*|ES.*|ERI.*|EST.*|FAL.*|FR.*|FJI.*|FIN.*|F.*|FG.*|FP.*|GAB.*|WAG.*|GEO.*|GH.*|GIB.*|WG.*|GR.*|GRO.*|GB.*|GUA.*|GUM.*|GCA.*|RG.*|GUB.*|GUY.*|RH.*|HCA.*|HOK.*|IND.*|RI.*|MAN.*|IRQ.*|IR.*|IRL.*|IS.*|IL.*|I.*|JA.*|J.*|YEM.*|JOR.*|YU.*|KAI.*|K.*|CAM.*|CDN.*|KAN.*|CV.*|KAS.*|QAT.*|EAK.*|KIS.*|KIB.*|CO.*|KOM.*|RCB.*|ZRE.*|KOR.*|ROK.*|HR.*|C.*|KWT.*|LAO.*|LS.*|LV.*|RL.*|LB.*|LAR.*|FL.*|LT.*|L.*|MAC.*|RM.*|MK.*|MW.*|MAL.*|BIO.*|RMM.*|M.*|MA.*|MAR.*|MAT.*|RIM.*|MS.*|MAY.*|MEX.*|MIK.*|MD.*|MC.*|MON.*|MOT.*|MOZ.*|MYA.*|SWA.*|NAU.*|INEP.*|NKA.*|NZ.*|NIC.*|NL.*|NLA.*|RN.*|WAN.*|NIU.*|NMA.*|N.*|MAO.*|A.*|PK.*|PAL.*|PA.*|PNG.*|PY.*|PIN.*|PE.*|RP.*|PIT.*|PL.*|P.*|PRI.*|REU.*|RWA.*|RO.*|RUS.*|PIE.*|SOL.*|Z.*|WS.*|RSM.*|STP.*|SAU.*|S.*|CH.*|SN.*|SY.*|WAL.*|ZW.*|SGP.*|SK.*|SLO.*|SP.*|E.*|CL.*|HEL.*|SCN.*|WL.*|WV.*|ZA.*|SUD.*|SME.*|SD.*|SYR.*|TAD.*|RC.*|EAT.*|T.*|OTI.*|TG.*|TOK.*|TON.*|TT.*|CHD.*|CZ.*|TN.*|TR.*|TUR.*|TUC.*|TUV.*|EAU.*|UA.*|H.*|ROU.*|USB.*|VAN.*|V.*|YV.*|UAE.*|USA.*|VN.*|BY.*|RCA.*|CY.*|SCG.*|.*'.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

6.1.2.8 Prüfungen gegen Aufzählungslisten

fehlerhaftes Beispiel	korrektes Beispiel
<code><Kirchensteuerabzug konfession="xx" gueltig_ab="0101" /></code>	<code><Kirchensteuerabzug konfession="rk" gueltig_ab="0101" /></code>

Fehler:

Der Inhalt des Attributs „konfession“ ist nicht in der Konfessionsliste enthalten, der Inhalt ist daher ungültig.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: Datatype error: Value 'xx' must be one of [ak, ev, fa, fb, fg, fm, fr, fs, ib, il, is, iw, jd, jh, lt, na, rf, rk, --].

6.1.2.9 Attribut wurde vergessen

fehlerhaftes Beispiel	korrektes Beispiel
<code><Kirchensteuerabzug konfession="rk"/></code>	<code><Kirchensteuerabzug konfession="rk" gueltig_ab="0101" /></code>

Fehler:

Zum Kirchensteuerabzug fehlt das Attribut „gueltig_ab“.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: Attribute "gueltig_ab" is required and must be specified for element type "Kirchensteuerabzug".

6.1.2.10 Entities werden nicht korrekt maskiert

fehlerhaftes Beispiel	korrektes Beispiel
<code><ArbGName>Muster & amp; Co KG</ArbGName></code>	<code><ArbGName>Muster &amp; Co KG</ArbGName></code>

Fehler:

Das & muss durch „&“ maskiert werden, Leerzeichen führen dazu, dass diese Referenz nicht korrekt aufgelöst werden kann.

folgender Fehlertext wird vom Xerces gemeldet:

- org.xml.sax.SAXParseException: The entity name must immediately follow the '&' in the entity reference.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

7 Die HTML-Dokumentation zu den Schemata

7.1 Allgemeines

Die HTML-Dokumentation ist eine versions-bezogene Dokumentation und nur für diese eine Version gültig. Für jede Version wird eine eigene HTML-Dokumentation erstellt und veröffentlicht.

Um die HTML-Dokumentation nutzen zu können, muss diese entpackt werden, da andernfalls die Hyperlinks nicht ordnungsgemäß funktionieren.

7.2 Beschreibung

Schema **elster_elo_200502_extern.xsd** 1

schema location: Z:\xml\schemata\elster_elo_200502_extern.xsd 2
targetNamespace: <http://www.elster.de/2002/XMLSchema> 4

Elements: Complex types

Elster [DatenTeilCType](#) 5
[NutzdatenblockCType](#)
[NutzdatenCType](#)

schema location: <Z:\xml\schemata\allg000007.xsd> 3
targetNamespace: <http://www.elster.de/2002/XMLSchema> 4

Elements: Simple types

Allg [allg AnschriftenZusatzSType](#) 5
[allg AuslandsPLZSType](#)
[allg BundeslandSType](#)
[allg Datum8SType](#)
[allg Datum_JahrSType](#)
[allg DatumSType](#)
[allg ETIMSType](#)
[allg HausnummerSType](#)
[allg KonfessionSType](#)
[allg LaenderschluesseSType](#)
[allg NachNameSType](#)
[allg NamensvorsatzSType](#)
[allg NamenszusatzSType](#)
[allg NutzdatenBlockTicketSType](#)
[allg OrtSType](#)
[allg PLZSType](#)
[allg PostfachSType](#)
[allg SteuerklasseSType](#)
[allg StrSType](#)
[allg TitelSType](#)
[allg TransferTicketSType](#)
[allg TTMM](#)
[allg VersionSType](#)
[allg VornameSType](#)
[allg Zahl4](#)
[BundeslandSType](#)
[DatumSType](#)
[NutzdatenBlockTicketSType](#)
[TransferTicketSType](#)
[VersionSType](#)
[Zahl4](#)

Beschreibung:

Zuerst wird der Name des beschriebenen Schemas dokumentiert [1].

Für dieses [2] und jedes weitere in diesem Schema verknüpfte Schema [3] wird ein Index erstellt. Dieser beschreibt [4] den Namen des Schemas und [5] alle in diesem Schema deklarierten Elemente, Attribute und Gruppen.

Jede Deklaration ist per Hyperlink mit der entsprechenden Definition verbunden, so dass sich sehr einfach zu den einzelnen Definitionen navigieren lässt.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

element Elster			
diagram			
namespace	http://www.elster.de/2002/XMLSchema	10	
children	TransferHeader DatenTeil	11	
identity constraints	unique	Name	Refer
		EindeutigesNutzdatenTicket	
source	<pre> <xs:element name="Elster"> <xs:complexType> <xs:sequence> <xs:element name="TransferHeader" type="elster:TransferHeaderCType"/> <xs:element name="DatenTeil" type="elster:DatenTeilCType"> <xs:annotation> <xs:documentation xml:lang="DE">verschlüsselt</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType> <xs:unique name="EindeutigesNutzdatenTicket"> <xs:selector xpath="elster:DatenTeil/elster:Nutzdatenblock/elster:NutzdatenHeader"/> <xs:field xpath="elster:NutzdatenTicket"/> </xs:unique> </xs:element> </pre>		

Zu allen im Schema definierten Elementen, Attributen, Gruppen, Typendeklarationen gibt es eine Definition in oben angegebener Form.

In den Diagrammen werden XML-Tags **[6]** als Rechtecke dargestellt. Die Symbole **[+]** und **[-]** (bei **[6]** und **[8]**) zeigen an, dass das Tag weitere Kindelemente (Tags) hat.

Die **[7]** beschreibt eine Sequenz, d.h. das Elster-Tag wird zwingend in „TransferHeader“ und „DatenTeil“ unterteilt. Die Reihenfolge (von Oben nach Unten zu lesen) ist dabei zwingend einzuhalten.

[9] der Kommentar zum Tag „DatenTeil“.

Der Namensraum des Elster-Tag **[10]**.

Hyperlinks **[11]** zu den Kindelementen des Elster-Tag.

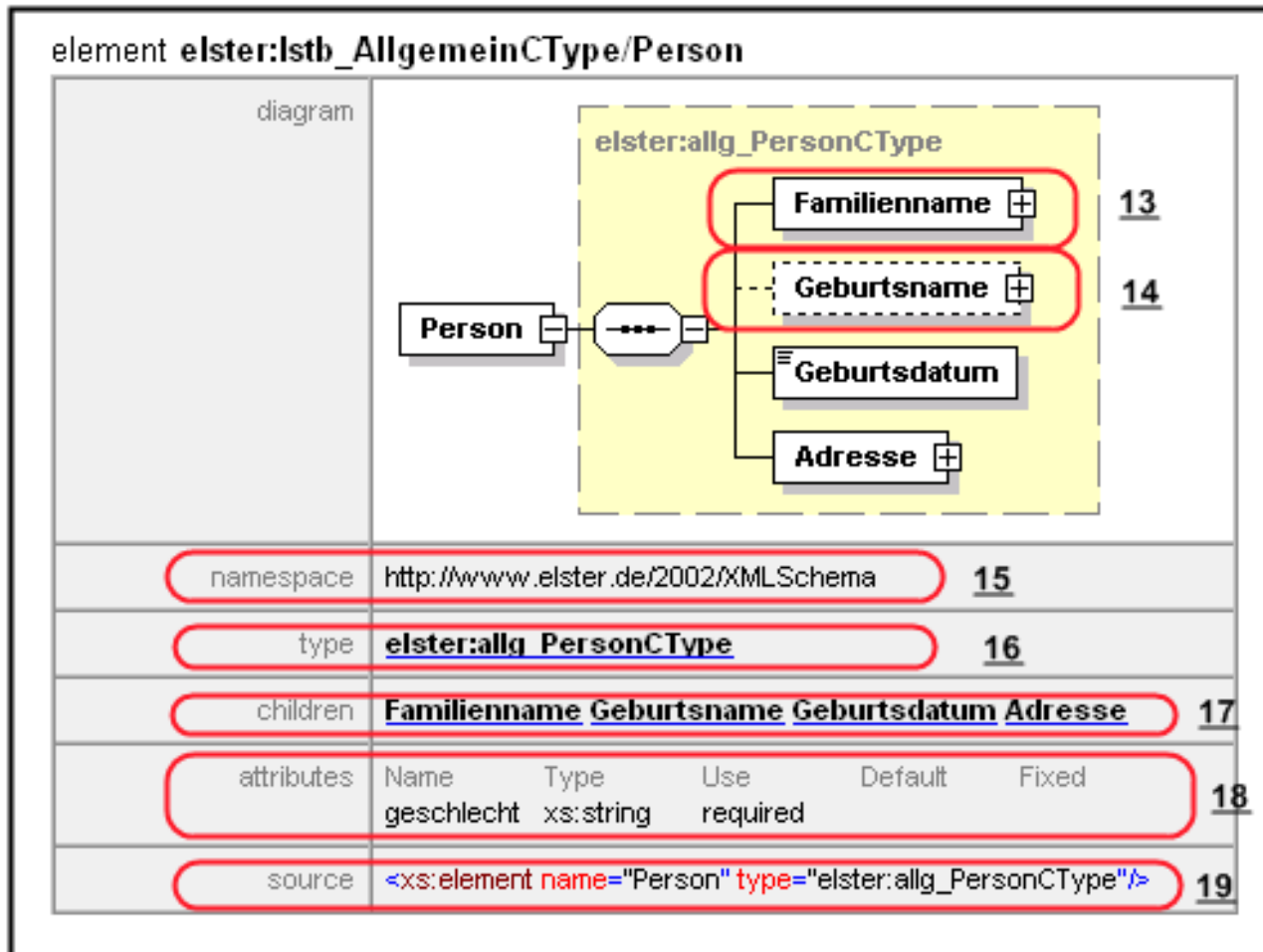
Der Sourcecode **[12]** zum Elster-Tag.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation



Das erste Kindelement von Person [13] unterlegt mit einem Hyperlink zum Familiennamen. Das [+] bedeutet, dass der Familienname selber wiederum eigene Kindelemente besitzt.

Der Geburtsname [14] ist optional, dies wird durch die gestrichelte Linie dokumentiert. Wie jedes Tag in der Grafik ist auch dieses mit einem Hyperlink zu seiner Definition verbunden.

Wichtig: Obwohl der Familienname optional ist, muss der Inhalt auch den in der Dokumentation beschriebenen Vorgaben entsprechen, wenn dieses XML-Tag erstellt wird. Andernfalls ist die Lohnsteuerbescheinigung nicht verarbeitbar und wird fehlerhaft zurückgewiesen. Dass nur eine optionale Angabe nicht den Vorgaben entspricht, ist hierbei unerheblich. Optionale Tags haben meist eine vorgegebene Syntax, so dass diese, wenn sie nicht gefüllt werden, grds. zu einem Fehler führen.

Der Namensraum [15] des Tag „Person“.

Die Definition der Person [16] mittels eines komplexen Typen, dieser ist per Hyperlink verbunden.

Die Kindelemente des Tag Person [17] alle per Hyperlink verbunden.

Die Attribute zum Tag Person [18], **Attribute werden nie im Diagramm angezeigt!**

Der Sourcecode zum Tag Person [19].



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

complexType **elster:allg_PersonCType**

diagram

namespace <http://www.elster.de/2002/XMLSchema>

children [Familienname](#) [Geburtsname](#) [Geburtsdatum](#) [Adresse](#)

used by element [elster:lstb_AllgemeinCType/Person](#)

Name	Type	Use	Default	Fixed	Annotation
geschlecht	xs:string	required			

source

```
<xs:complexType name="allg_PersonCType">
  <xs:sequence>
    <xs:element name="Familienname">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Name" type="elster:allg_NachNameSType"/>
          <xs:element name="Vorname" type="elster:allg_VornameSType"/>
          <xs:element name="Namensvorsatz" type="elster:allg_NamensvorsatzSType" minOccurs="0"/>
          <xs:element name="Namenszusatz" type="elster:allg_NamenszusatzSType" minOccurs="0"/>
          <xs:element name="Titel" type="elster:allg_TitelSType" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Geburtsname" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Name" type="elster:allg_NachNameSType"/>
          <xs:element name="Namensvorsatz" type="elster:allg_NamensvorsatzSType" minOccurs="0"/>
          <xs:element name="Namenszusatz" type="elster:allg_NamenszusatzSType" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Geburtsdatum" type="elster:allg_Datum8SType"/>
    <xs:element name="Adresse" type="elster:allg_AdresseCType"/>
  </xs:sequence>
  <xs:attribute name="geschlecht" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="M"/>
        <xs:enumeration value="W"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

[20] hier endet der Hyperlink zu [16]. Das Diagramm des komplexen Typs Person unterscheidet sich nur dadurch, dass kein Elternelement „Person“ sondern der komplexe Typ „Person“ [21] beschrieben wird. Im Gegensatz zur Definition der Person findet man jedoch bei dem komplexen Typ den kompletten Source zur Person.

Der optionale Geburtsname [22] vgl. auch [14]

Der Source zum optionalen Geburtsnamen [25]. Mittels des Zusatz `minOccurs=0` [24] wird definiert, dass dieser optional ist. Für die Kindelemente muss dies jedoch einzeln wieder deklariert werden. Daraus folgt, wenn ein Geburtsname übermittelt wird, muss dieser mindestens das Kindelement „Name“ beinhalten. Die Kindelemente „Namensvorsatz“ und „Namenszusatz“ sind wiederum optional.

Das zum Tag Person gehörende Attribut [23] mit Kurzbeschreibung.

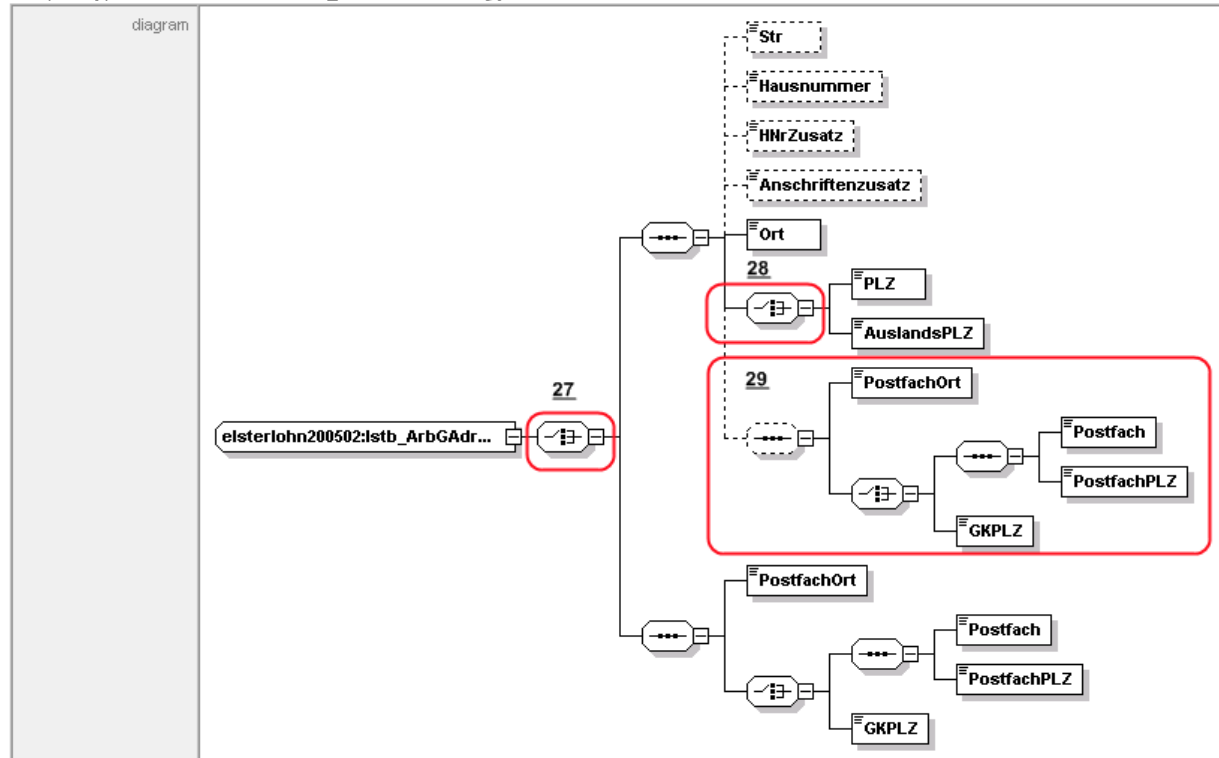
Im Source [26] wird dieses näher beschrieben, insbesondere die Einschränkungen durch die enumerative Aufzählung (hier „M“ und „W“) sind nur aus dem Source ersichtlich.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

complexType **elsterlohn200502:Istb_ArbGAdresseCType**

Die Verzweigung **[28]** steht für eine Wahl; in diesem Beispiel eine Wahl zwischen PLZ und AuslandsPLZ. Hier darf entweder eine inländische Postleitzahl (PLZ) oder eine ausländische Postleitzahl (AuslandsPLZ) übermittelt werden.

Die Wahl kann nicht nur auf Elementebene sondern auch auf Sequenzebene **[27]** erfolgen.

Die **[29]** zeigt ein Beispiel für eine optionale Sequenz mittels der ein PostfachOrt sowie wahlweise eine Sequenz aus Postfach und PostfachPLZ bzw. einer Großkundenpostleitzahl übermittelt wird.



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

element **elsterlohn200502:Istb_BesteuerungsmerkmaleCType/Kirchensteuerabzug**

diagram	Kirchensteuerabzug			
namespace	http://www.elsterlohn.de/2005-02/XMLSchema			
type	elsterlohn200502:Istb_KirchensteuerabzugCType 30			
attributes	Name	Type	Use	Default
	konfession	elster:allg_KonfessionSType	required	
	gueltig_ab	elster:allg_TTMM	required	31
source	<xs:element name="Kirchensteuerabzug" type="elsterlohn:Istb_KirchensteuerabzugCType" maxOccurs="12"/>			

Das Tag Kirchensteuerabzug, das selbst keine weiteren Kindelemente besitzt. Die Definition findet sich in KirchensteuerabzugCType **[30]**, welches per Hyperlink verknüpft ist. Es kann maximal 12 mal übermittelt werden **[31]**.

complexType **elsterlohn200502:Istb_KirchensteuerabzugCType**

diagram	<div>elsterlohn200502:Istb_Kirchenst...</div>					
namespace	http://www.elsterlohn.de/2005-02/XMLSchema					
used by	elements elsterlohn200502:Istb_BesteuerungsmerkmaleCType.Kirchensteuerabzug elsterlohn200502:Istb_BesteuerungsmerkmaleCType.KirchensteuerabzugEheg					
attributes	Name	Type	Use	Default	Fixed	Annotation
	konfession	elster:allg_KonfessionSType	required			
	gueltig_ab	elster:allg_TTMM	required			
source	<div><xs:complexType name="Istb_KirchensteuerabzugCType"> <xs:attribute name="konfession" type="elster:allg_KonfessionSType" use="required"/> <xs:attribute name="gueltig_ab" type="elster:allg_TTMM" use="required"/> </xs:complexType></div>					<div>32</div>

Der komplexe Typ **[32]** der den Aufbau bestimmt. Hier ist ein Sonderfall, da der Kirchensteuerabzug nur Attribute besitzt und selber keinen Wert annimmt. Definition des Attributes „konfession“ **[33]**, leider nicht verlinkt, muss über die Liste am Anfang der HTML-Datei gesucht werden



ElsterLohn

verwendete XML-Techniken

HTML-Dokumentation

simpleType elster:allg_KonfessionSType	
namespace	http://www.elster.de/2002/XMLSchema
type	restriction of xs:string 34
used by	attributes elster:lstb_KirchensteuerabzugCType/@konfession elsterlohn200502:lstb_KirchensteuerabzugCType/@konfession
facets	enumeration ev enumeration lt 35 enumeration rf enumeration fr enumeration ak enumeration is enumeration fb enumeration ib enumeration iw enumeration fg enumeration fm enumeration fs enumeration fa enumeration il enumeration jd enumeration rk enumeration --
annotation	documentation ev = Evangelische Kirchensteuer documentation lt = Evangelisch lutherisch 36 documentation rf = Evangelisch reformiert documentation fr = Französisch reformiert documentation ak = Alt Katholische Kirchensteuer documentation is = Israelitische Kultussteuer Frankfurt documentation fb = Kirchensteuer der Freireligiösen Landesgemeinde Baden documentation ib = Kirchensteuer der Israelitischen Religionsgemeinschaft Baden documentation iw = Kirchensteuer der Israelitischen Religionsgemeinschaft Württembergs documentation fg = Freireligiöse Landesgemeinde Pfalz documentation fm = Freireligiöse Gemeinde Mainz documentation fs = Freireligiöse Gemeinde Offenbach/Mainz documentation fa = Freie Religionsgemeinschaft Alzey documentation il = Israelitische Kultussteuer der Kultusberechtigten Gemeinden documentation jd = Jüdische Kultussteuer documentation rk = Römisch-Katholische Kirchensteuer documentation -- = für Steuerpflichtige mit Wohnsitz im Ausland
source	<pre> <xs:simpleType name="allg_KonfessionSType"> <xs:annotation> <xs:documentation xmt:lang="DE">ev = Evangelische Kirchensteuer</xs:documentation> <xs:documentation xmt:lang="DE">lt = Evangelisch lutherisch</xs:documentation> <xs:documentation xmt:lang="DE">rf = Evangelisch reformiert</xs:documentation> <xs:documentation xmt:lang="DE">fr = Französisch reformiert</xs:documentation> <xs:documentation xmt:lang="DE">ak = Alt Katholische Kirchensteuer</xs:documentation> <xs:documentation xmt:lang="DE">is = Israelitische Kultussteuer Frankfurt</xs:documentation> <xs:documentation xmt:lang="DE">fb = Kirchensteuer der Freireligiösen Landesgemeinde Baden</xs:documentation> <xs:documentation xmt:lang="DE">ib = Kirchensteuer der Israelitischen Religionsgemeinschaft Baden</xs:documentation> <xs:documentation xmt:lang="DE">iw = Kirchensteuer der Israelitischen Religionsgemeinschaft Württembergs</xs:documentation> <xs:documentation xmt:lang="DE">fg = Freireligiöse Landesgemeinde Pfalz</xs:documentation> </xs:annotation> </xs:simpleType> </pre>

Die Basis der Konfession **[34]** ist ein String.

Einschränkungen der Basisdefinition **[35]** (in diesem Beispiel wird der String durch die enumerative Auflistung aller Konfessionskürzel, die auf Lohnsteuerkarten eingetragen werden dürfen, eingeschränkt)

Die im Schema enthaltenen Kommentare zu den Konfessionskürzeln **[36]**, an dieser Stelle werden verwendete Abkürzungen beschrieben und ggf. weitere Erläuterungen gegeben.