eCH-0018: XML Best Practices

Name	XML Best Practices
Standard-Nummer	eCH-0018
Kategorie	Standard
Reifegrad	experimental
Status	Genehmigt
Ausgabedatum	2005-06-21
Gültig seit	2005-08-25
Gültigkeitsdauer	-
Version	1.0
Ersetzt Standard	-
Basiert auf	-
Änderungen	-
Sprachen	Deutsch und Französisch
Herausgeber / Vertrieb	Verein eCH, Laupenstrasse 18a, 3008 Bern 031 560 00 20 / info@ech.ch / http://www.ech.ch/
Autoren	Fachgruppe XML-Standards: Hanspeter Salvisberg (hanspeter.salvisberg@unisys.com) (bis Version 0.07) Alexander Pina (alexander.pina@unisys.com) (bis Version 0.07) Erik Wilde (net.dret@dret.net)

Zusammenfassung

Das vorliegende Dokument beschreibt Regeln, welche bei der Benutzung von XML und von XML Schemas in eCH Standards zu berücksichtigen sind. Dabei wird das Schwergewicht auf Basismechanismen und Grundsatzüberlegungen gestellt, welche sich die Benutzern von XML Schemas in der Regel stellen.

Inhaltsverzeichnis

1 Status des Dokuments

Das vorliegende Dokument wurde vom Expertenausschuss am 25. August 2005 ***genehmigt.*** Es hat für das definierte Einsatzgebiet im festgelegten Gültigkeitsbereich normative Kraft.

2 Anwendungsbereich

Der vorliegende Standard definiert Richtlinien für die Verwendung und Erstellung von XML Dokumenten und für die Verwendung von XML Schemas in eCH Standards. Da auch XML Schemas XML Dokumente sind, werden in einem ersten Teil (Teil I) allgemeine Richtlinien zu XML verfasst, während in den darauffolgenden Teilen spezielle Richtlinien zu XML Schemas (Teil II) und zu XML Dokumenten als Instanzen von XML Schemas (Teil III) erläutert werden.

2.1 XML Terminologie

In der Literatur werden die Begriffe "XML Dokument" und "XML Instanz" oftmals synonym verwendet. Der XML Standard spricht konsequent von "XML Dokumenten", zudem legt der Begriff der "Instanz" die Existenz einer Klassenbeschreibung (im Fall von XML kann das z.B. eine DTD oder ein XML Schema sein) nahe, aber das Konzept des well-formed XML erlaubt auch Dokumente ohne Klassenbeschreibung. Aus diesem Grund wird hier konsequent der Begriff "XML Dokument" verwendet.

Das "Document Element" eines XML Dokumentes ist das Element, das alle anderen Elemente enthält. Häufig wird hierfür der Begriff "Root Element" verwendet, dieser ist jedoch nicht im Standard definiert und wird hier daher vermieden.

Im Zusammenhang mit XML Schema ist zu beachten, dass ein "XML Schema" eine logische Menge an Komponenten ist, die in einem oder mehreren "Schema Dokumenten" definiert sein können. Ein Schema Dokument ist ein XML Dokument, das die XML Elemente des XML Schema Namespaces verwendet, um Schema Komponenten zu definieren. Ein Schema Dokument kann andere Schema Dokumente mittels `xs:include` und `xs:redefine` einbinden, und das gesamte XML Schema besteht dann aus der Auflösung aller solcher Referenzen zwischen "Schema Dokumenten" zu einem logischen "XML Schema". Mit `xs:import` werden Referenzen auf (globale) Komponenten in anderen Schemas ermöglicht (die referenzierten Komponenten werden in diesem Fall aber nicht Teil des referenzierenden Schemas.)

2.2 Terminologie der Empfehlungen

Richtlinien in diesem Dokument werden gemäss der Terminologie aus [RFC2119] angegeben, dabei kommen die folgenden Ausdrücke zur Anwendung, die durch GROSSSCHREIBUNG als Wörter mit den folgenden Bedeutungen kenntlich gemacht werden (Zitat aus RFC 2119):

- **MUST**: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

- **MUST NOT**: This phrase, or the phrase "SHALL NOT", mean that that definition is an absolute prohibition of the specification.
- **SHOULD**: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT**: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- **MAY**: This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

2.3 Weiterführende eCH-Dokumente

Neben dem vorliegenden Dokument beschäftigen weiterführende Dokumente mit verschiedenen Aspekten von XML-Technologien, die vom vorliegenden Dokumente unbehandelte Fragen diskutieren. Das vorliegende Dokument enthält keinerlei normative Referenzen auf die weiterführenden Dokumente, die sich zum Zeitpunkt der Veröffentlichung von eCH-0018 noch in Entwicklung befinden, aber nach ihrer Veröffentlichung als begleitende Lektüre zu eCH-0018 dienen können.

- Beschreibung von XML Namespaces [eCH-0033]
- Design von XML Schemas [eCH-0035]
- Modellierung für den XML-orientierten Datenaustausch [eCH-0036]

Teil I: XML Dokumente

3 XML Dokumente als Instanzen von Klassen

XML Dokumente können im Prinzip als well-formed Dokumente keinen anderen Regeln als denen des XML-Standards gehorchen, werden aber in aller Regel darüberhinaus Instanzen einer Klasse von Dokumenten sein. Die Klassen von Dokumenten werden häufig von Schemasprachen definiert, die durch Regeln oder andere Definitionen festlegen, welche Bedingungen eine Instanz dieser Klasse erfüllen muss.

3.1 Problemstellung

XML selber definiert eine eigene Schemasprache, die *Document Type Definition (DTD)*. DTDs haben eine spezielle Stellung, weil sie integraler Bestandteil des XML-Standards sind, und deshalb von jeder XML-Software unterstützt werden müssen. Auf der anderen Seite haben DTDs deutliche Schwächen in der Leistungsfähigkeit, vor allem im Bereich der Datentypen und Modellierungsmechanismen. Aus diesem Grund hat das W3C mit *XML Schema* [XMLSchema1,XMLSchema2] eine alternative Schemasprache entwickelt, die leistungsfähiger ist als DTDs. Durch seine Verwendung in Standards wie WSDL (Schnittstellenbeschreibungen für Web Services) und XQuery (Abfragesprache für XML Datenbanken) hat sich XML Schema als neuer de-facto-Standard für XML Schemabeschreibungen entwickelt. Doch auch XML Schema hat gewissen Schwächen, die in speziellen Anwendungsbereichen von alternativen Schemasprachen wie Schematron [ISO19757-3] oder RELAX NG [RELAXNG] besser abgedeckt werden.

Für den Entwickler eines XML Vokabulars stellt sich daher die Frage, welche der Schemasprachen er auswählt, um einerseits ein möglichst gutes Schema schreiben zu können, andererseits aber auch eine Schemasprache zu verwenden, die von anderen Benutzern des Schemas verstanden und mit Softwarekomponenten unterstützt werden kann.

3.2 Empfehlungen

- **SHOULD**: Klassen von XML Dokumenten sollten durch ein Schema beschrieben werden, das es Benutzern ermöglicht, die Regeln zu verstehen, nach dem Dokumente dieser Klasse aufgebaut sein müssen.
- **SHOULD**: Klassen von XML Dokumenten sollten durch XML Schema beschrieben werden, das in den Bereichen der Datentypen und der Modellierung deutlich leistungsfähiger ist als DTDs.
- **SHOULD NOT**: Klassen von XML Dokumenten sollten nicht durch DTDs beschrieben werden, da DTDs eine weniger präzise Definition erlauben als XML Schema.
- **MAY**: Falls durch den Anwendungsfall gerechtfertigt, können ergänzende oder alternative Schemas in anderen Schemasprachen (z.B. Schematron oder RELAX NG) verwendet werden, die zwar weniger verbreitet sind als XML Schema, dafür aber andere Features bieten.

- **SHOULD**: Bei der Verarbeitung von XML Dokumenten sollten an kritischen Stellen im Workflow Validierungen vorgenommen werden, um korrektes XML sicherzustellen.

4 XML Versionen

XML existiert in zwei Versionen, XML 1.0 [XML10Third] als der ersten festgelegten Version, und XML 1.1 [XML11] als einer überarbeiteten Version. Die Unterschiede zwischen den beiden Versionen sind recht gering (http://www.w3.org/TR/xml11/#sec-xml11), es ist dabei zu beachten, dass bei der Verwendung von XML 1.1 automatisch auch XML Namespaces 1.1 [XMLNS11] verwendet werden, da der XML Namespace Mechanismus (siehe Abschnitt 8) keine Methode hat, eine Version (abgesehen von der XML-Version) anzugeben.

4.1 Problemstellung

XML 1.1 nimmt nur minimale Änderungen an XML 1.0 vor, kann jedoch nur von Software verarbeitet werden, die XML 1.1 unterstützt. Viele Software (Parser und andere XML-verarbeitende Programme) unterstützen kein XML 1.1, und abgesehen von den Features, die nun auch in XML die Zeilenende-Konventionen von Grossrechnern (http://www.w3.org/TR/xml11/#sec-line-ends) zulassen, ist XML 1.1 in den seltensten Fällen notwendig.

Viele XML-Software prüft nicht die Version des verarbeiteten XML und akzeptiert daher (oftmals fälschlicherweise) auch XML 1.1 als Eingabe. Ist die Software jedoch nicht explizit mit XML 1.1 kompatibel, so kann die Verwendung von XML 1.1 spezifischen Features in XML Dokumenten zu unstabilem Verhalten führen.

4.2 Empfehlungen

- **SHOULD NOT**: XML Dokumente sollten kein XML 1.1 verwenden, einzig wenn XML 1.1 Features unabdingbar notwendig und die Verwendung von XML 1.1 mit allen Beteiligten abgestimmt worden ist, kann XML 1.1 verwendet werden.
- **MUST**: Wird XML 1.1 verwendet, so muss dies in der Dokumentation deutlich gekennzeichnet sein und es muss darauf hingewiesen werden, dass die betreffenden Dokumente nur mit XML 1.1 fähiger Software verarbeitet werden dürfen. Den Dokumenten selbst sieht man die XML Version 1.1 in jedem Fall an, da die XML Deklaration (in der die Versionsangabe enthalten ist) in XML 1.1 verpflichtend angegeben werden muss (http://www.w3.org/TR/xml11/#NT-XMLDecl).

5 Generelle Form von XML Dokumenten

XML ist ein zeichenorientiertes Format, d.h. es ist auf der Grundlage von Zeichen definiert und nicht als binäres Format. Aus diesem Grunde sind im Zusammenhang mit der Verwendung von XML die klassischen Fragen zeichenorientierter Formate zu klären, insbesondere der verwendete Zeichenvorrat (Abschnitt 5.1), und der Umgang mit Zeilenenden (Abschnitt 5.3). Generelle

Begriffsdefinition und weitergehende Betrachtungen zur Verwendung von Zeichen und Zeichensätzen in einer offenen Umgebung finden sich in [WebChar10].

5.1 All-in-One Prinzip

XML definiert nicht nur ein Datenformat, sondern verschiedene Standards (XML selber und XML Schema ebenfalls) definieren ebenso ein Verarbeitungsmodell, das standardkonforme Software einhalten muss. Aus diesem Grund ist es möglich, ausgehend von diesem Verarbeitungsmodell XML so zu verwenden, dass es noch vor der applikationsspezifischen Verarbeitung nicht-triviale Verarbeitung verlangt, so z.B. das Zusammensammeln verschiedener Teile eines Dokuments aus verschiedenen Quellen.

5.1.1 Problemstellung

DTDs (mit external Entities und Attribut-Defaults) und XML Schema (mit Attribut-Defaults) erlauben Dokumente und/oder Schemas zu definieren, deren Interpretation von externen Ressourcen und/oder dem Schema abhängt. Dies macht die Verarbeitung von Dokumenten solcher Klassen aufwendig und fehleranfällig.

5.1.2 Empfehlungen

- **SHOULD**: Dokumentenklassen und Dokumente sollten so definiert und verwendet werden, dass alle zur Verarbeitung eines Dokuments notwendige Information in diesem Dokument enthalten ist.
- **MUST NOT**: Default-Werte von Attributen dürfen im Schema nicht definiert werden, ausser wenn es sich um durch das Schema definierte Konstanten handelt, deren Existenz bei der Verarbeitung vorausgesetzt werden kann.

Kommentar: Dies betrifft nicht logische Abhängigkeiten auf der Applikationsebene, die natürlich über geeignete Mechanismen repräsentiert werden können. Es geht in dieser Empfehlung einzig darum, keine XML oder Schema-spezifischen Mechanismen zu verwenden, die die Information auf verschiedene Ressourcen aufteilen.

5.2 Character Encodings

XML verwendet als Zeichensatz (das Character Repertoire) immer Unicode oder eine Untermenge davon (z.B. ASCII oder ISO-8859-X, also eine der ISO-8859 Codierungen). Das spezifische Character Encoding hingegen (also die Frage, wie ein Zeichen des Zeichensatzes codiert wird), kann unterschiedlich sein.

5.2.1 Problemstellung

Durch Verwendung von unterschiedlichen Character Encodings kann es zu Inkompatibilitäten bezüglich Character Encodings kommen. Die entwickelten Schemas und XML Dokumente sollten mit einer zukunftssicheren Codierung weltweit einsetzbar sein.

In der XML-Spezifikation http://www.w3.org/TR/REC-xml/#charencoding ist festgehalten, dass die beiden Codierungen UTF-8 und UTF-16 als 'MUST' definiert sind, wobei Dokumente mit UTF-16 als 'MUST' und mit UTF-8 als 'MAY' mit einer „Byte Order Mark (BOM)“ zu kennzeichnen sind.

Dies hat zur Folge, dass ASCII Dokumente automatisch unproblematisch sind, weil ASCII Dokumente nichts anderes als UTF-8 ohne BOM sind (was erlaubt ist, s.o.), die nur die Zeichen U+0000 bis U+007F benutzen.

Legacy Systeme, die keine UTF-8 Codierung unterstützen, müssen eine andere Form der Codierung wählen. Die Unterstützung der Codierung ist in diesem Fall abhängig von der verwendeten XML-Software.

5.2.2 Empfehlungen

- **SHOULD**: Als Codierung sollte UTF-8 verwendet werden.
- **MUST**: Die „encoding“ Deklaration in der XML-Deklaration muss immer angegeben werden.
- **MUST**: Ist die verwendete Codierung nicht US-ASCII oder UTF-8, so muss die „encoding“ Deklaration in der XML-Deklaration angegeben werden.

Die Art der Codierung der XML Dokumente muss in der XML-Deklaration in jedem XML Dokument angegeben werden.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Im Falle von Legacy Systemen, welche UTF-8 nicht unterstützen, sind nach Absprache aller beteiligten Partner die von allen Systemen unterstützten Codierungen zu benutzen, diese müssen in der XML-Deklaration angegeben werden.

Beispiele:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<?xml version="1.0" encoding="ISO-2022-JP"?>
```

5.3 Zeilenumbrüche

XML basiert auf einer zeichenbasierten Syntax (Markup), in der die Struktur von Daten durch Elemente und Attribute bestimmt wird. Viele XML-Tools (z.B. XML-Editoren) erzeugen von sich aus viele Zeilenumbrüche zwischen Elementen, die kein relevanter Teil der Daten sind, sondern lediglich die Menschenlesbarkeit des XML vergrössern (http://www.w3.org/TR/REC-xml/#sec-white-space).

5.3.1 Problemstellung

Aus Applikationssicht sind so genannte Whitespace Text Nodes (Text-Knoten, die nur aus Whitespace bestehen, also Space, TAB, NL oder CR) fast nie relevant, und Applikationen sollten so programmiert werden, dass diese Unterschiede keine Auswirkungen auf die Interpretation haben.

ACHTUNG: Whitespace Text Nodes in XML Dokumenten sind signifikant, falls im Dokument durch das `xml:space="preserve"` Attribut gekennzeichnet, oder falls das Schema für das Element Mixed Content erlaubt (in DTDs mit `(#PCDATA | ...)*` markiert, in XML Schema mit `mixed="true"`). Auf diese signifikanten Whitespace Text Nodes bezieht sich die folgende Empfehlung nicht!

5.3.2 Empfehlungen

- **SHOULD**: XML sollte so formatiert werden, dass es gut menschenlesbar ist (Zeilenumbrüche und Einrückungen), soweit dies aus Platzgründen vertretbar ist.
- **MUST**: Applikationen dürfen nicht davon ausgehen. dass der Whitespace in XML-Dokumenten in einer bestimmten Art formatiert ist.

5.4 XML und binäre Daten

XML ist hauptsächlich für textuelle Inhalte geeignet und erlaube keine direkte Integration binärer Daten. Durch eine Transformation binärer Daten in eine Textrepräsentation können diese zwar in ein XML-Dokument integriert werden, diese Art der Repräsentation ist jedoch ineffizient und nur für kleine Datenmengen geeignet

5.4.1 Problemstellung

Sollen XML-Daten mit binären Daten kombiniert werden, so sollten nur kleine Mengen an binären Daten in das XML eingebettet werden, grössere Mengen sollten separat codiert und per Verweis integriert werden. Für diese Art von Daten definiert XML Schema die Datentypen `xs:hexBinary` (hexadezimale Ziffern als Zeichen) und `xs:base64Binary` (binäre Daten dargestellt als Buchstaben gemäss [RFC2045]). Die `xs:base64Binary` Darstellung ist kompakter als die `xs:hexBinary` Darstellung (`xs:base64Binary` vergrössert das Datenvolumen um ca. 33%, `xs:hexBinary` dagegen um 100%).

5.4.2 Empfehlungen

- **SHOULD**: Grössere Mengen an binären Daten sollten nicht in XML Dokumente eingebettet werden, sondern separat codiert und per Verweis referenziert werden. Der Verweis kann anwendungsspezifisch sein, oder sich eines allgemeinen Standards wie [XLink] bedienen.
- **SHOULD**: Kleinere Mengen an binären Daten können in XML Dokumente eingebettet werden. Als Codierung sollte XML Schemas `xs:base64Binary` Datentyp verwendet werden. Für die Verwendung und insbesondere die Beachtung der Behandlung von Whitespace in XML Schema Base64 sollte XML Schema konsultiert werden: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/#base64Binary

5.5 Versionierung

XML Anwendungen haben eine lange Lebensdauer und während dieser Lebensdauer können verschiedene Versionen von Schemas definiert werden und koexistieren. Um die Identifikation der Schemaversion zu ermöglichen, müssen Schemas versioniert werden.

5.5.1 Problemstellung

Die Problemstellung der Versionierung von Schemas und der Dokumentation verschiedener Versionen wird auf der Ebene der Schemadefinition in Abschnitt 10 und auf der Ebene der Schemabeschreibung in [eCH-0033] behandelt. Auf der Ebene des Dokuments muss sichergestellt werden, dass ein Dokument die Information enthält, die es als Dokument einer bestimmten Version identifiziert.

5.5.2 Empfehlungen

- **MUST**: XML Dokumente müssen eine Versionsangabe tragen, falls sie Instanzen eines Schemas sind. Die Versionsangabe identifiziert die Version des Schemas.

6 Entities

In XML gibt es das Konzept von Entities, unter denen man sich generell ein Stück referenzierbaren Inhalts vorstellen kann. Entities werden deklariert (http://www.w3.org/TR/REC-xml/#sec-entity-decl, durch DTD Konstrukte) und referenziert (http://www.w3.org/TR/REC-xml/#sec-references, mittels ihres Namens und dem Markup `&Name;`). Je nachdem, ob das referenzierte Stück Inhalt intern oder extern ist, gibt es internal (Abschnitt 6.1) oder external (Abschnitt 6.2) Entities. Character References (Abschnitt 6.3) sind technisch gesehen keine Entity Referenzen, sind aber syntaktisch und von der Funktion her sehr ähnlich und werden daher ebenso im vorliegenden Abschnitt behandelt.

Das generelle Problem mit Entities ist, dass sie von XML Schema nicht unterstützt werden, d.h. XML Schema hat keine Sprachmittel zur Deklaration von Entities.

6.1 Internal Entities

Das aus HTML bekannte Konstrukt `&uuml;` für ein "ü" ist nichts anderes als eine Referenz auf ein in der HTML-DTD definiertes Entity. Da die Entity-Deklaration direkt den so genannten Replacement-Text enthält, handelt es sich um ein Internal Entity (http://www.w3.org/TR/REC-xml/#sec-internal-ent). Im Fall von XML Schema Dokumenten können keine Internal Entities verwendet werden, es sei denn, sie werden im "Internal Subset" (http://www.w3.org/TR/REC-xml/#NT-intSubset) eines XML Dokumentes deklariert.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE text [
<!ENTITY uuml "&#xFC;" >
```

```
]>
<text>Internal Entity u Umlaut: &uuml;
Character Reference u Umlaut: &#xFC;
Unicode Character u Umlaut: ü
</text>
```

6.1.1 Empfehlungen

- **SHOULD NOT**: XML Dokumente sollten keine Entity-Deklarationen im Internal Subset enthalten. Sind XML Dokumente Instanzen einer DTD, so sind Referenzen auf in der DTD definierte Internal Entities unproblematisch, im Fall von XML Schema sollte auf die Verwendung von im Internal Subset definierten Internal Entities verzichtet werden.
- **SHOULD**: Falls Entities verwendet werden, sollten soweit irgend möglich die in XHTML 1.0 [XHTML10Sec] definierten Entities (http://www.w3.org/TR/xhtml1/#h-A2) verwendet werden, weil dadurch auf etablierte Standardbezeichnungen für Sonderzeichen zurückgegriffen wird.

Die 5 in XML vordefinierten Entities für Markup-Zeichen sind von diesen Empfehlungen ausgenommen, da sie per Definition von jedem XML Prozessor erkannt werden müssen. Es handelt sich dabei um die Entities `amp` (Ampersand: &), `lt` (Less Than: <), `gt` (Greater Than: >), `apos` (Apostrophe: '), und `quot` (Quote: ").

6.2 External Entities

External Entities sind der Mechanismus, mit dem man in XML (mit Hilfe von DTD-Mechanismen) Include-ähnliche Funktionen implementiert. Im Gegensatz zu einem herkömmlichen Include muss man bei External Entities zunächst das Entity deklarieren (http://www.w3.org/TR/REC-xml/#sec-external-ent, mit der URI der externen Ressource) und anschliessend referenzieren. Die Referenzierung des External Entities ist dann das "Include-Statement", das das Einfügen der externen Ressource bewirkt.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE text [
<!ENTITY xx SYSTEM "includes/xx.xml" >
]>
<text>Einfügen des Inhalts von Dokument xx.xml: &xx;</text>
```

6.2.1 Empfehlungen

- **SHOULD NOT**: XML Dokumente sollten keine Entity-Deklarationen im Internal Subset enthalten. Sind XML Dokumente Instanzen einer DTD, so sind Referenzen auf in der DTD definierte External Entities unproblematisch, im Fall von XML Schema sollte auf die Verwendung von im Internal Subset definierte External Entities verzichtet werden.
- **SHOULD**: Sind Verweise auf externe Ressourcen notwendig, so sollte dafür XInclude [XInclude] verwendet werden.

6.3 Character References

Character References (http://www.w3.org/TR/REC-xml/#NT-CharRef) sehen Entity Referenzen ähnlich, referenzieren aber kein deklariertes Entity, sondern ein durch seinen Code Point identifiziertes Zeichen aus dem Unicode Zeichensatz. Im Prinzip kann man jedes Zeichen (in Element-Inhalten und Attributwerten) in einem XML Dokument durch eine entsprechende Character Reference ersetzen, dies verringert jedoch die Lesbarkeit des Dokuments und vergrössert das Dokument erheblich. Aus diesem Grunde werden oftmals nur Zeichen als Character Reference angegeben, die sich nicht über die Tastatur eingeben lassen, oder die sich mit dem für das XML Dokument verwendeten Character Encoding nicht codieren lassen (z.B. kann ein ü nicht mit US-ASCII codiert werden, aber über die Character Reference `&#xFC;` dennoch in einem US-ASCII codierten XML Dokument verwendet werden).

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE text [
<!ENTITY uuml "&#xFC;" >
]>
<text>Internal Entity u Umlaut: &uuml;
Character Reference u Umlaut: &#xFC;
Unicode Character u Umlaut: ü
</text>
```

6.3.1 Empfehlungen

- **SHOULD**: Zeichen, die im gewählten Character Encoding direkt verwendet werden können, sollten nicht als Character References verwendet werden.
- **SHOULD**: Character References sollten soweit wie möglich vermieden werden. Dies ist am einfachsten und in den meisten Fällen möglich durch eine geeignete Wahl des Character Encodings (siehe Abschnitt 5.1) des XML Dokuments.
- **MAY**: Sollte das Character Encoding des XML Dokuments gewünschte Unicode Characters nicht codieren können, oder sollten diese Zeichen nicht direkt eingegeben werden können, so können die entsprechenden Zeichen als Character References im Dokument aufgeführt werden.
- **SHOULD**: Bei der Verwendung von Character References ist die hexadezimale Schreibweise (`&#x263A;`) der dezimalen Schreibweise (`&#9786;`) vorzuziehen.

7 Namen von Elementen und Attributen

Namen kommen in XML Dokumenten für Elemente und Attribute (und Processing Instruction Targets, auf die nicht näher eingegangen wird) vor. Die Festlegungen im vorliegenden Abschnitt beziehen sich auf die in XML Dokumenten sichtbaren Namen (Elemente und Attribute). Abschnitt 10 behandelt als Ergänzung dazu die Namen in XML Schemas (alle benannten Komponenten neben Elementen und Attributen, also Notationen, Typen, Identity Constraints und Named Groups).

7.1 Die Sprache der Namen

7.1.1 Problemstellung

Wenn die Schemas mit lokalisierten Namen, also auf der Grundlage lokaler Sprachen, entwickelt werden, besteht die Gefahr, dass die Schemas nur in dieser spezifischen Sprachregion akzeptiert werden. Um der Mehrsprachigkeit der Schweiz, aber auch internationalen Anforderungen, gerecht zu werden, muss eine weltweit gültige Lösung angestrebt werden. Zudem wird dadurch die Konsistenz bei der Kombination verschiedener Schemas erhöht, die durch das Auftreten verschiedensprachiger Namen im gleichen XML Dokument stark leiden würde.

7.1.2 Empfehlungen

- **SHOULD**: Die Sprache der Namen ist Englisch.
- **MAY**: Falls im Schema Namen verwendet werden, die eine sprachspezifische Bedeutung haben, die sich nicht übersetzen lässt (z.B. juristische Fachwörter), so können diese in der Sprache geschrieben werden, für die die Bedeutung erhalten bleiben muss.
- **SHOULD NOT**: Die obige Ausnahmeregelung sollte nicht dafür verwendet werden, um Schemas dreisprachig parallel zu führen.
- **MUST**: Als Zeichen, die in Namen verwendet werden dürfen, sind ausschliesslich die ASCII Gross- und Kleinbuchstaben (a-z und A-Z), die Ziffern 0-9, und die Zeichen Underscore (_), Punkt (`.`), und Bindestrich (`-`) zulässig.

Es wird angenommen, dass die Personen, welche mit den Schemas arbeiten, der englischen Sprache mächtig sind. Englische Namen in XML Dokumenten haben sich mit wenigen Ausnahmen in praktisch allen Anwendungsbereichen etabliert.

Die Empfehlungen in diesem Abschnitt sind im Zusammenhang mit Abschnitt 9.2 zu lesen, der sich mit der Sprache von Aufzählungswerten beschäftigt und in Abschnitt 9.2.2 die gleichen Empfehlungen formuliert wie die hier aufgeführten Empfehlungen für die Sprache der Namen von Elementen und Attributen.

7.2 Namenskonventionen

7.2.1 Problemstellung

Um die Lesbarkeit von XML Dokumenten zu erhöhen und eine allgemein gültige Grundlage für alle XML-Entwickler zu schaffen, ist eine Standardisierung der Struktur der verwendeten Namen (und nicht nur ihrer Sprache) extrem wichtig.

Für die Anwender eines Schemas ist es um vieles einfacher, wenn das Vokabular selbsterklärende und intuitive Namen verwendet, Abkürzungen vermieden werden, und Konsistenz hinsichtlich der Verwendung der Namenskonventionen besteht.
XML ist case-sensitive, d.h. jede Auswahl an Klein- und Grossbuchstaben ist relevant und muss in genau dieser Form überall verwendet werden, und aus diesem Grund werden die folgenden Empfehlungen gegeben.

7.2.2 Empfehlungen

7.2.2.1 Notation der Namen

- **SHOULD**: Alle Namen (mit Ausnahme von Akronymen, siehe Abschnitt 7.2.2.2) werden in Kleinbuchstaben geschrieben.

Beispiel:
```
<name>Klaus Huber</name>
<city>Zürich</city>
```

- **MUST NOT**: Doppelpunkte (**:**) dürfen nicht in Namen verwendet werden, dies würde zu Konflikten mit XML Namespaces (Abschnitt 8) führen und die Dokumente mit praktisch allen XML-Technologien unverarbeitbar machen.
- **SHOULD NOT**: Unterstriche (_), Bindestriche (-) und Punkte (**.**) sollten nicht in Namen verwendet werden.

Beispiel:
Person**.**Name → falsch
Binary_Field → falsch

- **SHOULD**: Namen aus zusammengesetzten Begriffen sollten soweit möglich vermieden werden.
- **MAY**: Sind Namen aus zusammengesetzten Begriffen unumgänglich, so werden sie mit CamelCase Notation (mit kleinem Anfangsbuchstaben) verwendet.

Beispiel:
```
<recordNumber>12345678</recordNumber>
```

- **SHOULD**: Unabhängig von der in einem Schema gewählten Konvention hinsichtlich der Struktur der Namen und der für die Namen verwendeten Zeichen ist darauf zu achten, dass diese Konvention innerhalb des Schemas konsistent eingehalten wird.

7.2.2.2 Abkürzungen (Akronyme)

Akronyme kommen in verschiedener Form vor, meistens komplett aus Grossbuchstaben (XML), aber teilweise auch als Mischung (WebDAV), manchmal auch mit kleinem Buchstaben beginnend (xNAL). Um die Abkürzungen wieder erkennbar zu machen, sollte nichts an der etablierten Schreibweise verändert werden, falls sie verwendet werden sollen.

- **SHOULD**: Grundsätzlich sollten keine Abkürzungen verwendet werden. Auch wenn Abkürzungen in einer Wissensdomäne für allgemein bekannt gehalten werden, heisst das nicht, dass sie von allen möglichen Anwendern verstanden werden.
- **MAY**: Gängige Abkürzungen können verwendet werden, müssen aber allgemein bekannt sein.
- **MUST**: Die Bedeutung der Abkürzungen muss im Schema explizit beschrieben und dokumentiert werden.

Beispiel:

- EAN (European Article Numbering) EAN-Artikelnummer
- ISBN (International Standard Book Number) ISBN-Nummer eines Buches

7.3 Einheitliche Verwendung von Namen

Element- und Attributnamen in XML sind im Prinzip immer kontextabhängig. In DTDs werden Elemente global definiert, so dass ein Elementname immer ein Element der gleich Deklaration bezeichnet. Attribute werden in DTDs dagegen lokal deklariert (in der `ATTLIST` des Elements), so dass der gleiche Namen mehrfach verwendet werden kann. DTDs kennen keinen Mechanismus für global deklarierte Attribute, so dass die konsistente Verwendung des gleichen Attributnamens der Disziplin des DTD-Designers überlassen bleibt.

In XML Schema ist die Situation komplizierter, da Element- und Attributdeklarationen sowohl lokal als auch global sein können. Aus diesem Grund ist es in XML Schema möglich, Elemente und/oder Attribute zu definieren, die den gleichen Namen haben, aber an verschiedenen Stellen in einem Dokument vorkommen können und komplett unterschiedlich definiert sind.

7.3.1 Problemstellung

Für Benutzer eines Schemas ist es verwirrend, wenn Elemente und/oder Attribute den gleichen Namen haben, aber unterschiedlich verwendet werden müssen (weil es in verschiedenen Kontexten verschiedene lokale Definitionen gibt), und/oder eine unterschiedliche Bedeutung haben (weil sich die Semantik in Abhängigkeit vom Kontext ändert). Ein Element- oder Attributnamen sollte eindeutig hinsichtlich seines Typen und seiner Semantik sein, um Missverständnisse und Schwierigkeiten in der Verwendung eines Schemas zu vermeiden.

7.3.2 Empfehlungen

- **SHOULD**: Elemente und Attribute mit dem gleichen Namen sollen den gleichen Typen haben, d.h. die Verwendung von Elementen und Attributen mit dem gleichen Namen, die an verschiedenen Stellen eines Dokuments erlaubt sind, soll überall die gleiche sein. Ändert sich der Typ eines Elements oder Attributes zwischen verschiedenen Kontexten, so sollte es auch jeweils einen neuen Namen erhalten. (Dieses Problem kann nur bei lokal definierten Elementen und Attributen auftreten, global definierte Elemente und Attribute haben in jedem Fall den gleichen Typ.)

- **SHOULD**: Elemente und Attribute mit dem gleichen Namen sollen die gleiche Semantik haben, d.h. ihre Bedeutung soll sich nicht ändern abhängig vom Kontext, in dem sie auftreten. Die Semantik eines Elements oder Attributes ist unabhängig von seiner Syntax, aus diesem Grund ist eine gleich bleibende Syntax (d.h. ein gleichbleibender Typ, so wie im vorangehenden Abschnitt gefordert) noch keine Garantie für eine gleichbleibende Semantik.

8 XML Namespaces

Die Verwendung und vor allem die Beschreibung und Verwaltung von XML Namespaces im eCH Umfeld wird in einem anderen eCH Dokument [eCH-0033] beschrieben. XML Namespaces [XMLNS] werden verwendet, um Namen eines Vokabulars (z.B. durch eine DTD oder ein XML Schema definiert) eindeutig zu benennen, der Namespace Name (eine URI [RFC3986]) und der lokale Name ergeben zusammen einen so genannten "Qualified Name" (QName), der die Namen weltweit eindeutig macht. Namespaces werden durch die Verbindung des Namespace Namens (Abschnitt 8.1) mit einem Präfix deklariert (Abschnitt 8.1), und in QNames durch die Verwendung eines deklarierten Präfixes (Abschnitt 8.2) referenziert.

8.1 Namespace Deklarationen

Namespace Deklarationen treten in einem XML Dokument als Attribute auf, die mit der Buchstabenfolge `xmlns` beginnen. Namespaces können in einem beliebigen Element deklariert werden und sind dann deklariert für dieses Element und alle Nachfolger (direkte oder indirekte Kinder des Elementes).

ACHTUNG: Der Default Namespace (deklariert mit `xmlns="…"`) gilt nicht für Attribute, d.h. Attribute ohne Präfix sind nie einem Namespace zugeordnet. Sollen Attribute aus einem Namespace referenziert werden, so muss dies explizit mittels eines Präfix erfolgen.

8.1.1 Problemstellung

Namespaces können irgendwo in einem XML Dokument deklariert werden, was es schwierig machen kann, ausgehend von einem Element einfach alle dort gültigen Namespaces zu erkennen. Es sollte daher von dieser grossen Flexibilität von Namespace-Deklarationen kein Gebrauch gemacht werden, um die Übersichtlichkeit von XML Dokumenten mit Namespaces zu vergrössern.

8.1.2 Empfehlungen

- **SHOULD**: Namespaces sollten nur im Document Element deklariert werden.
- **SHOULD**: Namespaces sollten nicht umdeklariert werden, d.h. der gleiche Präfix mit einem anderen Namespace Name belegt werden an verschiedenen Stellen im Dokument.
- **SHOULD**: Namespaces sollten nicht undeklariert werden, d.h. Namespace-Deklarationen aufgehoben werden, so dass sie an manchen Stellen im Dokument gültig sind, an manchen daruntergelegenen Stellen aber nicht (in XML Namespace 1.0 [XMLNS] ist das Undeklarieren nur erlaubt für den Default Namespace, in XML Namespaces 1.1 [XMLNS11] auch erlaubt für mit Präfix deklarierte Namespaces).

8.2 Namespace Präfixe

An sich haben Namespace Präfixe nur eine lokale Bedeutung, sie werden verwendet, um in einem XML Dokument den Bezug zwischen der Namespace-Deklaration (`xmlns:html="http://www.w3.org/1999/xhtml"`) und einem qualifizierten Namen (`<html:title>`) herzustellen. Auf die Präfixe kann sogar ganz verzichtet werden, wenn nur ein Namespace verwendet wird und dieser über die Deklaration des Default Namespace (`xmlns="http://www.w3.org/1999/xhtml"`) verwendet wird, so dass die Elemente ohne Präfix (`<title>`) verwendet werden können. Bei der Verwendung verschiedener Namespaces in einem Dokument wird es aber umständlich und unübersichtlich, den Default Namespace an verschiedenen Stellen im Dokument je nach Bedarf umzudefinieren, so dass spätestens dann Präfixe verwendet werden.

8.2.1 Problemstellung

Die Wahl des Präfixes ist lokal und kann vom Erzeuger eines XML Dokuments vorgenommen werden. "Sprechende" Namespace Präfixes sind die Regel und machen ein Dokument einfacher lesbar. Eine gleichartige Wahl des Präfixes für alle Dokumente, die Namen eines bestimmten Namespaces verwenden, macht die Dokumente einfacher wieder erkennbar und einfacher vergleichbar.

8.2.2 Empfehlungen

- **SHOULD**: Namespaces sollten mit Ihren empfohlenen oder üblichen Namespace Präfixen (soweit definiert oder etabliert) verwendet werden. Abweichungen sollten nur in Ausnahmen (z.B. Konflikt zweier Präfixe) vorgenommen werden.
- **SHOULD**: Bestehen keine definierten oder etablierten Präfixe, so sollte ein Name gewählt werden, der den Namespace möglicht kurz und prägnant referenziert.
- **MAY**: Falls es in einem Dokument so etwas wie einen "hauptsächlichen" Namespace gibt, so kann dieser als Default Namespace deklariert werden, was das Dokument kürzer und damit übersichtlicher macht.

9 Die Sprache von Inhalten

XML Dokumente verwenden Markup, d.h. sie kombinieren Struktur-Informationen (Elemente und Attribute) mit Inhalt (Element-Inhalte und Attributwerte). Zur Sprache des Markup sind die Empfehlungen in Abschnitt 7 gegeben worden, der vorliegende Abschnitt beschäftigt sich mit der Sprache von Inhalten.

9.1 Textuelle Inhalte

XML-Dokumente enthalten an vielen Stellen Text als Inhalt, im Prinzip an all denen Stellen, an denen der Inhalt durch das Schema nicht auf stärker festgelegt Typen (wie Zahlen, Datumsdarstellungen oder die in Abschnitt 9.2 behandelten Wertaufzählungen) eingeschränkt ist.

9.1.1 Problemstellung

Inhalte in XML Dokumenten können in unterschiedlichen Sprachen auftreten, und mit dem `xml:lang` Attribut (http://www.w3.org/TR/REC-xml/#sec-lang-tag) definiert XML einen Mechanismus, wie die Sprache von Inhalten angegeben werden kann. Das `xml:lang` Attribut muss in einem Schema (z.B. einer DTD oder einem XML Schema) definiert werden wie jedes andere Attribut auch, aber durch seine Herkunft aus dem XML Namespace selber ist es über alle Schemas hinweg identifizierbar als das Attribut zur Sprachidentifizierung.

9.1.2 Empfehlungen

- **MUST**: Sprach-Codierungen sind im `xml:lang` Attribut anzugeben.
- **MUST**: Sprach-Codierungen sind gemäss RFC 3066 [RFC3066] darzustellen. Applikationen müssen die zweistelligen ISO 639 [ISO639] Codierungen (de, fr, it, rm, en) oder die kombinierten ISO 639/3166 [ISO3166] Codierungen (de-CH, en-US) verwenden.
- **SHOULD**: Die detaillierten ISO 639/3166 Codierungen (de-CH, en-US) sind, soweit angebracht, den weniger detaillierten ISO 639 Codierungen (de, fr, en) vorzuziehen.

9.2 Aufzählungswerte

Im Gegensatz zu den in Abschnitt 9.1 beschriebenen textuellen Inhalten kann es auch durch das Schema vorgegebene Inhalte geben, insbesondere neben sprachunabhängigen Werten wie Zahlen oder Daten auch Aufzählungen, deren Werte in einer bestimmten Sprache definiert sind.

9.2.1 Problemstellung

Aufzählungen legen ähnlich wie die in Abschnitt 7 beschriebenen Namen von Elementen und Attributen Namen fest, die in Dokumenten verwendet werden. Aus diesem Grund sollten sich diese Werte sollten sich an der Sprache des Schemas orientieren, und die folgenden Empfehlungen sind entsprechen denen aus Abschnitt 7.1.2.

9.2.2 Empfehlungen

- **SHOULD**: Die Sprache der Aufzählungswerte ist Englisch
- **MAY**: Falls im Schema Aufzählungswerte verwendet werden, die eine sprachspezifische Bedeutung haben, die sich nicht übersetzen lässt (z.B. juristische Fachwörter), so können diese in der Sprache geschrieben werden, für die die Bedeutung erhalten bleiben muss.
- **SHOULD NOT**: Die obige Ausnahmeregelung sollte nicht dafür verwendet werden, um Schemas dreisprachig parallel zu führen.

Teil II: XML Schemas

10 Kennzeichnungen im XML Schema

XML Schemas können in sehr einfacher Form benutzt werden, insbesondere ist es nicht notwendig, einen Namespace für die definierten Namen anzugeben. Des Weiteren muss auch keinerlei Versionierungsinformation angegeben werden, XML Schema bietet hier keine Unterstützung für Versionierung. Die folgenden Richtlinien behandeln die damit zusammenhängenden Aspekte

10.1 Namespaces

XML Namespaces (wie Abschnitt 8 beschrieben) können durch beliebige Mechanismen beschrieben werden, im besonderen durch ein XML Schema.

10.1.1 Problemstellung

Jedes Schema, das in einem offenen Umfeld benutzt wird, sollte einen Namespace definieren, und dadurch die im Schema definierten Namen eindeutig referenzierbar machen. Die Struktur von Namespace Namen ist im entsprechenden eCH-Dokument [eCH-0033] näher beschrieben.

10.1.2 Empfehlungen

- MUST: Jedes XML Schema muss einen XML Namespace definieren
- MUST: Auf den von einem Schema definierten Namespace muss mit dem `targetNamespace` Attribut des `xs:schema` Elementes verwiesen werden.

10.2 Versionierung

Schemas verändern sich im Laufe der Zeit, sie werden korrigiert, erweitert, neuen Bedürfnissen angepasst, und sollten trotzdem zuverlässig anwendbar sein. In einem offenen Umfeld muss man mit der Koexistenz verschiedener Versionen, Instanzen verschiedener Versionen, und Software-Komponenten mit Support für verschiedene Versionen rechnen. Die folgenden Richtlinien definieren dafür Minimalanforderungen.

10.2.1 Problemstellung

Gemäss eCH-Dokument [eCH-0033] wird die Versionierung von Major Versions (also von Versionen, die nicht rückwärtskompatibel sind) über den Namespace Namen geführt. Damit bleibt als offenes Problem die Versionierung von Minor Versions. XML Schema kennt zwar ein `version` Attribut auf dem `xs:schema` Element, dieses hat jedoch keine definierte Bedeutung und ist vor allem auch nicht in der Instanz sichtbar. Aus diesem Grund definiert eCH ein eigenes Schema (mit einem eigenen Namespace) mit einem `minorVersion` Attribut, dass in allen eCH Schemas importiert werden sollte. Dieses Schema dient dazu, die Minor Version in eCH Dokumenten in einer standardisierten Weise

erkennen zu können. Dabei ist es Anwendungen freigestellt, ob sie die Minor Version als einfache Nummer führen, oder als weiter strukturierten Identifier (z.B. "2.3.5" als Minor Version), der es erlaubt, eine noch detailliertere Markierung von Versionen vorzunehmen.

10.2.2 Empfehlungen

- **MUST**: Schemas müssen so geschrieben sein, dass die Minor Version auf dem Document Element mittels eines Attributes angegeben werden muss, und dieses Attribut muss auf `use="required"` gesetzt sein.
- **SHOULD**: Für das oben verlangte Attribut sollte das `minorVersion` Attribut aus dem `http://www.ech.ch/xmlns/minorversion/v1` Namespace verwendet werden. Dieser Namespace ist unter dem genannten Namespace Namen dokumentiert.

11 Namen von Schema-Komponenten

Abschnitt 7 beschreibt Richtlinien für die Form von Namen in XML Dokumenten. In XML Schema gibt es Namen für alle benannten Komponenten, also Elemente, Attribute, Notationen, Typen, Identity Constraints und Named Groups (`http://www.w3.org/TR/xmlschema-1/#concepts-data-model`). Generell betrachtet sind Namen von Schema-Komponenten auch Namen innerhalb von XML Dokumenten, und aus diesem Grund gelten die in Abschnitt 7 aufgeführten Richtlinien für die Sprache von Namen (Abschnitt 7.1) und die Form von Namen (Abschnitt 7.2) auch für Namen von Schema-Komponenten.

11.1 Namen von Typen

11.1.1 Problemstellung

Typen in XML Schema sind entweder Simple oder Complex Types, die als Grundlage für andere Typen (Typableitung) oder für Elemente oder Attribute verwendet werden. Typen erhalten einen Namen (falls sie nicht anonym verwendet werden, also direkt in einer anderen Definition enthalten sind) und werden über diesen Typnamen referenziert.

11.1.2 Empfehlungen

- **MUST**: Typen in XML Schema werden mit den gleichen Namensregeln wie Element- und Attributnamen definiert.
- **MUST**: Am Ende des Namens ist "Type" anzubringen.
- **SHOULD**: Der Name des Typs sollte auf seine Verwendung hinweisen. Wird ein Typ ausschliesslich oder hauptsächlich für ein Element/Attribut verwendet, so sollte er dessen Namen (mit dem Zusatz "Type") tragen.

Beispiel:

- personType → richtig
- personTyp → falsch ("Typ" statt "Type")
- CustomerType → falsch (muss mit Kleinbuchstaben beginnen)

11.2 Namen von Named Groups

11.2.1 Problemstellung

Named Groups in XML Schema sind wieder verwendbare Komponenten, die Teile eines Complex Types enthalten können, entweder Attributmengen ("Attribute Groups"), oder Teile von Inhaltsmodellen ("Named Model Groups").

11.2.2 Empfehlungen

- **MUST**: Named Groups in XML Schema werden mit den gleichen Namensregeln wie Element- und Attributnamen definiert.

11.2.3 Empfehlungen Attribut Gruppen

- **MUST**: Bei Attribut-Gruppen wird am Ende des Namens "AttributeGroup" angebracht.

Beispiel:

- genericAttributeGroup → richtig
- transactionAttributGruppe → falsch ("AttributGruppe" statt "AttributeGroup")
- TransactionAttributeGroup → falsch (muss mit Kleinbuchstaben beginnen)

11.2.4 Empfehlungen Named Model Gruppen

- **MUST**: Bei Named Model Groups wird am Ende des Namens "Group" angebracht.

Beispiel:

- keyValueGroup → richtig
- keyValueGruppe → falsch ("Gruppe" statt "Group")
- TransactionFragmentGroup → falsch (muss mit Kleinbuchstaben beginnen)

12 Dokumentation

12.1 Problemstellung

Ein XML Schema definiert nur die syntaktische Struktur von XML Dokumenten, es sagt nichts über deren Bedeutung aus. Die Bedeutung der Syntax sollte zum Teil durch Dokumentation im Schema selber (sofern es die Bedeutung der kleinsten Einheiten und der daraus zusammengesetzten Komponenten ist), und darüber in begleitender Dokumentation (wenn es um Zusammenhänge und den grösseren Anwendungskontext geht) erfolgen. Für die Dokumentation im Schema können Kommentare im XML Schema in XML Kommentare oder in spezielle XML Schema Elemente verwendet werden, beide Arten von Kommentaren haben keine formale Bedeutung für das XML Schema.

Die Sprache der Dokumentation ist dem Kontext entsprechend zu wählen. Je nach Verwendungszweck des Schemas ist es nicht immer nötig und/oder finanziell vorteilhaft, alle Sprachgruppen zu unterstützen. Es ist das für das Anwendungsfeld angemessenes Mass für die Sprache/n der Dokumentation zu finden. Folgende Anwendungskontexte wurden identifiziert:

- Projektübergreifend (beteiligt sind mehrere Stellen der öffentlichen Verwaltung)
- Projektspezifisch (lokalisiert)
- Technisch

12.2 Empfehlungen

- **MUST**: Der `targetNamespace` des Schemas muss auf eine Namespace Beschreibung gemäss [eCH-0033] zeigen, über die das Schema selber, begleitende Dokumentation, und weitere relevante Ressourcen zugänglich gemacht sind.
- **MUST**: Die Dokumentation in XML Schemas ist mittels `xs:annotation` Elementen und darin enthaltenen `xs:documentation` Elementen zu führen.
- **MUST NOT**: Dokumentation in XML Schemas darf nicht in XML Kommentaren (`<!-- ... -->`) geführt werden.
- **MUST**: Alle globalen Komponenten (d.h. die auf dem Top-Level des Schemas befindlichen, d.h. die direkten Kinder des `xs:schema` Elements) sind zu dokumentieren.
- **SHOULD**: Alle vorhandenen Komponenten sollten dokumentiert werden, soweit sie nicht trivial oder selbsterklärend sind.
- **MUST**: Die `xs:documentation` Elemente müssen durch `xml:lang` Attribute hinsichtlich der Dokumentationssprache beschrieben werden (siehe Abschnitt 9 zu allgemeinen Regeln zur Sprachmarkierung in XML Dokumenten). Ist die Dokumentation einsprachig, so genügt die Codierung der Dokumentationssprache im `xml:lang` Attribut des `xs:schema` Elements.

- **MUST**: Schemas im Schweiz-weiten Kontext sind mehrsprachig zu dokumentieren. Die Sprache jeder Dokumentation ist im `xml:lang` Attribut des `xs:documentation` Elementes anzugeben.
    - Deutsch (zwingend falls nicht Englisch)
    - Französisch (zwingend falls nicht Englisch)
    - Italienisch (optional)
    - Es ist ebenso möglich, die Dokumentation ausschliesslich auf Englisch zu führen.
    - Unabhängig von der Wahl der Dokumentationssprache isr darauf zu achten, dass diese Wahl konsistent für die gesamte Dokumentation im Schema eingehalten wird, so dass die Dokumentationssprache für das gesamte Schema einheitlich ist.
- **MAY**: Projektspezifische (lokalisierte) Schemas können in der Projektsprache dokumentiert werden.
- **MAY**: Technische Schemas können zusätzlich oder ausschliesslich in Englisch dokumentiert werden.
- **MUST**: Alle Schemas müssen eine Top-Level Beschreibung besitzen. Die Beschreibung muss eine kurze prägnante Einleitung unter Angabe des Verwendungszwecks und des Kontextes des Schemas bieten.

Teil III: XML Schema Instanzen

13 Schema-Information in Instanzen

XML Dokumente, die Instanzen eines XML Schemas sind, sollten diese Information in einer Art und Weise enthalten, die den Umgang mit den Instanzen möglichst einfach gestaltet. Dies betrifft zum einen die Frage, wie die Zugehörigkeit zu einem Schema überhaupt signalisiert wird, und des weiteren die Frage, wie Versionierungsinformation behandelt wird.

13.1 Zugehörigkeit zu einem XML Schema

Die Zugehörigkeit eines XML Dokuments zu einem Schema kann im Prinzip über zwei Mechanismen sichtbar werden: zum einen über den verwendeten Namespace (falls, wie in Abschnitt 10.1 empfohlen, das Schema einen Namespace Namen definiert), und zum anderen über die Möglichkeit des Dokumentes, mittels des `xsi:schemaLocation` Attributes direkt auf das Schema zu verweisen.

13.1.1 Problemstellung

Mit dem Namespace Namen und dem `xsi:schemaLocation` Attribut gibt es praktisch zwei alternative Methoden, die Schema-Zugehörigkeit eines XML Dokumentes zu beschreiben, wobei die Verwendung des Namespaces zwingend notwendig ist, damit das XML Dokument korrekt ist.

Der Namespace Namen ist ein abstrakter Name für das verwendete Vokabular und verweist nicht notwendigerweise auf eine existierende Ressource. Im Rahmen von eCH legt das Dokument [eCH-0033] jedoch fest, dass der Namespace Name auf eine Beschreibung des Schemas verweisen muss, und wie von dieser Beschreibung auf das Schema verwiesen wird. Konkret wird in Applikationen jedoch fast immer so gearbeitet, dass die Applikation den Namespace Namen erkennt, als einen bekannten Namespace Namen identifiziert, und das lokal vorhandene Schema für die Validierung verwendet. In welcher Weise das Schema lokal gespeichert wird (fest konfiguriert oder in einem Cache, der dynamisch erweitert werden kann) ist vollständig in der Verantwortung des Applikationsdesigners.

13.1.2 Empfehlungen

- **MUST**: XML Dokumente müssen den verwendeten Namespace deklarieren (dies sollte gemäss den in Abschnitt 8 aufgeführten Richtlinien erfolgen).
- **SHOULD NOT**: XML Dokumente sollten nicht mittels des `xsi:schemaLocation` Attributes auf das Schema verweisen. Der Namespace Namen sollte die relevante Identifikation des Schemas sein.
- **SHOULD NOT**: Enthält eine Instanz ein `xsi:schemaLocation` Attribut, so sollte sich eine Applikation nicht darauf verlassen, unter der angegebenen URI tatsächlich das XML Schema aufzufinden, sondern der Interpretation den Namespace Namens Priorität geben.

13.2 Versionierung

Ein weiterhin offenes Problem ist die Frage, wie die Minor Version in der Instanz vermerkt ist. Mit dem Namespace Namen ist die Major Version abgedeckt, weil gemäss [eCH-0033] die Major Version Teil des Namespace Namen sein sollte.

13.2.1 Problemstellung

Die Minor Version sollte in der Instanz vermerkt sein, damit Applikationen auch abhängig von der Minor Version (und nicht nur abhängig von der im Namespace Namen vermerkten Major Version) die richtige Verarbeitung einleiten können. Die folgenden Empfehlungen sind insbesondere im Zusammenhang mit den Empfehlungen aus Abschnitt 10.2 zu lesen.

13.2.2 Empfehlungen

- **MUST**: XML Dokumente, die Instanzen von XML Schemas sind, müssen die komplette Versionsinformation enthalten. Da im Fall von eCH XML Schemas nur die Major Version im Namespace Namen enthalten ist, muss die Minor Version über ein Attribut des Document Element angegeben werden.
- **SHOULD**: Für das oben verlangte Attribut für die Minor Version Number sollte das `minorVersion` Attribut aus dem `http://www.ech.ch/xmlns/minorversion/v1` Namespace verwendet werden. Dieser Namespace ist unter dem genannten Namespace Namen dokumentiert.

Weitere Empfehlungen zu diesem Thema finden sich in Abschnitt 10.

14 Verwendung von Schema-Komponenten

14.1 Namen von Elementen und Attributen

Elemente und Attribute in XML Schema sind entweder global definiert (d.h. auf dem Top Level des Schemas, also direkt unterhalb des `xs:schema` Elements) oder lokal (d.h. als Teil anderer Komponenten und demnach in diesen definiert). Globale Namen eines Schemas müssen immer mit qualifizierten Namen (mit dem `targetNamespace` des Schemas qualifiziert) verwendet werden, während lokale Namen per Default unqualifiziert (also aus keinem Namespace) verwendet werden.

ACHTUNG: Der Default Namespace bezieht sich nur auf Elemente, nicht aber auf Attribute. Dies heisst, dass Attribute, die mit einem Namespace Namen referenziert werden sollen (weil sie global definiert wurden oder im XML Schema `attributeFormDefault="qualified"` gesetzt wurde), immer mit einem Präfix versehen werden müssen. Im Allgemeinen sind Attribute jedoch lokal definiert und es ist `attributeFormDefault="unqualified"` (der Default) gültig, so dass die Attribute unqualifiziert verwendet werden können.

Beispiele:

```
<el1 attr1="x" xmlns="http://example.com/"/>
```

```
<ex:el2 attr2="x" xmlns:ex="http://example.com/"/>
<ex:el3 ex:attr3="x" xmlns:ex="http://example.com/"/>
```

In diesem Beispiel ist `el1` aus dem Namespace `http://example.com/` und `attr1` aus keinem Namespace, auch `el2` ist aus dem Namespace `http://example.com/` und `attr2` aus keinem Namespace, während auch `el3` aus dem Namespace `http://example.com/` ist, aber in diesem Fall `attr3` ebenso. Der dritte Fall ist selten und kommt fast nur in Fällen vor, in denen Attribute verwendet werden, die auf vielen Elementen definiert sind, daher eine gewisse Eigenständigkeit haben, und aus einem eigenen Namespace stammen, wie z.B. das in Abschnitt 9 beschriebene `xml:lang` Attribut.

14.1.1 Problemstellung

Durch die `elementFormDefault` und `attributeFormDefault` Attribute des Schemas kann angegeben werden, in welcher Form lokal definierte Namen des Schemas verwendet werden müssen (global definierte Namen müssen immer qualifiziert referenziert werden). Diese Attribute haben also einen grossen Einfluss darauf, wie Instanzen eines Schemas hinsichtlich der Verwendung von Namespaces auszusehen haben.

14.1.2 Empfehlungen

- **MUST**: Ist `elementFormDefault="qualified"` angegeben, so kann (falls dieses Schema das für dieses Dokument hauptsächlich verwendete ist) der `targetNamespace` des Schemas als Default Namespace deklariert werden und alle Elemente ohne Präfix verwendet werden.
- **SHOULD**: Ist `attributeFormDefault="qualified"` angegeben, so sollte der `targetNamespace` des Schemas mit Präfix deklariert werden und die lokal definierten Attribute mit diesem Präfix versehen werden.

14.2 Defaultwerte für Elemente und Attribute

14.2.1 Problemstellung

XML Schema erlaubt es, Defaultwerte für Elemente und Attribute anzugeben (in DTDs ist dieser Mechanismus nur für Attribute vorhanden). Auf diese Weise können im Schema Werte angegeben werden, die im Dokument verwendet werden, falls die entsprechende Komponente im Dokument nicht vorhanden ist (bzw. vorhanden aber leer, dies gilt nur für Elemente).

Defaultwerte machen eine Validierung gemäss dem Schema zwingend notwendig, weil sich die Interpretation des Dokuments durch die Validierung ändert (die Defaultwerte kommen als Resultat der Validierung zu den im Dokument selbst vorgefundenen Informationen hinzu). Aus diesem Grund sind Defaultwerte ein recht problematischer Mechanismus, der grosse Auswirkungen auf das Verarbeitungsmodell hat.

14.2.2 Empfehlungen

- **SHOULD**: XML Dokumente sollten sich nicht auf in einem Schema definierte Defaultwerte verlassen, sondern Werte immer explizit angeben, auch wenn die Werte identisch mit im Schema definierten Defaultwert sind.
- **MUST**: Verlässt sich der Erzeuger eines XML Dokumentes auf Defaultwerte, indem er diese Werte in Dokumenten nicht angibt, so darf er das nur tun, falls hinsichtlich der weiteren Verarbeitung des Dokumentes sichergestellt ist, dass diese Schema-basiert erfolgt und daher die Defaultwerte in Betracht ziehen wird.

15 Haftungsausschluss/Hinweise auf Rechte Dritter

eCH-Standards, welche der Verein **eCH** dem Benutzer zur unentgeltlichen Nutzung zur Verfügung stellt, oder welche **eCH** referenziert, haben nur den Status von Empfehlungen. Der Verein **eCH** haftet in keinem Fall für Entscheidungen oder Massnahmen, welche der Benutzer auf Grund dieser Dokumente trifft und / oder ergreift. Der Benutzer ist verpflichtet, die Dokumente vor deren Nutzung selbst zu überprüfen und sich gegebenenfalls beraten zu lassen. **eCH**-Standards können und sollen die technische, organisatorische oder juristische Beratung im konkreten Einzelfall nicht ersetzen.
In **eCH**-Standards referenzierte Dokumente, Verfahren, Methoden, Produkte und Standards sind unter Umständen markenrechtlich, urheberrechtlich oder patentrechtlich geschützt. Es liegt in der ausschliesslichen Verantwortlichkeit des Benutzers, sich die allenfalls erforderlichen Rechte bei den jeweils berechtigten Personen und/oder Organisationen zu beschaffen.
Obwohl der Verein **eCH** all seine Sorgfalt darauf verwendet, die **eCH**-Standards sorgfältig auszuarbeiten, kann keine Zusicherung oder Garantie auf Aktualität, Vollständigkeit, Richtigkeit bzw. Fehlerfreiheit der zur Verfügung gestellten Informationen und Dokumente gegeben werden. Der Inhalt von **eCH**-Standards kann jederzeit und ohne Ankündigung geändert werden.
Jede Haftung für Schäden, welche dem Benutzer aus dem Gebrauch der **eCH**-Standards entstehen ist, soweit gesetzlich zulässig, wegbedungen.

16 Urheberrechte

Wer **eCH**-Standards erarbeitet, behält das geistige Eigentum an diesen. Allerdings verpflichtet sich der Erarbeitende mittels spezieller, schriftlicher Vereinbarung, sein betreffendes geistiges Eigentum oder seine Rechte an geistigem Eigentum anderer, sofern möglich, den jeweiligen Fachgruppen und dem Verein **eCH** kostenlos zur uneingeschränkten Nutzung und Weiterentwicklung im Rahmen des Vereinszweckes zur Verfügung zu stellen.
Die von den Fachgruppen erarbeiteten Standards können unter Nennung der jeweiligen Urheber von **eCH** unentgeltlich und uneingeschränkt genutzt, weiterverbreitet und weiterentwickelt werden.
eCH-Standards sind vollständig dokumentiert und frei von lizenz- und/oder patentrechtlichen Einschränkungen. Die dazugehörige Dokumentation kann unentgeltlich bezogen werden.
Diese Bestimmungen gelten ausschliesslich für die von **eCH** erarbeiteten Standards, nicht jedoch für Standards oder Produkte Dritter, auf welche in den **eCH**-Standards Bezug genommen wird. Die Standards enthalten die entsprechenden Hinweise auf die Rechte Dritter.

Anhang A – Beispiel XML-Schema und XML-Dokument

Im folgenden XML Schema sind einige (aber nicht alle) der im vorliegenden Dokument enthaltenen Empfehlungen demonstriert. Die Dokumentation ist nur im ersten `xs:annotation` Element zweisprachig ausgeführt und mit Text versehen, in den anderen Dokumentations-Elementen ist der Text weggelassen worden, um das Schema kürzer und damit besser lesbar zu machen. Im folgenden Schema werden die folgenden Empfehlungen demonstriert:

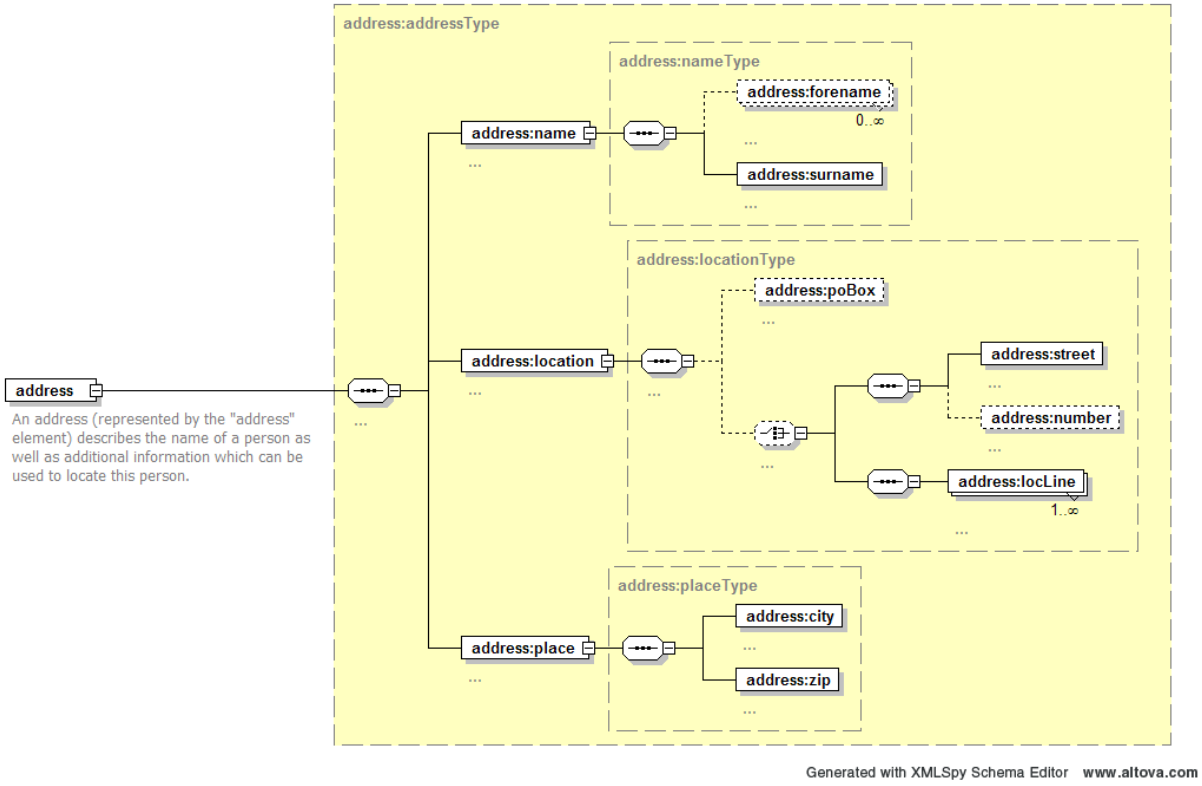
- XML Dokumente als Instanzen von Klassen (Abschnitt 3): Es wurde ein XML Schema definiert, um zu beschreiben, wie Dokumente strukturiert werden.
- Character Encodings (Abschnitt 5.2): Das XML Schema verwendet UTF-8, da es keine speziellen Anforderungen an den verwendeten Zeichensatz gibt.
- Zeilenumbrüche(Abschnitt 5.3): Das XML Schema wurde mit Zeilenumbrüchen einfacher lesbar formatiert.
- Versionierung (Abschnitt 5.5): Auf eine Minor Version wurde in diesem Beispiel verzichtet, die Major Version ist über den Namespace Name markiert.
- Die Sprache der Namen (Abschnitt 7.1): Die definierten Namen sind in englische Namen.
- Namenskonventionen (Abschnitt 7.2): Die Namen sind mit Kleinbuchstaben geschrieben und enthalten keine Sonderzeichen.
- Namen von Typen (Abschnitt 11.1): Die Typen haben Namen, die mit dem Suffix "Type" enden.
- Dokumentation (Abschnitt 12): Die wesentlichen Komponenten des XML Schemas sind dokumentiert (die Dokumentation wurde in fast allen Fällen aus Gründen der Übersicht für dieses Beispiel auf "..." reduziert).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:address="http://www.example.ch/xmlns/address/v1"
targetNamespace="http://www.example.ch/xmlns/address/v1" elementFormDefault="qualified">
	<xs:element name="address" type="address:addressType">
		<xs:annotation>
			<xs:documentation xml:lang="en">An address (represented by the "address"
element) describes the name of a person as well as additional information which can be used to
locate this person.</xs:documentation>
			<xs:documentation xml:lang="de">Eine Anschrift (durch das "address"
Element repräsentiert) beschreibt den Namen einer Person und die Auffindbarkeit der Person
durch  zusätzliche Information.</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:complexType name="nameType">
		<xs:sequence>
			<xs:element name="forename" minOccurs="0" maxOccurs="unbounded">
				<xs:annotation>
					<xs:documentation>...</xs:documentation>
				</xs:annotation>
			</xs:element>
			<xs:element name="surname">
				<xs:annotation>
					<xs:documentation>...</xs:documentation>
				</xs:annotation>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="placeType">
		<xs:sequence>
```

```
                        <xs:element name="city">
                                <xs:annotation>
                                        <xs:documentation>...</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="zip">
                                <xs:annotation>
                                        <xs:documentation>...</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="locationType">
                <xs:sequence>
                        <xs:annotation>
                                <xs:documentation>...</xs:documentation>
                        </xs:annotation>
                        <xs:element name="poBox" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>...</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:choice minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>...</xs:documentation>
                                </xs:annotation>
                                <xs:sequence>
                                        <xs:element name="street">
                                                <xs:annotation>
                                                        <xs:documentation>...</xs:documentation>
                                                </xs:annotation>
                                        </xs:element>
                                        <xs:element name="number" minOccurs="0">
                                                <xs:annotation>
                                                        <xs:documentation>...</xs:documentation>
                                                </xs:annotation>
                                        </xs:element>
                                </xs:sequence>
                                <xs:sequence>
                                        <xs:element name="locLine" maxOccurs="unbounded">
                                                <xs:annotation>
                                                        <xs:documentation>...</xs:documentation>
                                                </xs:annotation>
                                        </xs:element>
                                </xs:sequence>
                        </xs:choice>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="addressType">
                <xs:sequence>
                        <xs:annotation>
                                <xs:documentation>...</xs:documentation>
                        </xs:annotation>
                        <xs:element name="name" type="address:nameType">
                                <xs:annotation>
                                        <xs:documentation>...</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="location" type="address:locationType">
                                <xs:annotation>
                                        <xs:documentation>...</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="place" type="address:placeType">
                                <xs:annotation>
                                        <xs:documentation>...</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

Um das Beispiel-Schema besser verständlich zu machen, findet sich im Folgenden als Erläuterung eine graphische Darstellung des Beispiel-Schemas. Diese Darstellung wurde mit dem kommerziellen Tool "XML Spy" erzeugt, es gibt eine Reihe kommerzieller Tools, um solche graphischen Darstellungen zu erzeugen. Generell ist es der Übersichtlichkeit und dem Verständnis eines XML Schema sehr dienlich, eine graphische Darstellung zu haben, da die XML Syntax eines XML

Schemas schwer zu lesen ist und zudem die inhaltlichen Zusammenhänge des XML Schemas unzureichend repräsentiert.



Zu dem oben aufgeführten XML Schema ist das folgende XML Dokument eine Beispiel-Instanz, die auf einfache Art die im XML Schema definierten Strukturen verwendet. Das XML Dokument demonstriert eine Reihe der aufgestellten Empfehlungen, die in der folgenden Liste aufgeführt sind:

- XML Versionen (Abschnitt 4): Es wird XML Version 1.0 verwendet, da keine Notwendigkeit besteht, die neuen Features von XML 1.1 zu verwenden.
- Character Encodings (Abschnitt 5.2): Das Dokument verwendet UTF-8, da es keine speziellen Anforderungen an den verwendeten Zeichensatz gibt.
- Zeilenumbrüche(Abschnitt 5.3): Das Schema wurde mit Zeilenumbrüchen einfacher lesbar formatiert.
- Versionierung (Abschnitt 5.5): Auf eine Minor Version wurde in diesem Beispiel verzichtet, die Major Version ist über den Namespace Name markiert.
- Entities (Abschnitt 6): In diesem Beispiel sind keine externen Daten notwendig, und alle internen Daten können direkt als UTF-8 Zeichen angegeben werden (insbesondere die Umlaute, die nicht als Internal Entities oder Character References benutzt werden).
- Namespace Deklarationen (Abschnitt 8.1): Die Deklaration des verwendeten Namespaces geschieht im Document Element.
- Namespace Präfixe (Abschnitt 8.2): Da das Dokument nur Namen aus einem Namespace verwendet, wird dieser als Default Namespace deklariert.

- Zugehörigkeit zu einem XML Schema (Abschnitt 13.1): Durch die Namespace-Deklaration kann eine Applikation feststellen, dass die zu diesem Namespace gehörigen Definitionen in einem XML Schema definiert sind. Das Dokument verweist nicht direkt auf das XML Schema mittels des `xsi:schemaLocation` Attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<address xmlns="http://www.example.ch/xmlns/address/v1">
      <name>
            <forename>Hans</forename>
            <forename>Peter</forename>
            <surname>Rüdisüli</surname>
      </name>
      <location>
            <poBox>Postfach 776</poBox>
            <street>Lägernstrasse</street>
            <number>77b</number>
      </location>
      <place>
            <city>Bern</city>
            <zip>3005</zip>
      </place>
</address>
```

Anhang B – Referenzen

[eCH-0033]	Erik Wilde, *Beschreibung von XML Namespaces*, Technical Report eCH-0033, eCH, 2005 (in Entwicklung).
[eCH-0035]	Erik Wilde, *Design von XML Schemas*, Technical Report eCH-0035, eCH, 2005 (in Entwicklung).
[eCH-0036]	Erik Wilde, *Modellierung für den XML-orientierten Datenaustausch*, Technical Report eCH-0036, eCH, 2005 (in Entwicklung).
[ISO639]	International Organization for Standardization, *Codes for the Representation of Names of Languages*, ISO 639, July 2002.
[ISO3166]	International Organization for Standardization, *Codes for the Representation of Names of Countries and their Subdivisions*, ISO 3166, November 2001.
[ISO19757-3]	International Organization for Standardization, *Information Technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*, to be published as ISO 19757-3, 2005 (in Entwicklung).
[RELAXNG]	James Clark, Murata Makoto, *RELAX NG Specification*, Organization for the Advancement of Structured Information Standards (OASIS) Committee Specification, December 2001. http://www.oasis-open.org/committees/relax-ng/spec-20011203.html
[RFC2045]	Ned Freed and Nathaniel Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Internet RFC 2045, November 1996. http://www.ietf.org/rfc/rfc2045.txt
[RFC2119]	Scott O. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,

	Internet RFC 2119, March 1997. ftp://ftp.isi.edu/in-notes/rfc2119.txt
[RFC3066]	Harald Tveit Alvestrand, *Tags for the Identification of Languages*, Internet RFC 3066, January 2001. ftp://ftp.isi.edu/in-notes/rfc3066.txt
[RFC3986]	Tim Berners-Lee, Roy Fielding, Larry Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, Internet RFC 3986, January 2005. ftp://ftp.isi.edu/in-notes/rfc3986.txt
[WebChar10]	Martin J. Dürst, François Yergeau, Richard Ishida, Misha Wolf, Tex Texin, *Character Model for the World Wide Web 1.0: Fundamentals*, World Wide Web Consortium, Recommendation REC-charmod-20050215, Februar 2005. http://www.w3.org/TR/2005/REC-charmod-20050215
[XHTML10Sec]	Steven Pemberton, *XHTML 1.0: The Extensible HyperText Markup Language (Second Edition)*, World Wide Web Consortium, Recommendation REC-xhtml1-20020801, August 2002. http://www.w3.org/TR/2002/REC-xhtml1-20020801
[XInclude]	Jonathan Marsh, David Orchard, *XML Inclusions (XInclude) Version 1.0*, World Wide Web Consortium, Recommendation REC-xinclude-20041220, December 2004. http://www.w3.org/TR/2004/REC-xinclude-20041220/
[XLink]	Steven J. DeRose, Eve Maler, David Orchard, *XML Linking Language (XLink) Version 1.0*, World Wide Web Consortium, Recommendation REC-xlink-20010627, Juni 2001. http://www.w3.org/TR/2001/REC-xlink-20010627
[XML10Third]	Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau, *Extensible Markup Language (XML) 1.0 (Third Edition)*, World Wide Web Consortium, Recommendation REC-xml-20040204, February 2004. http://www.w3.org/TR/2004/REC-xml-20040204
[XML11]	Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan, *XML 1.1*, World Wide Web Consortium, Recommendation REC-xml11-20040204, February 2004. http://www.w3.org/TR/2004/REC-xml11-20040204
[XMLNS]	Tim Bray, Dave Hollander, Andrew Layman, *Namespaces in XML*, World Wide Web Consortium, Recommendation REC-xml-names-19990114, January 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114
[XMLNS11]	Tim Bray, Dave Hollander, Andrew Layman, Richard Tobin, *Namespaces in XML 1.1*, World Wide Web Consortium, Recommendation REC-xml-names11-20040204, February 2004. http://www.w3.org/TR/2004/REC-xml-names11-20040204
[XMLSchema1]	Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, *XML Schema Part 1: Structures Second Edition*, World Wide Web Consortium, Recommendation REC-xmlschema-1-20041028, Oktober 2004. http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[XMLSchema2]	Paul V. Biron, Ashok Malhotra, *XML Schema Part 2: Datatypes Second Edition*, World Wide Web Consortium, Recommendation REC-xmlschema-2-20041028, Oktober 2004. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

Anhang C – Mitarbeit & Überprüfung

Hans-Ulrich Bucher, Avataris AG
Remo Dick, ISC EJPD
Claude Eisenhut, Eisenhut Informatik
Urs Gähler, VRSG
Jürg Hotz, Kanton Thurgau
Adrian K. Keller, SAG Software Systems AG
Willy Müller, ISB
Hubert Münst, Data-Factory
Patrick Ostertag, Etat de Fribourg
Cédric Perrenoud, ISC EJPD
Alexander Pinar, Unisys (Schweiz) AG
Fabian Probst, Fachhochschule Solothurn Nordwestschweiz
Hanspeter Salvisberg, Unisys (Schweiz) AG
Harald Schmid
Verena Sieber, T-Systems
Hans Ulrich Wiedmer, KOGIS - LT
Hansruedi Vock, BIT
Erik Wilde, ETH Zürich

Anhang D – Zuständigkeit und Mutationswesen

Für die Überarbeitung dieses Standards ist die eCH Fachgruppe XML-Standards zuständig.

Anhang F – Glossar

Ein ausführliches Online-Glossar ist unter http://dret.net/glossary/ zu finden.

ASCII	*American Standard Code for Information Interchange*	Der kleinste gemeinsame Nenner (fast) aller Zeichencodierungen (7-Bit Codierung), darauf aufbauende Codierungen erweitern ASCII z.B. um Umlaute (ISO 8859-1) oder ganz allgemein internationale Zeichen (Unicode).
Attribut		Attribute sind Elementen zugeordnet , sie enthalten zusätzliche Informationen zu einem Element.
Content Model		Bestimmt in einem Schema (DTD oder XML Schema) den erlaubten Inhalt eines Element-Typs (definiert damit die erlaubte Verwendung des Element-Typs).

DTD	*Document Type Definition*	Die im XML Standard selber definierte Schema Sprache für XML Dokumente, hat insbesondere Schwächen im Bereich der Datentypen.
Element		Grundlegendstes Strukturierungsmerkmal eines XML Dokuments, Nur Elemente können verschachtelt werden und erlauben damit die komplexen Strukturen, die in XML oftmals verwendet werden.
Entity		Ein Stück XML Text, kann definiert und referenziert werden (es gibt verschiedene Typen, die wichtigsten sind General und Parameter Entities).
HTML	*Hypertext Markup Language*	Dokumentenformat des WWW, basiert auf SGML und ist inkompatibel mit XML (aus diesem Grund gibt es mit XHTML eine XML-basierte Version von HTML).
Information Set		Das Datenmodell von XML, modelliert XML als Information Items mit Properties.
ISO 8859		8-Bit Codierung von Zeichen, weit verbreitet, aber inkompatibel mit UTF-8.
Markup		Die physische Form eines XML Dokuments (Serialisierung des Information Set).
Namespaces		Methode zur Benennung und Verwendung von Namensräumen in XML.
Processing Instruction		Verarbeitungsanweisungen in XML Dokumenten, nicht Teil des eigentlichen Inhalts, sondern eher Zusatzinformationen (Syntax: "<?name content?>").
Schema		Beschreibung einer Klasse von XML Dokumenten, die bekanntesten Dialekte sind DTD (Teil von XML selber) und XML Schema (separater Standard).
Unicode		Standard für die Referenzierung und Codierung sehr vieler Zeichen.
URI/URL	*Universal Resource Identifier/Locator*	Standard für die Adressierung von Ressourcen auf dem Web. besteht aus einem *Scheme Identifier* (z.B. "http:") und weiterer Information zur Adressierung.
UTF-8	*Unicode Transformation Format 8*	Die XML Default-Codierung von Unicode Zeichen, jedes ASCII Dokument ist auch ein UTF-8 Dokument (UTF-8 codiert Zeichen als 1-6 Bytes). Jede XML Software muss UTF-8 unterstützen.
UTF-16	*Unicode Transformation Format 16*	Eine Zeichencodierung, die jedes Unicode Zeichen als 16 oder 32 Bit codiert (alle geläufigen Zeichen werden als 16 Bit codiert). Jede XML Software muss UTF-16 unterstützen.
valid		Well-formed und den Einschränkungen einer DTD gehorchend.
well-formed		Ein XML Dokument ist well-formed, wenn es den XML Syntax-Regeln gehorcht.
XInclude	*XML Include*	Regelt die Einbindung von externen XML-Ressourcen in XML Dokumente.
XML	*Extensible Markup Language*	Sprache für die Repräsentation von baumstrukturierten Dokumenten und ihr Schema.
XML Schema		Schemasprache für XML, die deutlich leistungsfähiger ist als die in XML DTDs definierten Mechanismen.
XSLT	*XSL Transformations*	Teil von XSL, auf XML-Transformation spezialisierte Programmiersprache die es ermöglicht, ein XML Dokument in eine andere Repräsentation (XML, HTML, ...) zu transformieren.